symmetry

Article

Parallelization of Finding the Current Coordinates of the Lidar Based on the Genetic Algorithm and OpenMP Technology

Lesia Mochurad [1,*] and Natalia Kryvinska [2]

[1] Department of Artificial Intelligence, Lviv Polytechnic National University, Kniazia Romana Str., 5, 79905 Lviv, Ukraine
[2] Faculty of Management, Comenius University in Bratislava, Odbojárov 10, 820 05 Bratislava, Slovakia; Natalia.Kryvinska@fm.uniba.sk
* Correspondence: lesia.i.mochurad@lpnu.ua; Tel.: +380-97-868-30-14

Abstract: The problem of determining the position of the lidar with optimal accuracy is relevant in various fields of application. This is an important task of robotics that is widely used as a model when planning the route of vehicles, flight control systems, navigation systems, machine learning, and managing economic efficiency, a study of land degradation processes, planning and control of agricultural production stages, land inventory to evaluations of the consequences of various environmental impacts. The paper provides a detailed analysis of the proposed parallelization algorithm for solving the problem of determining the current position of the lidar. To optimize the computing process in order to accelerate and have the possibility of obtaining a real-time result, the OpenMP parallel computing technology is used. It is also possible to significantly reduce the computational complexity of the successive variant. A number of numerical experiments on the multi-core architecture of modern computers have been carried out. As a result, it was possible to accelerate the computing process about eight times and achieve an efficiency of 0.97. It is shown that a special difference in time of execution of a sequential and parallel algorithm manages to increase the number of measurements of lidar and iterations, which is relevant in simulating various problems of robotics. The obtained results can be substantially improved by selecting a computing system where the number of cores is more than eight. The main areas of application of the developed method are described, its shortcomings and prospects for further research are provided.

Keywords: optimization task; parallel algorithm; robotics; the performance indicator; iterative algorithm; parallelization according to the data

check for updates

Citation: Mochurad, L.; Kryvinska, N. Parallelization of Finding the Current Coordinates of the Lidar Based on the Genetic Algorithm and OpenMP Technology. *Symmetry* **2021**, *13*, 666. https://doi.org/10.3390/sym13040666

Academic Editor: Aviv Gibali

Received: 24 March 2021
Accepted: 11 April 2021
Published: 13 April 2021

1. Introduction

One of the most interesting areas of study of algorithms is to solve optimization and modeling problems using artificial intelligence systems. To date, a large number of scientific works on this topic was written [1–6]. Of these, you can draw a proper understanding of the search algorithms used for a variation of the desired parameters with high accuracy. The purpose of such algorithms consists in predicting the end coordinates, which will find an object. Machine learning algorithms to a certain extent allow you to localize the work and classify the obstacles with which it may be faced. In detecting obstacles, such algorithms allow that this is a new obstacle, that is, states will not be linked.

In this paper, we assume that the environment is unchanged and the object is the only moving. To avoid collisions, it is desirable not only to know the location of the obstacles, but also be able to predict the coordinates of the object for the near future. To solve this task will help a genetic algorithm. But computation speed has been optimized by parallelizing this algorithm, making the proposal suitable for real-time applications. As you know [1], the genetic algorithm is a search procedure based on the mechanisms of natural selection and succession. It uses the evolutionary principle of survival of the most adapted individuals. It differs from traditional optimization methods by several basic elements. In particular, the genetic algorithm:

- Processes the value of the parameters of the task itself, and their encoded form;
- Searches a solution to leave not from a single point, but with their some population;
- Uses only target function, not its derivatives or other additional information;
- Applies probabilistic rather than deterministic selection rules.

In the paper [7] the authors presented a parallel implementation of genetic algorithm using map/reduce programming paradigm. Hadoop implementation of map/reduce library is used for this purpose. The model described here for parallelization of the genetic algorithm has a significant disadvantage—low quality of solution because of species problem.

The article [8] performed a survey of different deep learning techniques applied to various agricultural problems. So, the paper analyzed the specific employed models, the source of the data, the performance of each study, the employed hardware, and the possibility of real-time application to study eventual integration with autonomous robotic platforms.

The problem of finding the current coordinates of the lidar with optimal accuracy can be accomplished through a range of methods that are inherently computationally cumbersome.

This paper aims to develop an approach of finding the lidar position with optimal accuracy based on environment information, optimizing the proposed algorithm using the genetic algorithm, and parallelizing certain parts of it.

The main contribution of this paper can be summarized as follows:

1. We have proposed a simple yet accurate and computationally efficient approach to finding the lidar position based on a genetic algorithm and the OpenMP parallel computing technology. It provides the possibility of significant optimization of the computing process by its parallelization; the ability to solving the task for the case of an extensive data processing;
2. We have developed the algorithmic implementation of the proposed method. It is especially relevant in the development of the multi-core architecture of modern computers.
3. We have demonstrated the reduction in computational complexity using the proposed method; we have received an acceleration that goes to the number of cores of the appropriate computing system (we have achieved the parallel efficiency of about 1 in this case).

