Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/30/20230108

A search and rescue robot design based on LiDAR technology

Yichen Ding[1,3] and Fei Ye[2,4]

[1]Keystone Academy, Beijing, China
[2]Shanghai Jiao Tong University, Shanghai, China

[3]yichendingacademic@163.com
[4]yefei5212022@126.com

Abstract. For a long time, search and rescue operations during natural disasters and man-made catastrophes have been a major challenge. Due to the rapidly changing environment in disasters, deploying rescue teams for search missions entails significant risks. With advancements in technology, the latest innovations can be applied to search and rescue tasks to reduce these risks. LiDAR (Light Detection and Ranging) sensors can be installed on unmanned search and rescue vehicles to explore the space. This article utilizes solid-state LiDAR technology, along with various algorithms like SLAM (Simultaneous Localization and Mapping) and EKF (Extended Kalman Filter), to design a remotely controlled unmanned exploration vehicle. By capturing point cloud data, it enables modelling and recording of indoor or outdoor spaces, allowing for space exploration and the identification of trapped individuals and other important rescue-related information before rescue personnel enter the premises. This significantly reduces the risks and time involved in search and rescue operations. The prototype vehicle designed in this paper possesses the advantages of low cost and high flexibility, making it feasible for direct deployment after minor optimization. Finally, the author provides a summary and outlook for this research.

Keywords: LiDAR, rescue robotics, point-cloud, EKF, SLAM.

1. Introduction

It had been long speculated that novel technologies may be incorporated into rescue missions in critical situations such as earthquakes and tsunamis. In major earthquakes, it is possible to identify opportunities in using autonomous robotic exploration vehicles in exploring, charting, and modelling the interior spaces of collapsed buildings, or even to lo cate survivors, increasing the efficiency and safety of the rescue mission, which in turn improves the chances of survival and the safety of the emergency response personnel. This is a major improvement over the previous system of personnel plus rescue dogs since it decreases the risks involved in a fully manual extraction operation. A significant example of this may be seen in the Wenchuan earthquake of 2008, which measured 8.0 on the Richter Scale and caused major damage to structures in the vicinity of ground zero. According to Hakami et al. [1], for such events, there is a decreasing rate of survival as time spent before rescue increases, where the survival rate quickly drops off to less than 20% or even 5-10% after 72 hours, as illustrated by Figure 1 below. This is known as the Golden 72 Hours. Since the rate of survival drops significantly with time, it is important to extract trapped survivors shortly after the incident to ensure a high survival rate, preferably in under 3 days. However, search teams are often overwhelmed by the number of tasks needed to be done at the site. By utilizing the combinations of technologies listed above, it is possible to improve the overall speed of the extraction process by scanning and charting the structures before manual exploration, thus decreasing the fatality rate by enabling the extraction team to locate and extract survivors with higher efficiency.

Figure 1. Rescuing was difficult after the 2008 Wenchuan earthquake.

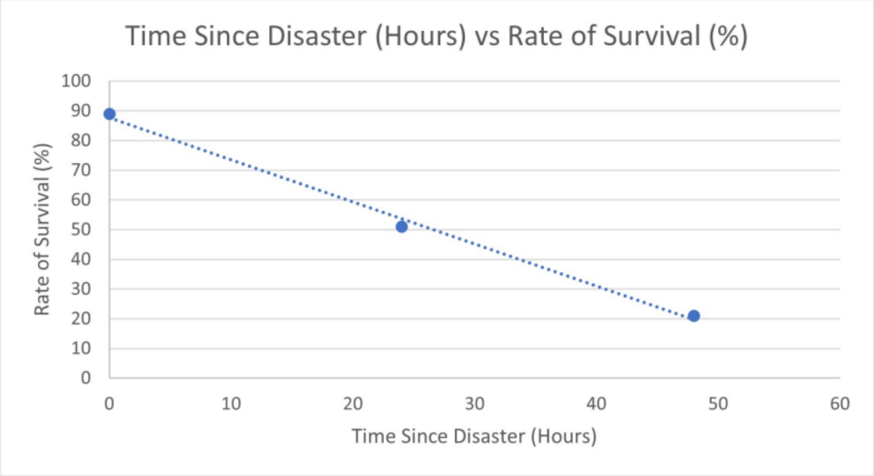

Figure 2. The relationship between time and survival rate [1].

With these possible applications in mind, with advances in the field of robotics and LiDAR (Light Detection and Ranging) technology, the possibilities of combining the two tech neologies and applying them in real scenarios are becoming more achievable by the day. By utilizing a combination of possibility-based SLAM (Simultaneous Localization and Mapping) algorithms and LiDAR point-cloud modelling technologies, it is possible to create relatively small autonomous vehicles that possess the ability to scan, record, and explore interior spaces before the extraction team becomes involved. The LiDAR camera possesses the ability to accurately chart the surroundings using point-cloud modelling technologies, while the SLAM algorithms utilize the above data and orient the vehicle accordingly by using a combination of mapping through observations, starting with a set “beacon” for marking the deploying location for calibration, and localization through the recorded mapping data and odometrical modelling. Artificial intelligence algorithms may be applied to navigate through the terrain and even return to the entrance with the observation data collected in the exploration process for use in another mission.

LiDAR sensors are similar to echolocation techniques used by bats and dolphins in the natural world [2], but with light waves instead of sound. By actively generating pulses of laser beams that reflect off of surfaces and measuring the time it takes for the laser beam to return to the sensor, the LiDAR sensor is able to measure the distance of objects in the path of the laser beam. This is achieved by applying the following equation.

$$R = \frac{ct}{2} \tag{1}$$

In the above equation, R is the distance measured, c is the speed of light, and t is the time between the transmission and reception. Essentially, this allows the sensor to measure the distance of an object by measuring the time it takes for its laser beam to reach and return, similar to a sonar system naturally found on bats and applied on underwater survey vessels. Naturally, this system is heavily affected by noise data, generated by the noise in the location of the sensor, the accuracy and precision of the laser beams, and the density of air which affects the local speed of light [3]. This could be accounted for in the product by applying an anti-noise algorithm that attempts to reconstruct the data by treating the noise data with a distribution-based correction algorithm, likely that of a Gaussian distribution. LiDAR sensors are suitable for this task since this task requires mapping the shape of a space. By using LiDAR sensors, a system of points in a virtual 3D space known as point cloud that corresponds to the real space can be created. Individual points in the 3D space is created with individual laser pulses, which then can be reconstructed into a 3d model of the space of the rescue environment.