The rest of this article is organized as follows: analysis of literary sources are presented in Section 2. The formulation of the problem, descriptions of the classic genetic algorithm, and designed approaches are shown in Section 3. Section 4 comprises the results of numerical experiments of the proposed algorithmic implementation of the developed method. The results of the discussion are presented in Section 5. Conclusions and prospects for further research are presented in the last section.

2. Analysis of Literary Sources

At the moment, the world is increasingly investigating a variety of robotics problems [9–12]. One of the most interesting is to determine the current position of the lidar. In manufacturing and life, such a task is widely used as a model in many spheres, such as planning a route of vehicles, machine training, economic efficiency management.

Several literary sources that were relevant to the parallelization of the lidar's finite coordinate algorithm are analyzed. However, most of them were not useful, because they did not provide the results of practical implementation and contained mainly a general overview of the problem.

Latest studies use such algorithms for investigating the problems of finding coordinates: a genetic algorithm, an simulated annealing algorithm, an ants colony algorithm, and an algorithm of a neural network. Among them, the genetic algorithm has advantages in greater stability and stronger global search capabilities, and therefore applies to the problem of planning trajectories.

Also was found a work where was investigated the parallelization of the genetic algorithm with CUDA [1]. It presents a new method of parallelization of the genetic algorithm for solving a traveling salesman problem. The solution provides information on routes, except for all services required by autonomous vehicles in transport clouds.

In the article [3], a method of genetic search is developed to solve the problem of navigation. Here the search node or the path is represented by a row of integers, each of which represents a cell on the terrain. The results of simulation, which compare the proposed genetic algorithm and a sequential navigation algorithm are presented.

The article [13] presented an efficient implementation of the Extended Kalman filter for Simultaneous localization and mapping algorithm on a multi-processor architecture. The overall accuracy of the algorithm depends on the number of the landmarks in the state vector and the matched observations.

To optimize computing processes in order to accelerate and avoid numerical instability in works [14–18], the use of various technologies of parallel calculations such as OpenMP, Java Threads, Java Forkjoin, AMIDST and CUDA is analyzed. A method to detect and classify landing sites from lidar data in parallel on multi- and manycore systems using OpenMP is presented in the work [19]. Technologies of parallelizing are widely used in ensemble models based on artificial neural networks [19,20]. This practice is especially characteristic of the stacking models, where it is necessary to carry out a prediction based on a given set of data with a large number of members of the stacking neural network ensemble. The parallelization in this case is necessary to minimize time resources to perform procedures as training as well as the use of such ensembles [20,21].

The disadvantages of analyzed works are: insufficient number of numerical experiments conducted, high computational complexity of algorithms, inconsistency of input data with expected results. That is why we have been decided to develop its algorithm, to conduct a detailed analysis of a speed based on a number of numerical experiments and optimize it with parallel computing.

3. Materials and Methods

In a general task, the definition of the current position of the lidar can be described in this way. Let, we have a lidar that is inside the room. It sequentially performs the following steps:

- Scans space around itself with a particular error.
- Moves on some vector $\{dx,\ dy\}$ also with a particular error.

It is necessary to accurately estimate the possible position of the lidar after several iterations based on the data obtained from the first step.

In order to solve the problem, let us suppose without reducing the universality that the room will be depicted in the form of a plane (in our case a polygon), which is represented by a set of points (x, y) (see Figure 1). By connecting these points consistently we will receive the model of our room. The lidar is presented in the form of a point that will be located in the middle of our plane and the size of which we can neglect.

When applying a genetic algorithm for the problem, it is encoded in such a way that its solution can be represented as an array of composition of the chromosome composition [22–24]. Randomly in an array there is a certain number of initial elements of "persons," or an initial population. Individuals are evaluated using the bumping function, as a result of which each person is assigned a certain value of binding, which determines the possibility of survival of a person. After that, with the use of the received impairment values, persons admitted to cross (breeding) are selected. The persons are used "Genetic Operator" (in most cases it is a crossover (crossover) and mutation operator), thus creating the next generation of persons. The next generation persons are also evaluated by the use of genetic operators and selection and mutation are performed. It is so modeled by the evolutionary process that continues several life cycles (generations) until the algorithm stop criterion will be performed.

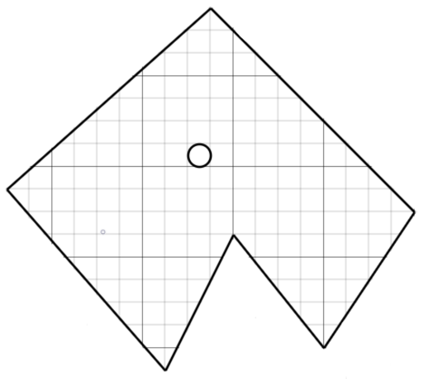
Figure 1. Visual representation of a problem model.

As a result, it is possible to distinguish the following steps of the genetic algorithm:

1. Creating an initial population.
2. Calculation of bumping function for populations (evaluation).
3. Repeat to perform the algorithm stop criterion:
    a. Choosing Individuals from current population (selection);
    b. Crossing or/and mutation;
    c. Calculation of bumping function for all persons;
    d. Formation of a new generation.

3.1. Modeling Sequential Algorithm Support

- *Creating an initial population*

In order to predict where our lidar shifts we need to create an initial population–a set of points that correspond to the possible starting position of the lidar.

This task has two possible cases:

1. The initial point is known.
2. The initial point is not specified.

For the first option, everything is quite unambiguous. Let us create an initial population of a certain size and fill in its starting point. For the second case we need to generate arbitrary points and choose them as candidates for possible initial placement. Over time they will evolve and accumulate around the right placement. In order to generate a certain number of points inside the polygon, you need to perform certain actions.

We find the size of a square that describes our room. To do this, you need to calculate the maximum and minimum coordinate values for our municipality. I portray the points in Figure 2 and we specify that they will be the points of a rectangle that describes our room.

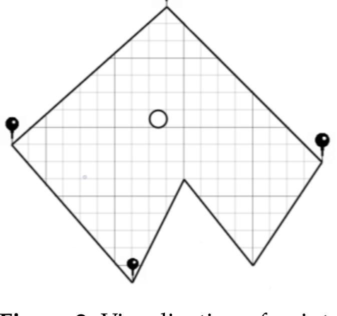
Figure 2. Visualization of points that describe the room.

In Figure 3 it is shown at the extreme points of our plane rectangle. With information about it arbitrarily generating points within its limits to check whether they belong to our polygon.

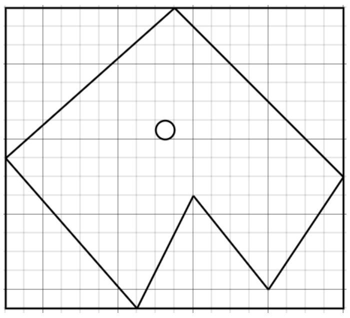

Figure 3. Built a rectangle based on the extreme points of the room.

To verify whether the point within the multipourer is used, we use the method described below. Generate an arbitrary point that does not lie inside our rectangle. Through the point we check out and the one that generated outside the plane is directly conducted. It is believed that if such a direct crosses the odd number of sides of the polygon, then, respectively, the point lies within its limits (inside).

Let us prove this pattern. Let the point that lies inside the polygon (checking point) should cross the line with a generated point that is outside the plane. In this regard, the line must cross the rectangle side and be beyond it.

In the future, if the line crosses the side, it means that it again is inside the plane and it needs to go out to reach the generated point. From this it can be concluded that direct, which combines our two points should cross the odd number of sides (see Figure 4).

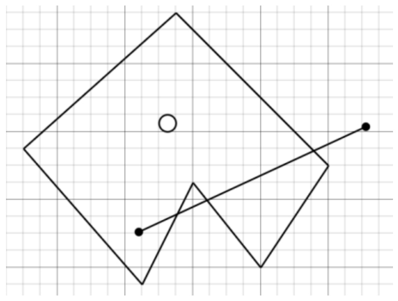

Figure 4. Line connecting the point checking (is in a multi-tape) and generated.

Let us also check the point that lies outside our municipality (see Figure 5). With Figure 5 we can see that in this case, the direct that connects our points crosses the number of parties.

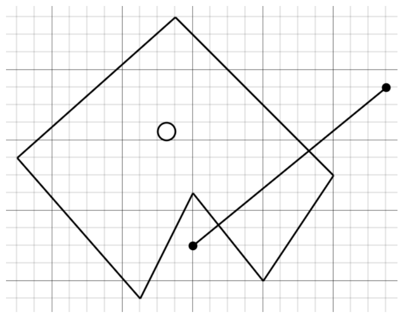

Figure 5. The line connecting the point checked (is outside the polygon) and generated.

With this method we can generate an arbitrary number of points for the initial population.

- *Lidar's Space Scanning Process*

In the scanning process, our lid produces to lasers and finds the distance to the nearest obstacle in a certain direction (see Figure 6). It is allowed that ours $K = 36$, the lidar will release a ray, and do a turn on $360/K = 10$ degrees. Thus, the liberus, using the laser return time, will be able to evaluate its relative distance to obstacles.

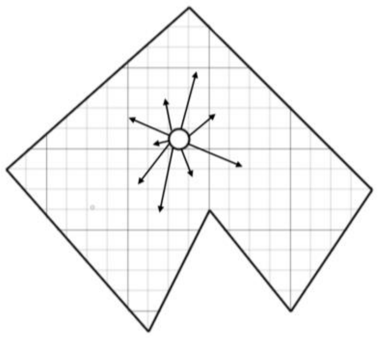

Figure 6. The visual representation of the lidar that produces lasers.

With the help of this assessment, in the future we will conduct a selection and evolution of certain possible states of placement of lidar, because not all of the states are selected in this iteration corresponds to this received during scanning.