More commonly used conventional spinning LiDAR sensors are more suitable in an autopilot driving application on urban roads, for example Velodyne, Hesai and Ouster. There are a series of problems when they are applied to rescue tasks, since the mechanical spinning sensors are relatively heavy and large, resulting in high requirements for the size and structure the chassis, which conflict with the objectives of this system. Data comparisons can be seen in table 1. Thus, a solid-state LiDAR is chosen instead.

This system yields several benefits compared to other applicable systems of navigation and terrain modelling when applied in this scenario. One common alternative system of localizing is the Global Positioning System (GPS) [4]. However, GPS signals require open access to satellite signals, which may be unavailable for indoor environments where survivors may be expected to be found, causing drops in accuracy or even complete losses of localization data. In applied scenarios, losing localization data may have catastrophic consequences, which may result in the loss of the vehicle. Furthermore, a GPS system does not allow the scanning of surfaces to create a point-cloud model, thus a scanning system must be attached externally, which may increase cost and weight. In comparison, the local system of LiDAR may be more stable under application scenarios and allows scanning and localizing to be packaged into one tool, simplifying the system while improving robustness. Another common alternative system is the use of conventional cameras as scanning devices coupled with depth algorithms or two cameras placed a certain distance apart to emulate the binocular vision of human eyes [5], similar to the technique used on certain smartphones. However, this system is less accurate compared to LiDAR systems as it does not use a reliable physical system to measure the distances. Rather, it relies on post-processing algorithms to approximate the distance, which relies heavily on the robustness of the algorithm to function [3]. When a high amount of noise sources is present, camera systems may not work at their full capacity. Furthermore, cameras are essentially passive measuring tools that can only receive light, limiting their capability of operating in dark environments, where they would need an external light source to function. LiDAR, in comparison, uses the speed of light to measure the distance from an object. By using a universal constant that could be calculated instead of relying on computer analysis, the robustness of the LiDAR could be significantly higher than that of depth algorithms. By generating the laser beams itself instead of relying on external sources, the LiDAR sensor essentially functions as its own flashlight, negating the requirement of an external light source. Thus, by using a LiDAR sensor to map and localize, it is possible to avoid the flaws of other commonly used localization and mapping methods, improving the accuracy, efficiency, and robustness of the product.

Concretely, the contributions of this paper include:

1. To design a low-cost and flexible rescue vehicle. The solid-state LiDAR is installed upon a standard servo on the rescue vehicle to solve the limitation of small and fixed FOV (Field of View) caused by a compromise of using the solid-state LiDAR;
2. To design an extended Kalman filter capable of combining different sources of information to accurately reconstruct localization data;
3. To design an experiment to accurately model the noise/uncertainty of the LiDAR measurement.

2. Related works

2.1. Robotic rescue systems

There are existing works on this subject, which include theories regarding the design of rescue vehicles for emergencies from different perspectives, which this paper aims to incorporate into the design of the final product. There are different categories of papers written around the subject, which will be discussed below. Since this paper incorporates a wide variety of areas of research, the papers reviewed is also comprehensive, factoring in a large number of papers.

Prior works have shown that it is possible to construct a system of rescue and logistics in emergency situations with analysis. An example of this is Hakami et al. [1], who detailed the structure of an efficient rescue effort in “Application of Soft Systems Methodology in Solving Disaster Emergency Logistics Problems”, through case studies and logical analysis. It applies SSM (Soft Systems Methodology) to construct a disaster emergency logistics system in earthquakes. The efforts of Hakami et al. and others of the field to describe rescue procedures provided a robust basis for the applications of this project.

Previous research has established the fact that the structure and layout of wheeled robots greatly affect their cross-country capabilities, perhaps as to be greatly expected. “Mobility evaluation of wheeled all-terrain robots” by Thueer et al. [6] proposes a system of mobility evaluation. This paper discusses the physical considerations that may go into evaluating wheeled robots. A model for the kinematic models of a rover and a model for the control architecture was developed, which was applied to static and kinematic analysis in on different modular hardware configurations. It is helpful in that they can aid with the development of the hardware system of this project, specifically the chassis and the support structure of the vehicle.

2.2. LiDAR odometry and mapping

There are several previous methods to adjust the odometry and mapping systems. One important method is by using iterative closest point algorithms, often shortened to ICP. This method calculates the distance between points to stitch different point-clouds together. One improved version of this tool is known as the G-ICP, G meaning “general”, which utilizes both point-to-point and point-to-plane. However, these two methods both rely on data association. Due to a reliance on data association, data matching may become unreliable when the data given is not clear enough. Take the L515 for example, it offers 70 degrees for horizontal FOV. Although enough for most scenarios, low FOVs may cause an issue when facing a wall, since the algorithms will find a lack of features for the point-clouds to be accurately matched, causing significant errors.

In addition, these methods face the problem of point cloud distortion caused by the robot’s motion during the period of the scanning. To address this problem, LOAM (LiDAR Odometry and Mapping) methods were proposed, which fix inconsistencies and aliasing produced due to motion.

Whilst much study had been committed towards the aforementioned individual sectors of research, there is little research done that compiles them together to create a usable product that could be applied in a real scenario. There are papers devoted to compiling some of the aforementioned areas of research, but many of them were only applied in an abstract and generic scenario while requiring further elaboration on creating a usable product. This paper aims to change the aforementioned situation by compiling prior research and novel research to create a theoretical structure of LiDAR rescue robot plus a usable product that could be used in demonstration real-time.

3. Methodology

3.1. Hardware design

The onboard computer used in this project is Raspberry Pi 4B with 4GB ram. It is chosen for its compact size, which means that it could be fitted directly onto the chassis without requiring a larger chassis that would decrease manoeuvrability, and its ARM processor, capable of installing Ubuntu 20.04 for development and running required software, such as ROS (Robot Operating System) and various algorithms used to treat collected data. It serves as a central hub for all other components to be ultimately attached to. The Raspberry Pi is mounted near the back of the chassis on a metal floorboard, connected to a handheld powerband, while a servo mounted near the centre of the chassis serves as a base of the LiDAR sensor, providing a greater FoV with rotation.