Each time, after scanning, the lidar moves to a certain vector. The problem is that there is a certain error, and the lidar is at the point $[(x - \exp,\ x + \exp),\ (y - \exp,\ y + \exp)]$. Let us make it concluded that the possible location of the lidar extends in all directions, and after scanning the environment, on the contrary, is centrifuged around the most suitable states.

- *Data analysis received from lidar after scanning*

During each iteration, we have a set of possible states of placement of lidar. However, it is obvious that not all of them are equally optimal. Some are better and some are worse. In order to evaluate their importance, we use the following algorithm:

1. Save all k lidar measurements.
2. By approaching the states of the placement represented by a certain point, for each of them, we find a possible point of collision with an obstacle. Since we know on which angle of the laser sends a laser, and the distance it has passed we can find a point of collision $(x_1,\ y_1)$(see Figure 7):

$$x_1 = x + L * \cos(\frac{\varphi}{\pi} * 180);\ y_1 = y + L * \cos(\frac{\varphi}{\pi} * 180).$$

3. For each such point we find its deviation—a minimum among the distances to all sides of the polygon. We argue that the closest wall and will be able to go to the lidar laser, so we can estimate the deviation of this state as the amount of deviations of all measurements of the lidar:

$$G(x,y) = \sum_{i=0}^{n} f(x_i,\ y_i),$$

   where $f(x_i,\ y_i)$ is the minimum among the distances to the walls of our polygon.
4. Transfer an estimate of deviation in a probability using a function of normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

   where, mathematical expectation μ and standard deviation σ are given by the lidar parameter.

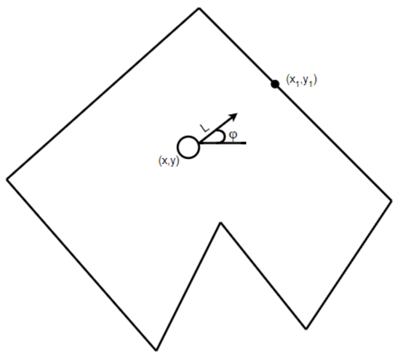

Figure 7. Visual representation of finding point of collision.

With such an analysis and transformation, we have been able to obtain a probability for each state that when these are the indicators of the lidar, this state can be correct.

- *The process of evolution and development of states*

An important part of the evolution process is that the conditions that have a small probability can also be correct, so to adhere to the greedy method and to leave only the states with great probability is not a right approach, so we need to mix the indicators of certain states (one of the main rules of genetic algorithms).

To do this, we use the method of normalization of intervals, which consists in selecting the next set of possible states, depending on the probability of current.

1. We find the probability of each state.
2. Normalize their likelihood so that the amount is equal to 1.
3. Convert a set of probabilities into an interval amount. For example:

$$\begin{gathered} 0.15,\ \ 0.4,\ \ 0.2,\ \ 0.25 \\ \downarrow \\ 0,\ \ 0.15,\ \ 0.55,\ \ 0.75,\ \ 1 \end{gathered}$$

4. Generate N arbitrary numbers with a normal distribution of the gap $[0,\ 1]$. Then, if the generated number lies on the interval corresponding to the state with the number, we add this state into a set of newly formed ones.

Thus, in a rather large number of generated numbers, we argue that in a new set of states, those that have a high probability will be more widespread, about both states with low probability have a chance for survival.

3.2. Parallel Algorithm

Consider the computational complexity of the sequential algorithm. Let:

n—the number of iterations of our lidar,
k—the number of measurements of lidar on each iteration,
m—number of vertices in our room,
q—number of generated initial states.

Then this complexity can be estimated as $O(n * (k + m) * q)$, and it is obvious that it grows quite quickly.

Therefore, one of the methods of optimizing this algorithm is the parallelization of certain parts. Since genetic algorithms are mostly iterative and develop on the basis of past iteration, we cannot make iterations in parallel, however, you can parallel the iteration itself.

In this algorithm there are two parts that can be optimized:

1. Analysis of data from lidar;
2. Evolutionary process.

It is also obvious that these parts of the algorithm are performed independently for each measurement of the lidar and the state of possible placement, which allows its apparatus effectively.

With this optimization, the complexity of the algorithm is reduced to $O(n * m)$, that provides a great advantage in time. To organize the parallelization of programs in order to increase the efficiency of their computer implementation on multi-core processors with the general memory, you can use the OpenMP software (Open Multiprocessing).

In order to paralyze some logic, it needs to be carried out in a separate function (callback) and cause it for a new stream. OpenMP creates the so-called task, which is a wrapper to create a new stream [15].

Based on the #pragma omp parallel and #pragma omp tast functions, the analysis of lidar measurements was paralleled and a new population was created. The #pragma omp parallel directive tells the compiler that the following code uses a lot of streaming, and ensures that threads work safely if they share information. #pragma omp tast–tells the compiler that the next block of code should be wrapped in omp tast and run in a separate thread.

Sometimes significant benefits cannot be achieved through false sharing. Simultaneous updates of individual items in the same cache line coming from different cores of processor invalidate entire lines of the cache, although these updates are logically independent of each other. Each update of an individual cache line item marks the line as invalid. Other processors that access another item on the same line see the string marked as invalid. They have to get a newer copy of the string from memory or elsewhere, even if the item they accessed has not been changed. This is because cache coherence is maintained based on the cache line, not for individual elements.

As a result, there will be an increase in the relationship between traffic and overhead costs. Also, while the cache line is being updated, access to items in the line is denied. This situation is called false exchange. If this happens often, the performance and scalability of OpenMP will be significantly affected. Therefore it is necessary to use various ways of distribution of iterations: schedule(type, chunk_size) [15].

4. Research Results

Then we give the results of the proposed parallel algorithm. To visualize the results, the dependencies and diagrams of the display of achieved efficiency and acceleration will be depicted. Tables 1 and 2 show the results of performance of iterations with parallelized data at different amounts of measurement of the lidar at 4-core and 8-core processors. Figure 8 shows a graph of the dependence of the time of execution of a parallel algorithm from various measurements of the lidar at the 8-core processor in variation in the number of threads. These measurements mean how many rays the lidar emits. It should be noted that the more of these rays, the more precisely determines the current position of the lidar. That is, at 100 dimensions, the algorithm works faster, but issues a fairly large inclination. It is also necessary to take into account that if the number of such measurements will be more than 500, then the error will be minimal. If you take into account 1000 measurements, then the error will be almost the same as at 500, but the time on the calculation will be exposed more. It should also be taken into account that the number of such measurements of the lidar depends on the area of the room, that is, it is larger, the more necessary rays to save an error. Then the usual sequential algorithm can be long or calculations will be too large to fit in RAM. Denote t_s—time of sequential algorithm, a t_p—the time of parallel algorithm, n—the number of iterations, k—the number of measurements of lidar on each iteration, S—parallel acceleration, E—efficiency.

Table 1. Results of performance of iterations with parallel data in different measurements of lidar at 4-core processor.

k	**4-Core Processor**									
	t_s, **ms**	**2 Threads**			**4 Threads**			**8 Threads**		
		t_p, **ms**	S	E	t_p, **ms**	S	E	t_p, **ms**	S	E
100	375	337	1.3274	0.2781	290	1.3923	0.3232	370	1.4082	0.25325
200	703	562	1.4761	0.3252	504	1.5312	0.37022	478	1.7361	0.418
500	1325	974	1.694	0.3314	793	1.7837	0.38308	713	2.0863	0.43312
1000	2172	1873	1.8431	0.3908	1418	1.9658	0.4238	1256	2.3595	0.5259

Table 2. Results of performance of iterations with parallel data in different measurements of lidar at 8-core processor.

k	**8-Core Processor**									
	t_s, **ms**	**2 Threads**			**4 Threads**			**8 Threads**		
		t_p, **ms**	S	E	t_p, **ms**	S	E	t_p, **ms**	S	E
100	367	133	1.6738	0.34492	175	1.9857	0.262	167	1.223	0.1529
200	424	184	1.9741	0.345	172	2.3463	0.3641	156	3.426	0.42825
500	621	237	2.7459	0.3389	204	2.9746	0.3655	193	3.774	0.4717
1000	1230	445	3.0735	0.34112	436	3.8735	0.37512	405	4.103	0.53662

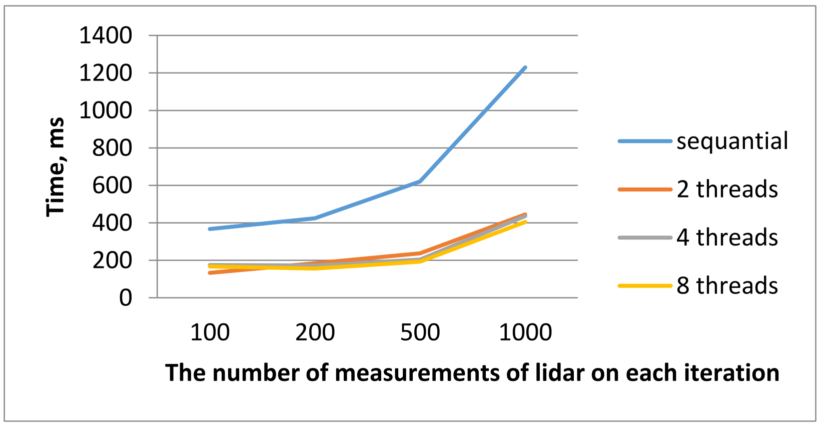

Figure 8. Dependence of the time of execution of the sequential and the parallel algorithms from various measurements of the lidar at the 8-core processor in variation in the number of threads.

Next in Table 3 and in Figures 9–11 shows the overall results of the parallelization of the algorithm proposed.