The chassis is a simple aluminium alloy tracked chassis with 3 load wheels at each side. The drive wheel installed on top of the side panels provides sufficient power for smooth locomotion with 2 brushed motors connected to each wheel, each possessing a decelerating gearbox for increased torque. As this product does not require fast locomotion and in fact need to avoid sudden movements to allow the use of more efficient constant velocity locomotion models in performing state estimation with a greater accuracy and precision, using this setup minimizes the cost of the product without sacrificing performance.

Intel Realsense L515 was chosen to be applied to this project as the LiDAR sensor. Compared to mechanical counterparts such as Velodyne VLP-16, L515's main advantages are dimensions and weight, which is essential for small autonomous vehicles that need to be mass-produced. Weighing 95g and being comparably smaller [7], it is easier to transport on the chassis compared to VLP-16. Other notable advantages are its significantly higher resolution and accuracy, which allows for mapping small details that may otherwise be difficult to map, improving the detail of the final point-cloud model [7]. Its main disadvantage compared to VLP-16 is the field of view, or FoV for short. Compared to a 360° view offered by Velodyne, it only offers a 70° FoV horizontally [7]. However, this would not be a significant hindrance, as it is possible to rotate the sensor by attaching it to a servo capable of turning 290° or more, allowing 360° mapping without using a mechanical sensor. The data are offered below in Table 1.

Figure 3, Figure 4 and Figure 5 show the pipeline of this project.

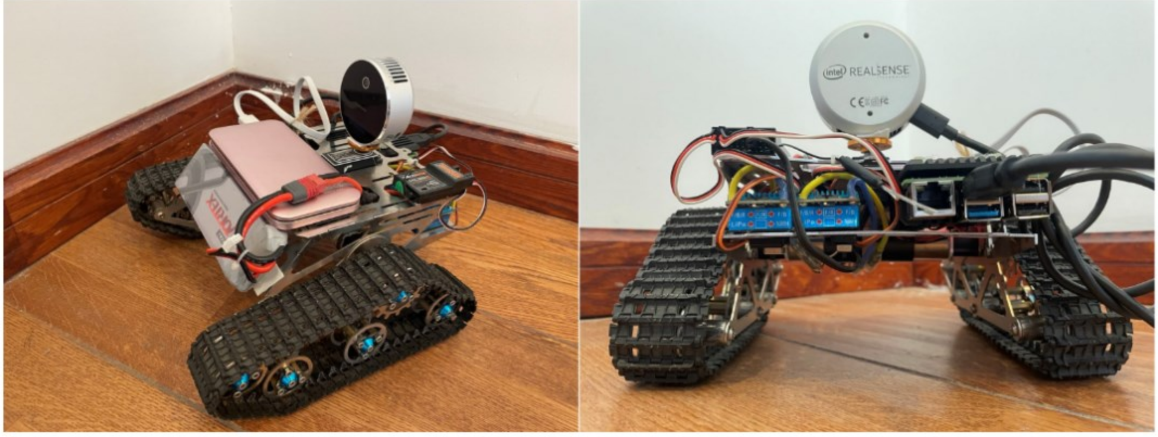

Figure 3. Hardware of the product.

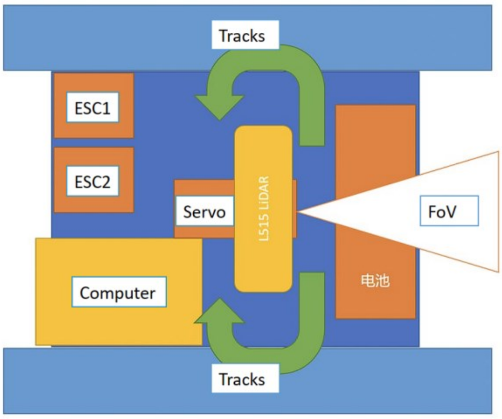

Figure 4. Hardware design diagram.

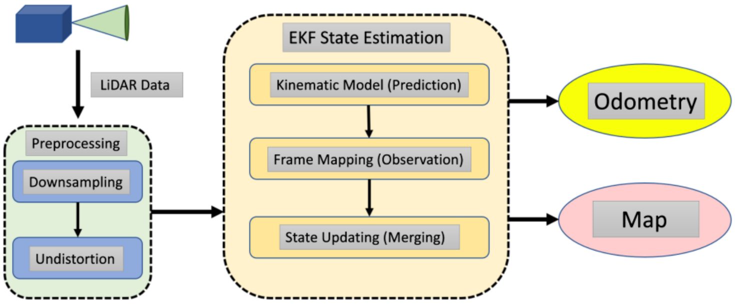

Figure 5. System overview.

3.2. *Framework overview*

At the software level, an EKF (Extended Kalman filter) was designed for achieving more accurate robot localization.

Table 1. Comparison between Velodyne HDL-32E and Intel Realsense L515 [8][9].

Sensor	Type	Framerate	FoV	X-Res	Y-Res	Range	Accuracy	Dimensions	Mass
Velodyne HDL-32E	Mechanical	5-20Hz	360°×41.33°	0.08°	0.33°	100m	2cm	144.2×85.3mm	1000g
Realsense L515	Solid	30Hz	70°×55°	0.07°	0.07°	9m	1.4cm	61×26mm	95g

3.3. *Pre-processing*

3.3.1. *Downsampling*

By employing a voxel filtering down-sampling algorithm, the time complexity of point cloud processing can be reduced, while simultaneously enhancing the overall compactness of information at the expense of lower details. This algorithm initially computes a cubic bounding box that tightly encapsulates the point cloud, which is then partitioned into various smaller cubic voxels based on a predetermined resolution. Subsequently, the centroid of each point within the voxel is calculated, and its coordinates are employed to approximate the position of points within the voxel. The following equation represents the indexing of the point cloud:

$$index_i = z\_index_i * \left[x\_size * y\_size\right] + y\_index_i * x\_size + x\_index_i$$

$$x\_index_i = \left(x_i - min\_x\right) / voxel\_resolution$$

$$x\_size = \left(max_x - min_x\right) / voxel\_resolution \tag{2}$$

The following is the calculation of the coordinates of the centroid. center_xIs the coordinate of the centroid.

$$center\_x = \sum_{i=0}^{n} \frac{x_0 + x_1 + \ldots + x_n}{n} \left(n = 0, 1, \ldots, n\right) \tag{3}$$

Figure 6 and Figure 7 are samples of downsampling. The algorithm retained many details while eliminating large amounts of points.