Table 3. Results of the implementation of the parallel algorithm for determining the current position of the lidar at different amounts of iterations on the 8-core processor.

n	**8-Core Processor**									
	t_s, **ms**	**4 Threads**			**8 Threads**			**16 Threads**		
		t_p, **ms**	S	E	t_p, **ms**	S	E	t_p, **ms**	S	E
367	109.807	26.151	4.0445	0.506	20.738	5.297	0.6621	14.107	7.7883	0.97354
982	214.007	60.913	3.513	0.4391	40.888	5.234	0.65424	27.468	7.791	0.973875
1854	335.25	84.8734	3.95	0.49375	62.986	5.323	0.6653	42.9917	7.798	0.97475

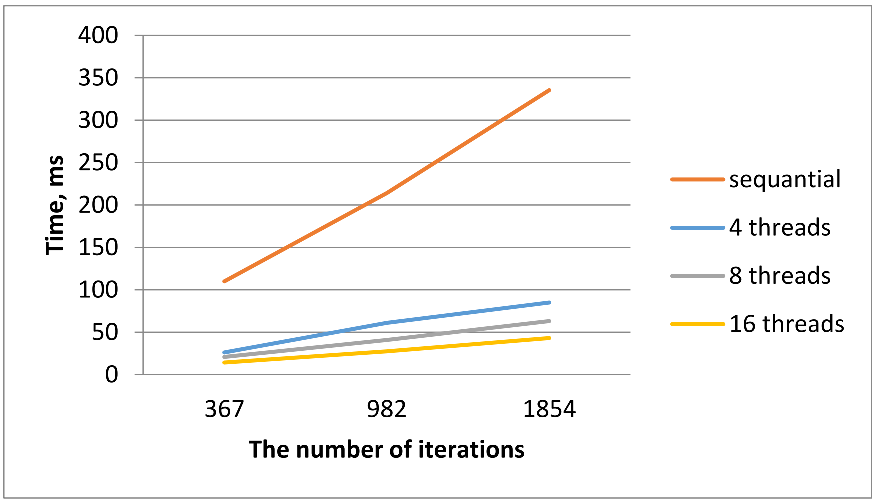

Figure 9. Dependence of the time of execution of the sequential and the parallel algorithms for determining the current position of the lidar at different amounts of iterations on the 8-core processor.

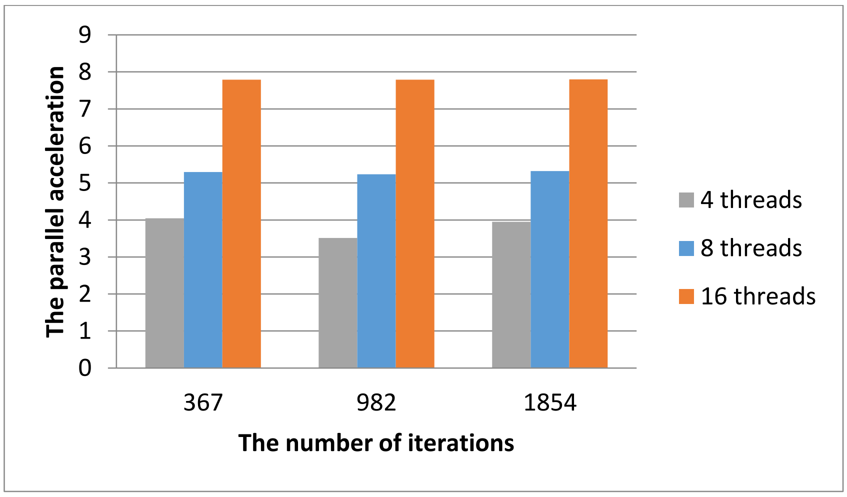

Figure 10. Diagram of displaying the acceleration of the parallel algorithm to determine the current position of the lidar at different amounts of threads for the 8-core processor.

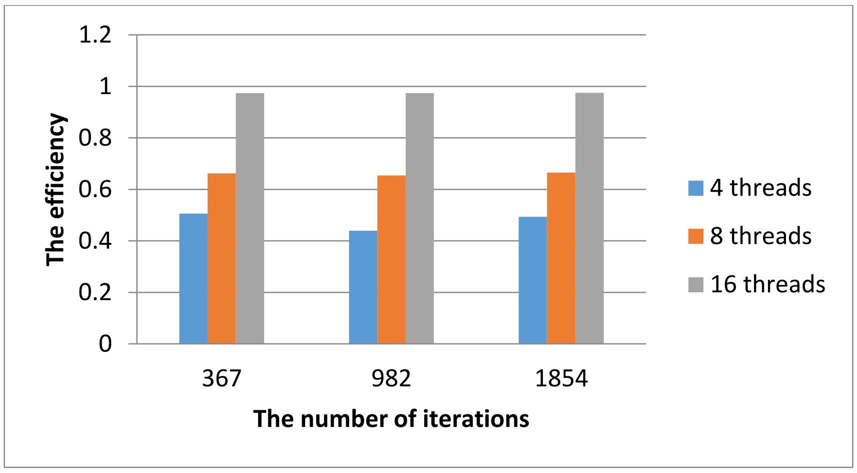

Figure 11. Diagram mapping the effectiveness of the parallel algorithm for determining the current position of the lidar at different amounts of iterations at different amounts of threads for the 8-core processor.