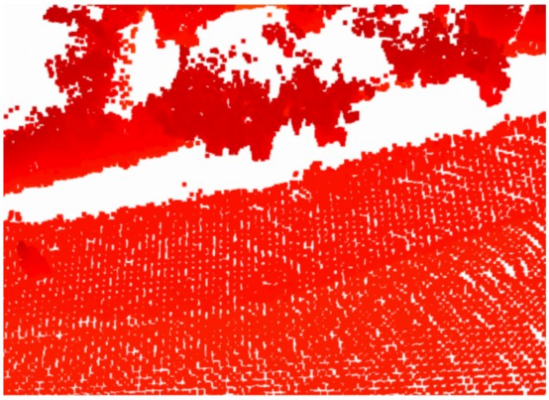

Figure 6. Pre-downsample point cloud.

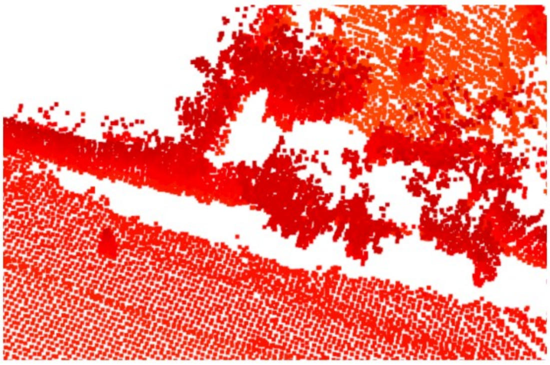

Figure 7. Post-downsample point cloud.

3.4. State estimation

3.4.1. Constant velocity model
The state *x* is defined below:

$$x_t = \left[{}^{G}R_t^T \quad {}^{G}p_t^T \quad {}^{G}v_t^T \quad {}^{G}\omega_t^T \right]^T \tag{4}$$

Through the constant velocity assumption, we can get a pose prior to $\hat{x}_k$ with covariance $\hat{P}_k$ based on last state x_{k-1} with covariance P_{k-1}. Let the time interval between x_k and x_{k-1} be *dt*, then the predicted rotation and position can be calculated through:

$$R_k = R_{k-1} dR$$

$$p_k = p_{k-1} + v_{k-1} * dt \tag{5}$$

where *dR* is a rotation matrix defined by the Rodrigues formula [10]:

$$dR = e^{\hat{r}\theta} = I + \hat{r}\sin\theta + \hat{r}^2\left(1-\cos\theta\right) \tag{6}$$

where $\theta = \left\|\omega_{k-1}\right\| dt$, $r = \frac{\omega_{k-1}}{\left\|\omega_{k-1}\right\|}$ and $\hat{r}$ is the skew symmetric matrix of *r* [11].

Then, the motion model is defined as:

$$\hat{x}_t = f\left(x_{t-1}, dt\right) = \left[dR^T R_{t-1}^T, p_{t-1}^T + v_{t-1}^T * dt, v_{t-1}^T, \omega_{t-1}^T\right]^T \tag{7}$$

The Jacobian matrix G_t is obtained by differentiating the stable variables and evaluating it at the current estimation of the state x_{t-1} :

$$F_t = \begin{bmatrix} e^{\hat{r}\theta} & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & Idt & I & 0 \\ Idt & 0 & 0 & I \end{bmatrix} \tag{8}$$

Denote the process noise in the constant velocity model to be Q_t , then the covariance of the estimated state $\hat{\mathrm{X}}_{\mathrm{t}}$ is given by $\hat{\mathrm{P}}_{\mathrm{t}}$:

$$\hat{P}_t = F_t P_{t-1} F_t^T + Q_t \tag{9}$$

3.4.2. Observation model for the LiDAR point cloud

With the prediction state $\hat{x}_t$ provided by the constant velocity assumption, the LiDAR scan data and the already constructed point cloud map, we can build an observation model to correct the pose. Given a LiDAR point ${}^{B}P_i$ in the LiDAR body frame after motion compensation, we first transform this point to the global frame as below:

$${}^{G}P_i = {}^{G}\overline{R}_t\, {}^{B}P_i + {}^{G}\overline{p}_t \tag{10}$$

Then, we assume the point ${}^{G}P_i$ belongs to the closest plane π_i fitted by its *N* nearby points ${}^{G}Q = \left({}^{G}Q_1, {}^{G}Q_2, \ldots, {}^{G}Q_N\right)$ in the global point cloud map. To complete the fitting of this adjacent plane, we first extract the partial sub-map from the global map according to the FoV of the LiDAR and the pose prior. Then a *k*-D tree (*k* = 3) [12] will be constructed on the point cloud of the sub-map. The reason for using sub-map is to save computation time in building the *k*-D tree, since the time complexity of *k*-D tree is $O\left(m\log\left(n\right)\right)$ where *m* is the dimension and *n* is the number of points in the point cloud map. For plane fitting of the nearby points, we calculate the point covariance matrix Σ:

$$\overline{Q} = \frac{1}{N}\sum_{i=1}^{N} {}^{G}Q_i \, ; \, \Sigma = \frac{1}{N}\sum_{i=1}^{N}\left({}^{G}Q_i - \overline{Q}\right)\left({}^{G}Q_i - \overline{Q}\right)^T \tag{11}$$

Let λ_k denote the *k*-th largest eigenvalue of matrix. Then we determine that these points form a plane when the following conditions are met:

$$\lambda_3 < \sigma_1 \text{ and } \lambda_2 > \sigma_2 \tag{12}$$

where σ_1 and σ_2 are the pre-set hyperparameters. Besides, the eigenvector u correspondent to λ_3 is considered to be the normal vector of the plane. Recall the assumption that the plane π_i is where the point ${}^{G}P_i$ truly belongs to, then the point to plane distance should be zero, which constitutes the residual. Denoting u_i the normal vector of the corresponding plane with a centriod ${}^{G}q_i$, then the residual z_i is computed as:

$$z_i = h\left(\overline{x}_t, {}^{B}P_i\right) = u_i^T\left({}^{G}P_i - {}^{G}q_i\right) = u_i^T\left({}^{G}\overline{R}_t\, {}^{B}P_i + {}^{G}\overline{p}_t - {}^{G}q_i\right) \tag{13}$$

Similarly to 3.4.1, we compute the Jacobian H_t by differentiating the state variables and evaluating it in the current estimation $\hat{x}_t$ bxt of the state:

$$H_t = \left[-u_i^T\, {}^{G}\overline{R}_t \left\lfloor {}^{L}p_i \right\rfloor_{\wedge}, u_i^T, 0_{1\times3}, 0_{1\times3}\right] \tag{14}$$

If all n LiDAR measurement points are considered together, the residuals z_i and the Jacobian matrix H_t are combined as follows:

$$z_t = h\left(\overline{x}_t, {}^{B}P\right) = \begin{bmatrix} h\left(\overline{x}_t, {}^{B}P_1\right) \\ h\left(\overline{x}_t, {}^{B}P_2\right) \\ \cdots \\ h\left(\overline{x}_t, {}^{B}P_n\right) \end{bmatrix}$$

$$H_t = \begin{bmatrix} -u_1^T\, {}^{G}\overline{R}_t \left\lfloor {}^{L}p_1 \right\rfloor_{\wedge}, u_1^T, 0_{1\times3}, 0_{1\times3} \\ -u_2^T\, {}^{G}\overline{R}_t \left\lfloor {}^{L}p_2 \right\rfloor_{\wedge}, u_2^T, 0_{1\times3}, 0_{1\times3} \\ \cdots \\ -u_n^T\, {}^{G}\overline{R}_t \left\lfloor {}^{L}p_n \right\rfloor_{\wedge}, u_n^T, 0_{1\times3}, 0_{1\times3} \end{bmatrix} \tag{15}$$

Considering that each LiDAR point has varying levels of noise, which are represented by covariance w_i, we optimize the pose to minimize the weighted covariance residuals. As a result, we obtain the updated state of the radar observation model, denoted as $\overline{x}_t$.

3.4.3. State update

After obtaining the prior pose estimate $\hat{x}_t$ and $\hat{P}_t$ based on the assumption of constant velocity model, we incorporate n valid measurements from the current frame of the LiDAR. With this information, we construct a Maximum A Posteriori (MAP) estimation.

$$\min_{x_t}\left(\left\| x_t \boxminus \hat{x}_t \right\|_{\hat{P}_t}^2 + \sum_{i=1}^{n} \left\| z_i - H_i \cdot \left(x_t \boxminus \overline{x}_t \right) \right\|_{w_i}^2 \right) \tag{16}$$

where $\boxminus$ denotes the operation defined on the manifold [13]. For rotation matrices R_1 and R_2, as well as three-dimensional vectors v_1 and v_2, the operation is defined as follows:

$$R_1 \boxminus R_2 = \log\left(R_2^T R_1\right)$$

$$\boldsymbol{v}_1 \boxminus \boldsymbol{v}_2 = \boldsymbol{v}_1 - \boldsymbol{v}_2 \tag{17}$$

We use the EKF to solve this Maximum A Posteriori (MAP) estimation problem. Considering all *n* valid LiDAR measurements, we define $H_t = \left[H_1^T, H_2^T, \ldots, H_n^T\right]^T$ and $W_t = diag\left(w_1, w_2, \ldots, w_n\right)$. The calculation of the Kalman gain K_t is given by the following equation:

$$K_t = \hat{P}_t H_t^T \left(H_t \hat{P}_t H_t^T + W_t\right)^{-1} \tag{18}$$

Finally, utilizing the Kalman gain K_t, we obtain the optimal updated state x_t and covariance matrix P_t as follows:

$$x_t = \hat{x}_t \boxplus K_t\left(z_t - h\left(\hat{x}_t, {}^{B}P\right)\right)$$

$$P_t = \left(I - K_t H_t\right)\hat{P}_t \tag{19}$$

3.5. Map update

Based on the optimized pose x_t obtained from the EKF, we transform each downsampled LiDAR point from the LiDAR coordinate frame to the world coordinate frame and store it in the global map.

4. Experimentation

4.1. Experiment design

The chassis used in this project is based on a differential drive kinematic model, which means that rotational motion is achieved by applying differential acceleration to the tracks. The chassis, including the tracks, is approximately 21 cm long, 23 cm wide, and 10 cm high. The L515 LiDAR is mounted on a servo near the centre of the chassis.

4.2. LiDAR noise modelling

The LiDAR sensor is subject to certain errors originating from multiple sources. The accuracy of distance measurements is influenced by the reflectivity of objects as LiDAR relies on laser reflections. According to official data from Intel, for a reflectivity of 15%, the effective range of the L515 LiDAR sensor is only 0.25 - 3.9 meters, whereas for a reflectivity of 95%, the effective range extends to 0.25 - 9 meters [8]. In summary, the accuracy of the L515 decreases significantly when measuring objects with lower reflectivity. In practical usage, this means that the precision of the L515 in measuring dark gray objects will be much lower compared to measuring white objects. Another noise source associated with LiDAR is the ambient light intensity in the environment. In environments with other bright light sources, particularly ample sunlight, the L515 may confuse its emitted laser beams with sunlight, resulting in random measurement noise. Due to the numerous factors involved, this type of noise is challenging to model. However, since this project primarily focuses on indoor usage, the impact of this noise on the experimental results is relatively minor.

Additionally, in accordance with the working principle of LiDAR, when an object is either too close or too far from the sensor, the LiDAR sensor is unable to accurately measure the distance. This is because LiDAR sensors rely on the speed of light to measure distances, and since the speed of light is extremely fast, the time difference between emission and reception becomes too small to measure accurately within very short distances, thus introducing errors. Moreover, due to the lower resolution of the laser emitted by LiDAR sensors, details of objects that are farther away cannot be adequately reconstructed due to the greater distance between the two laser pulses. Furthermore, the laser beam reflected from objects that are farther away is weaker, leading to distance measurement errors. To accurately predict and account for these errors, the author conducted distance measurement experiments using LiDAR. The following images depict the experimental setup.