5. Discussion

For the correct analysis of the results obtained in the work it is also necessary to take into account the computing capacity of computers on which numerical experiments were carried out. Such information will help us better understand how many threads should be processed to achieve maximum performance.

To implement a parallel algorithm for determining the current position of the lidar, an OpenMP library was used. Investigating and analyzing the results of parallel performance of iterations at different amounts of measurement of the lidar, you can make several conclusions. After reviewing the dependencies shown for parallelization of iterations at different amounts of measurement of the lower lidar, it is noteworthy that the time of the parallel and serial algorithm on the first 100 measurements is almost different, but after the dimension 200 we begin to observe a significant difference.

Regarding the general results of our algorithm, it is noticeable that the results of a sequential algorithm and results obtained during comprehensive work on the parallelization of the algorithm for determining the current position of the lidar are significantly different. Most of our results were obtained by parallelizing according to the data from our decades. As a result, it was possible to achieve efficiency ~0.97 (with 16 flows on the 8-core processor for 1854 iterations). It is already evident from the schedules of the dependence of the time of parallel execution that for the 8-core processor, the time of execution of a sequential algorithm and paralleled begins to be rapidly different from the 367th iteration.

The obtained results indicate the importance of using parallel computations based on OpenMP technology when solving certain problems for processing large data in order to obtain results in real time. It should also be taken into account the rapid development of multi-core architecture of computing systems, which will obviously afford to significantly improve the results.

6. Conclusions

This article proposes a parallel algorithm for determining the current position of the lidar by using the OpenMP technology and such a property as a lot of gaining based on the genetic algorithm. A detailed analysis of the problem solved is given. The computational complexity, acceleration and efficiency indicators are estimated. The parallelization is carried out. This property is investigated as multithreading. As a result, it was possible to accelerate the computing process about eight times and to take efficiency—0.97. With this optimization, the complexity of the algorithm is reduced to $O(n * m)$, where n–the number of iterations of our lidar, m–number of vertices in our room. The problem of erroneous exchange is investigated, which often has a negative effect on acceleration during parallelization using OpenMP. It should be noted that this study can be substantially optimized in the future by prospects for the development of multi-core architecture. That is, the software product developed in the work can afford better results with a larger number of cores of the corresponding computer. The prospects for further development of this study are the maximum optimization of the computational process based on CUDA technology by using graphic processors and consider the more complex spatial region that is the more difficult to assess the spatial parameters [25].