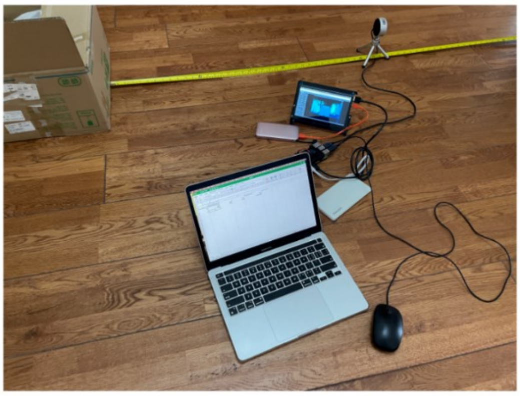

Figure 8. Experimental setup for distance measurement experiment.

We used the Intel Realsense Viewer application running on the Raspberry Pi for data collection.

By calculating upon these data, we could find the difference between the actual distance and the tested distance.

Table 2. Data processing for calculating average measured distances.

Actual Distance (m)	Measured Distance (m)			Average Measured Distance (m)
0.50	0.508	0.509	0.507	0.508
1.00	1.006	1.007	1.006	1.006
1.50	1.504	1.502	1.498	1.501
2.00	2.001	1.998	1.995	1.998
2.50	2.494	2.507	2.502	2.501
3.00	2.987	3.013	3.002	3.001
3.50	3.500	3.484	3.502	3.495
4.00	3.986	3.996	3.963	3.982
4.50	4.517	4.487	4.482	4.495
5.00	5.016	5.001	4.874	4.964

Table 3. Calculation of the error of measurement.

Actual Distance (m)	Avg. Measured Distance (m)	Difference (m)
0.50	0.508	-0.008000
1.00	1.006	-0.006333
1.50	1.501	-0.001333
2.00	1.998	0.002000
2.50	2.501	-0.001000
3.00	3.001	-0.000667
3.50	3.495	0.004667
4.00	3.982	0.018333
4.50	4.495	0.004667
5.00	4.964	0.036333

It is evident that as the actual distance increases, the testing error gradually amplifies. The overall trend is that points within close proximity (2m) are underestimated, while points beyond a certain distance (3m) are overestimated, and the overall accuracy of the sampling decreases as the distance increases. This indicates that the L515 exhibits higher precision in measuring points in close proximity compared to those farther away. Therefore, it is advisable to prioritize the processing of more accurate points in the vicinity during measurements and attempt to reduce the error by extrapolating the actual distance based on the observed trend. Although this relationship is clearly discernible, the maximum error of 0.036m is not significant enough to cause substantial modelling inaccuracies. Hence, the author considers the data processing for the encountered measurements unnecessary.

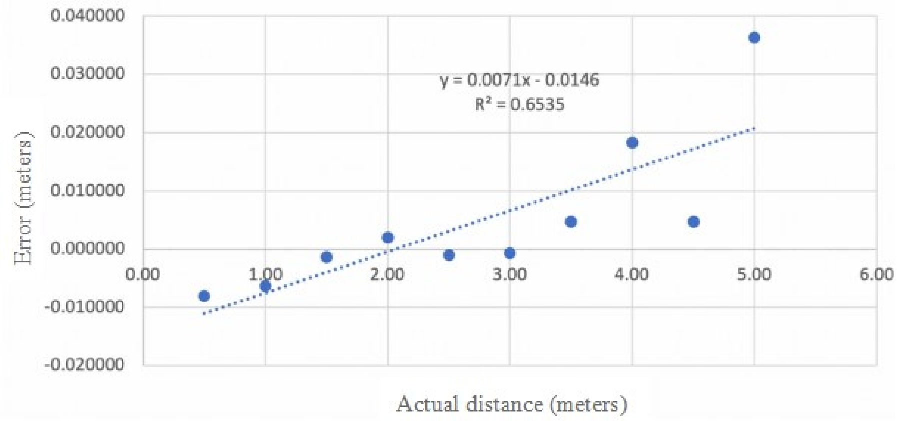

Figure 9. Graph depicting the relationship between actual distance and measurement error.

4.3. Real-life environment test

In this project, the LiDAR sensor needs to create a three-dimensional point cloud model on the rescue vehicle, which may operate in various environments. To test the modelling effectiveness of the LiDAR sensor in different environments, the author chose to place the LiDAR in different settings and attempted modelling in each environment. First, the author conducted a handheld LiDAR modelling experiment in a personal room.

In this experiment, the author processed the raw data using software packages, performing ICP (Iterative Closest Point) and downsampling to align the data collected in each capture and reduce the overall number of points in the point cloud to a manageable range. The algorithms used can be found in the implementation section.

It can be observed that the LiDAR sensor achieves highly accurate modelling results in confined spaces. Nearly all the details in the room are faithfully captured, including furniture, clutter piled up in corners, and the slanting of the ceiling. The only area not modelled is the wall with heating pipes, which was not captured due to limitations in the LiDAR camera’s positioning on the author’s personal computer, as the USB cable restricted movement.

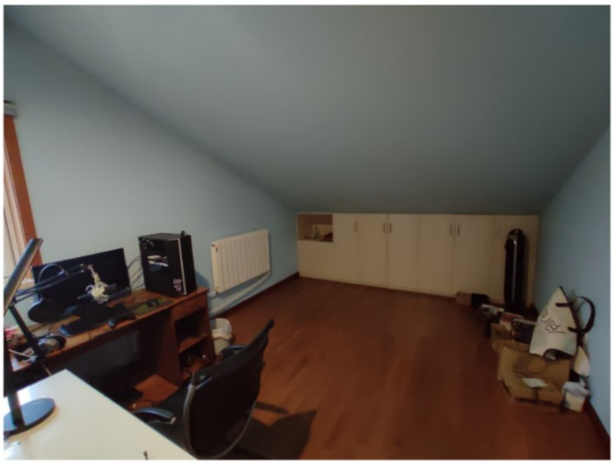

Figure 10. Space being modelled.