Author Contributions: Conceptualization, L.M.; methodology, L.M.; software, L.M.; validation, L.M. and N.K.; formal analysis, L.M.; investigation, L.M.; resources, N.K.; data curation, N.K.; writing—original draft preparation, L.M.; writing—review and editing L.M.; visualization, N.K.; supervision, L.M. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abbasi, M.; Rafiee, M.; Khosravi, M.R.; Jolfaei, A.; Menon, V.G.; Koushyar, J.M. An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems. *J. Cloud Comp.* **2020**, *9*, 6. [CrossRef]
2. Goldberg, D.E. (Ed.) *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*; Springer-Science+Business Media, B.V.: Berlin/Heidelberg, Germany; University of Illinois at Urbana-Champaign: Champaign, IL, USA, 2002; Volume 7.
3. Calaiselvy, C.; Yong, F.T.; Ping, L.W. A Genetic Algorithm for Robot Navigation. In *Parallel and Distributed Computing: Applications and Technologies*; Liew, K.M., Shen, H., See, S., Cai, W., Fan, P., Horiguchi, S., Eds.; PDCAT 2004. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3320. [CrossRef]
4. Christophe, S. *Genetic Algorithms with Deep Learning for Robot Navigation*; 2016; 80p. Available online: https://www.doc.ic.ac.uk (accessed on 13 March 2021).
5. López-González, A.; Meda Campaña, J.; Martínez, E.G.; Contro, P. Multi robot distance based formation using Parallel Genetic Algorithm. *Appl. Soft Comput.* **2019**, *86*, 105929. [CrossRef]
6. Rutkovskaya, D.; Pilignan, M.; Rutkovsky, L. *Neural Networks, Genetic Algorithms and Fuzzy Systems*; Translation Polish, I.D., Rudinsky, M., Eds.; Hotline. Telecom; Goryachaya Liniya-Telecom Publ: Moscow, Russia, 2007; 452p.
7. Keco, D.; Subasi, A. Parallelization of genetic algorithms using Hadoop Map/Reduce. *S. Eur. J. Soft Comput.* **2012**, *1*. [CrossRef]
8. Azevedo, F.; Shinde, P.; Santos, L.; Mendes, J.; Santos, F.N.; Mendonça, H. Parallelization of a vine trunk detection algorithm for a real time robot localization system. In Proceedings of the 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Gondomar, Portugal, 24–26 April 2019; pp. 1–6.
9. Li, Z.; Hodgson, M.E.; Li, W. A general-purpose framework for parallel processing of large-scale LiDAR data. *Int. J. Digit. Earth* **2018**, *11*, 26–47. [CrossRef]
10. Elhoseny, M.; Tharwat, A.; Hassanien, A.E. Bezier curve based path planning in a dynamic field using modified genetic algorithm. *J. Comput. Sci.* **2018**, *25*, 339–350. [CrossRef]
11. Jan, B.; Montrucchio, B.; Ragusa, C.; Khan, F.G.; Khan, O. Fast parallel sorting algorithms on GPUS. *Int. J. Distrib. Parallel Syst. IJDPS* **2012**, *3*, 107–118. [CrossRef]
12. Koguciuk, D. Parallel RANSAC for Point Cloud Registration. *Found. Comput. Decis. Sci.* **2017**, *42*, 15. [CrossRef]
13. Amor, N.B.; Baklouti, M.; Barhoumi, K.; Jallouli, M. Efficient embedded software implementation of a low cost robot localization system. In Proceedings of the 2019 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS), Gammarth, Tunisia, 28 April–1 May 2019; pp. 1–5.
14. Mochurad, L.; Solomiia, A. Optimizing the Computational Modeling of Modern Electronic Optical Systems. In *Lecture Notes in Computational Intelligence and Decision Making*; Lytvynenko, V., Babichev, S., Wójcik, W., Vynokurova, O., Vyshemyrskaya, S., Radetskaya, S., Eds.; ISDMCI 2019; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2020; Volume 1020, pp. 597–608. [CrossRef]
15. Mochurad, L.I.; Boyko, N.I. *Technologies of Distributed Systems and Parallel Computation: Monograph*; Publishing House "Bona": Lviv, Ukraine, 2020; 261p, ISBN 978-617-7815-25-8.
16. Mochurad, L.; Shchur, G. Parallelization of Cryptographic Algorithm Based on Different Parallel Computing Technologies. In Proceedings of the Symposium on Information Technologies & Applied Sciences (IT&AS 2021), Bratislava, Slovak Republic, 5 March 2021; Volume 2824, pp. 20–29, ISSN 1613-0073.
17. Agarwal, R.C.; Gustavson, F.; Zubair, M. A high-performance matrix multiplication algorithm on a distributed memory parallel computer using overlapped communication. *IBM J. Res. Dev.* **1994**, *38*, 673–682. [CrossRef]
18. Andrés, R.; Masegosa, A.M.; Martínez, D.R.-L.; Rafael Cabañas, A.; Salmerón, H.; Langseth, T.D.; Nielsen, A.L.M. AMIDST: A Java toolbox for scalable probabilistic machine learning. *Knowl. Based Syst.* **2019**, *163*, 595–597.
19. Lorenzo, O.G.; Martínez, J.; Vilariño, D.L.; Pena, T.F.; Cabaleiro, J.C.; Rivera, F.F. Landing sites detection using LiDAR data on manycore systems. *J. Supercomput.* **2017**, *73*, 557–575. [CrossRef]
20. Izonin, I.; Tkachenko, R.; Vitynskyi, P.; Zub, K.; Tkachenko, P.; Dronyuk, I. Stacking-based GRNN-SGTM Ensemble Model for Prediction Tasks. In Proceedings of the 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 8–9 November 2020; pp. 326–330. [CrossRef]
21. Izonin, I.; Tkachenko, R.; Dronyuk, I.; Tkachenko, P.; Gregus, M.; Rashkevych, M. Predictive modeling based on small data in clinical medicine: RBF-based additive input-doubling method. *Math. Biosci. Eng.* **2021**, *18*, 2599–2613. [CrossRef]
22. Lonely, V.; Dzelandzyk, U. Using genetic algorithms for approximation of function real numbers. In *Computer Science and Information Technologies: Bulletin of the National University "Lviv Polytechnic"*; No 694; Published by Lviv Polytechnic National University: Lviv Oblast, Ukraine, 2011; pp. 313–318.
23. Kanungo, P.; Nanda, P.K.; Ghosh, A. Detection of earth surface cracks using parallel genetic algorithm based thresholding. In Proceedings of the Accepted for the International conference on Advances in Computing, Control and Telecommunication Technology, ACT-2009, Kerala, India, 28–29 December 2009; pp. 4–11.

24. Kanungo, P.; Nanda, P.K.; Ghosh, A.; Samal, U.C. Classification of objects and background using Parallel Genetic Algorithm based Clustering. In Proceedings of the IEEEInternational Conference on Signal and Image Processing, BCET, Hubli, India, 7–9 December 2006; pp. 42–53.
25. Fivos, P. *Spatial Complexity: Theory, Mathematical Methods and Applications*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2020; 319p, ISBN 10 3030596702.