Additionally, a series of in-vehicle tests were conducted in various environments, including a residential area with dense vegetation (simulating a forest environment), a living room, and a garage. Figure 12 and Figure 13 show the data captured in the residential area's vegetation during night-time. It is evident that, despite the illumination provided by a flashlight, only a portion of the nearest trees can be seen in Figure 12, while Figure 13 can identify several trees in the farther distance. This is because LiDAR's principle involves actively emitting laser pulses to perceive the environment, whereas conventional cameras can only passively recognize reflected light. In real-life rescue scenarios, the lighting in a room may be extinguished, resulting in complete darkness. In such cases, LED illumination and ordinary cameras are inefficient due to their high-power consumption and lower imaging quality, rendering them unsuitable for 3D modelling. In comparison, the LiDAR on the rescue vehicle can actively scan the environment without requiring additional illumination, enabling the creation of a 3D model and allowing for more precise data acquisition.

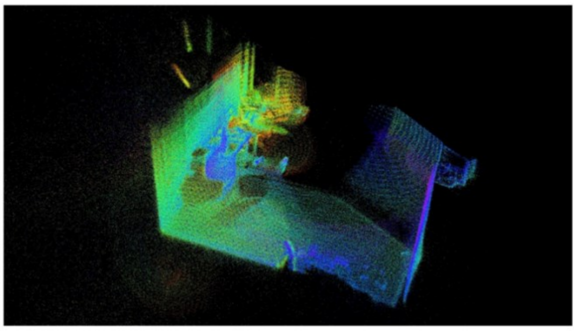

Figure 11. Handheld modelling of the room.

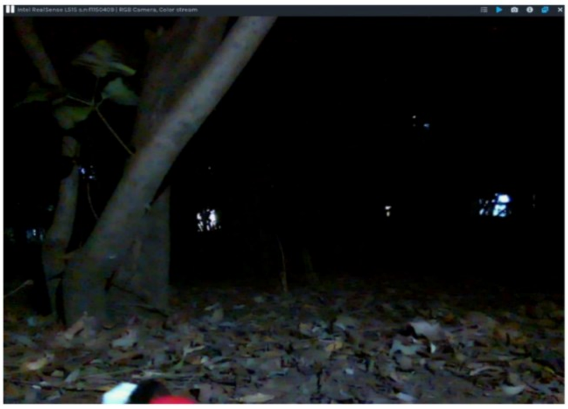
Intel RealSense L515 s.n:f1150409 | RGB Camera, Color stream

Figure 12. The first-person perspective RGB image in the thicket, with a distant bright light coming from the window, and illumination from a flashlight.

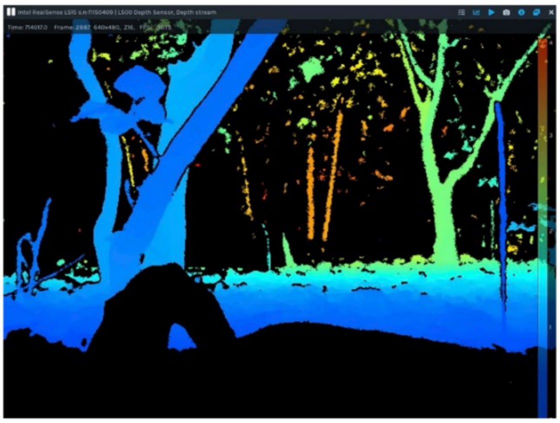
Intel RealSense L515 s.n:f1150409 | L500 Depth Sensor, Depth stream
Time: 714017.0 Frame: 2887 640x480, Z16, FPS: 30.13

Figure 13. The depth image in the thicket, pay attention to the orange and green trees in the distance, which are not visible in the RGB image.

Figure 14 shows the point cloud generated from the vegetation. It can be observed that the point cloud retains most of the details of the actual vegetation, including the boundaries of the foliage and the small path. While the quality is reasonably good, a major issue is the retention of numerous unnecessary details during the mapping process, such as small branches and leaves, which makes navigating through this point cloud quite challenging.

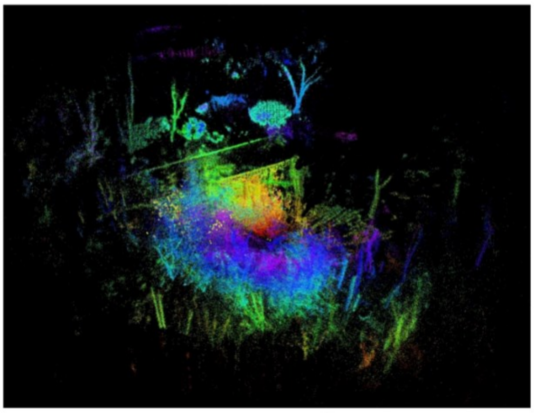

Figure 14. The point cloud of the thicket reveals that the ICP (Iterative Closest Point) algorithm remains effective in handling scenes with a high level of detail.

In addition to the vegetation scene, the author conducted experiments in an underground garage and a living room. In the underground garage scene, the low reflectivity of the dark-coloured car's side surface resulted in the loss of many points in the car's direction. Due to the lack of reference objects, LiDAR was unable to effectively model the car when facing it directly, resulting in the need to rely on points from other angles and directions.

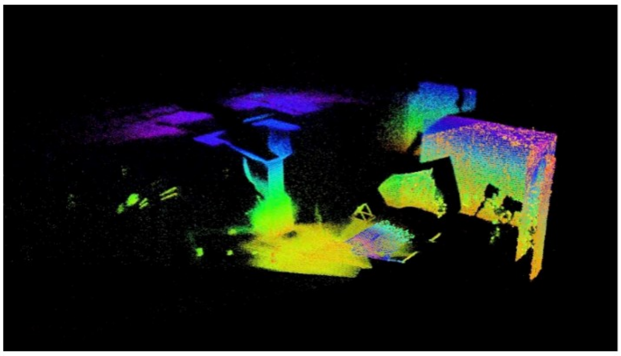

Figure 15. The point cloud of the underground car park, with the car positioned in the direction of the viewing angle.

From this experiment, we can observe the limitations of LiDAR in observing dark-coloured objects.

The best scanning result was obtained in the author's living room. The living room contains various furniture with different colours, shapes, and positions. The LiDAR camera accurately modelled the positions and shapes of the furniture. A comparison reveals that most of the details were accurately reproduced, including smaller features like table legs. This implies that in practical use, the rescue vehicle can precisely identify smaller obstacles such as wooden sticks, furniture, and trash cans, and assist search and rescue teams in detecting the presence of objects and survivors using the point cloud data.

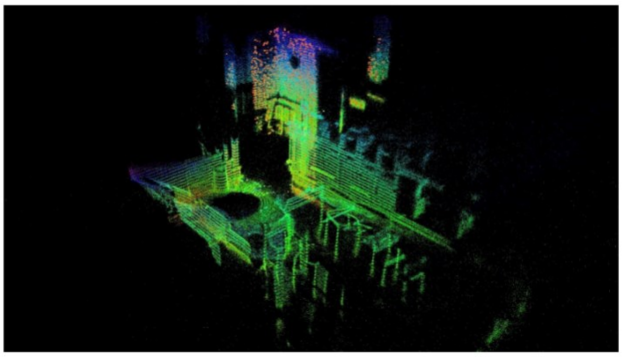

Figure 16. Point cloud modelling of the author's living room.

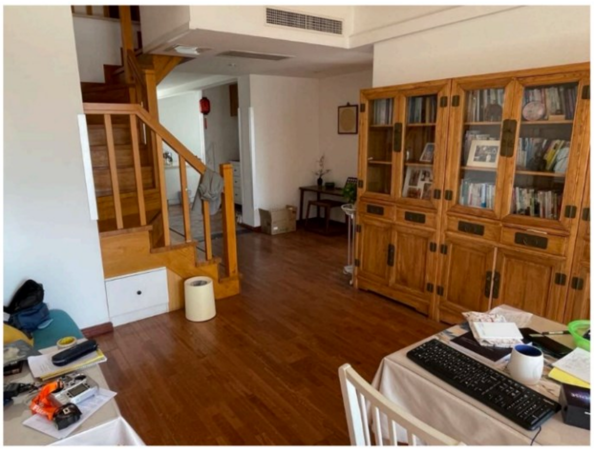

Figure 17. The photo of the author's living room shows that most of the details have been captured.

5. Conclusion and future improvements

In this paper, the author designed a prototype remote-controlled search and rescue vehicle using algorithms such as point cloud downsampling, EKF, and ICP, and conducted various tests. The described search and rescue vehicle can enter narrow spaces for data collection and accurately capture point cloud spatial data, enabling the rescue team to explore and significantly reduce the risks and time involved in search and rescue missions.

One future opportunity for extension is to install this system on a drone. Compared to a tracked chassis, drones have better maneuverability and can fly over obstacles that are difficult to climb. They can also access different floors through stairwells. The main challenges of this design are the influence of the drone's speed and vibrations on scanning accuracy, which can be optimized by designing algorithms specifically tailored for drones.

In future research, this study can be combined with pathfinding algorithms to achieve automatic exploration of indoor spaces. Previous research has established robust pathfinding algorithms. Yermo et al.'s paper "A fast and optimal pathfinder using airborne LiDAR data" [14] provides a detailed description of a pathfinding method based on LiDAR data. Although this paper focuses more on aerial pathfinding, the described method generates pathfinding results relatively quickly based on LiDAR data and can be adjusted according to project requirements. Regarding the use of pathfinding described in this article, Zhao et al.'s paper "Weighted octree-based 3D indoor pathfinding for multiple locomotion types" [15] explores the application of kD-trees in pathfinding algorithms. This paper demonstrates that by using a weighted kD-tree based on height, considering the height and capabilities of the host (in this case, the tracked search and rescue vehicle), it is possible to derive a customized pathfinding method for specific scenarios, allowing the search and rescue vehicle to autonomously explore a space and further enhance automation.

Acknowledgments

I would like to express my gratitude to Dr. Ye for her guidance and influence on my research and for providing valuable advice on the writing methods, techniques, and content of this paper. I also want to thank my parents for their support and numerous suggestions during the writing process. Additionally, I am grateful to my friends for their inspiration and advice on remote control systems.

References

[1] A. Hakami, A. Kumar, S. J. Shim, and Y. A. Nahleh. Application of soft systems methodology in solving disaster emergency logistics problems. International Journal of Industrial and Manufacturing Engineering, 7(12):2470–2477, 2013.

[2] W. Wang and F. Tai. Research progress on bat audition and echo localization in china. Journal of

Shaanxi Normal University: Natural Science Edition, 34(B03):121–127, 2006.

[3] N. Li. Theoretical modeling and experimental study on noise of laser radar receiver. 2016.

[4] W. Wei. Research on Positioning and Navigation System for Substation Inspection Robot based on Differential GPS. PhD thesis, Harbin: Harbin Institute of Technology, 2015.

[5] T. Shen, W. Liu, and J. Wang. Object ranging system based on binocular stereovision. Electronic Measurement Technology, (4):52–54, 2015.

[6] T. Thueer and R. Siegwart. Mobility evaluation of wheeled all-terrain robots. Robotics and Autonomous Systems, 58(5):508–519, 2010.

[7] H. Wang, C. Wang, and L. Xie. Lightweight 3-d localization and mapping for solid-state lidar. IEEE Robotics and Automation Letters, 6(2):1801–1807, 2021.

[8] Intel. Intel® realsense lidar camera l515 datasheet. https : / / www . intelrealsense . com / download / 7691/, 2021.

[9] Velodyne. Hdl-32e high resolution real-time 3d lidar sensor. https://velodynelidar.com/products/hdl-32e/.

[10] J. S. Dai. Euler–rodrigues formula variations, quaternion conjugation and intrinsic connections. Mechanism and Machine Theory, 92:144–152, 2015.

[11] J. Wang, J. Boyer, and M. G. Genton. A skewsymmetric representation of multivariate distributions. Statistica Sinica, pages 1259–1270, 2004.

[12] K. Zhou, Q. Hou, R. Wang, and B. Guo. Real-time kdtree construction on graphics hardware. ACM Transactions on Graphics (TOG), 27(5):1–11, 2008.

[13] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. Information Fusion, 14(1):57–77, 2013.

[14] M. Yermo, F. F. Rivera, J. C. Cabaleiro, D. L. Vilariño, and T. F. Pena. A fast and optimal pathfinder using airborne lidar data. ISPRS Journal of Photogrammetry and Remote Sensing, 183:482–495, 2022.

[15] J. Zhao, Q. Xu, S. Zlatanova, L. Liu, C. Ye, and T. Feng. Weighted octree-based 3d indoor pathfinding for multiple locomotion types. International Journal of Applied Earth Observation and Geoinformation, 112:102900, 2022.