The Scan Matchers Research and Comparison: Monte-Carlo, Olson and Hough

Kirill Krinkin, Anton Filatov, Artyom Filatov
Saint-Petersburg Electrotechnical University "LETI"
St. Petersburg, Russia
kirill.krinkin@fruct.org, {ant.filatov, art32fil}@gmail.com

Artur Huletski, Dmitriy Kartashov
The Academic University
St. Petersburg, Russia
{hatless.fox, dmakart}@gmail.com

***Abstract*—SLAM (simultaneous localization and mapping) problem appears in robot algorithms when it is necessary to orient in an unknown environment without GPS or some other world communication system. Data for considered SLAM method is received from laser rangefinders and odometry sensors. Scan matching is a localization step of the SLAM problem and the main idea is to find an offset between two laser scans – sets of points that presents a perception of environment from one observed point. Three different scan matchers based on various approaches are compared to determine the most accurate one.**

I. Introduction

The objective of the SLAM (simultaneous localization and mapping) is to locate in unknown world and to create a map during this location process. This problem could be decomposed into the following subproblems [1],[2]:

- get information about an environment with robot scanners (data from laser rangefinders and odometry sensors are considered);
- determine robot position using this scan and map built on the previous step;
- update existing map.

These steps presented on the Fig. 1.

The scan matching problem is presented on Fig. 1 and called "matching map and observation". It could be defined in the following way: it is required to find a rotation angle $\Delta\varphi$ and an offset $(\Delta x, \Delta y)$ for two given point sets $\{p_i\}$ and $\{q_i\}$ to make them identical enough:

$$\begin{pmatrix} q_{x,i} \\ q_{y,i} \end{pmatrix} = \begin{pmatrix} \cos(\Delta\varphi) & -\sin(\Delta\varphi) \\ \sin(\Delta\varphi) & \cos(\Delta\varphi) \end{pmatrix} \begin{pmatrix} p_{x,i} \\ p_{y,i} \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \quad (1)$$

The main problem is to find the robot pose delta in the world using data from its laser scanner observed on the previous and current steps.

There are a lot of different ways to solve this problem in specific cases [3],[4],[5]. To the best of our knowledge, there is no scan matcher suits for arbitrary environment when there are no hypothesis about its structure [6]. This paper presents different scan matchers which are applied to same environments and provides their comparison of their accuracy.

Two different kinds of scan matchers are discussed in this paper:

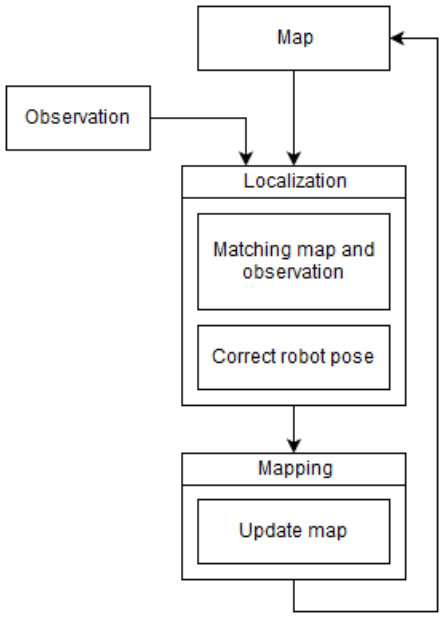


Fig. 1. The SLAM scheme

- a point to point scan matcher works only with data from the previous scan and the current scan;
- a map comparison scan matcher logs all scans in the environment map and uses it to match with data from current scan.

Three scan matchers are presented in this paper: Monte-Carlo scan matcher [3], Olson scan matcher [4], scan matcher based on the Hough transform [5]. The first one is a stochastic and bases on a random choice of robot pose correction. The second one bases on brute force search in the area that is likely to contain actual robot pose.. Finally the third bases on transformations in the Hough domain. The work results are presented in quantitative terms where it is possible and there are results of application these scan matchers in SLAM.

This paper presents first steps of algorithm research and the main purpose is to look through mathematical parts of considered approaches. The research area involves algorithm description and authors implementation [9] tested with computer simulation. The application in real conditions on a robot is next step that will be considered in the future work.

The paper is structured as follows: in Section II there is the description of existing comparison methods and a relevance of this paper; Section III provides a description of scan matchers that were tested; the result of the scan matchers comparison and usage recommendations are presented in Section IV.

II. Scan matching comparison

A. Problem definition

As it was mentioned above the essence of scan matching is to find a robot pose correction using data from sensors and map built previously. An example of input data is shown on Fig. 2.

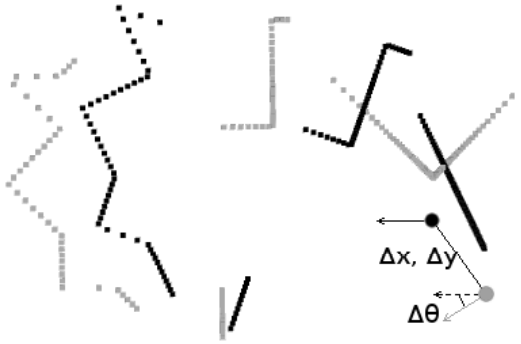


Fig. 2. Laser scanner data and robot position

The base estimation of robot offset $\Delta x, \Delta y, \Delta\varphi$ could be provided to scan matcher by odometry sensors. These sensors have uncaught noise in the output data so there is no way to recognize whether scan matchers afford correct answer. Of course, this noise could be reduced to the little value using groundtruth. Unfortunately, there is no groundtruth to the best of our knowledge for considered data sequences. So scan matchers were tested in the real world, but the estimation of execution was based on a visible robot trajectory and a built map.

So the formula (1) should be updated and it becomes:

$$\begin{pmatrix} q_{x,i} \\ q_{y,i} \end{pmatrix} = \begin{pmatrix} \cos(\hat{\Delta\varphi} + \xi_\varphi) & -\sin(\hat{\Delta\varphi} + \xi_\varphi) \\ \sin(\hat{\Delta\varphi} + \xi_\varphi) & \cos(\hat{\Delta\varphi} + \xi_\varphi) \end{pmatrix} \begin{pmatrix} p_{x,i} \\ p_{y,i} \end{pmatrix} + \\ + \begin{pmatrix} \hat{\Delta x} + \xi_x \\ \hat{\Delta y} + \xi_y \end{pmatrix}$$

where $\hat{\Delta\alpha}$ – the α value from odometry sensor, ξ_α – the noise value of α.

A value of ξ is supposed to be close to zero and this is an initial value for any scan matcher to find a correction value.

There is also another characteristic of scan matchers which ought to be considered and estimated – an execution time. The scan matching problem i.q. the SLAM problem applies to mobile robots that could have low performance processors, so it is desirable to choose a scan matcher that finds a correction as fast as possible.

B. Work motivation

A huge amount of different SLAM algorithms produce various approaches and implementations of scan matchers. Thus the purpose of this work is to establish the method of formal classification of several exciting scan matchers. Therefore the authors implementations of three scan matchers are considered and compared. The main goal was to determine where the implemented scan matchers could be used on the real sequences.

The sequences from MIT Stata Center Data Set [7] were chosen for scan matcher testing. Moreover considered scan matchers were used as a part of tinySLAM [8]. Using this package the map was built for each scan matcher. So it is possible to analyze how numeric values got from simulated world are applied to real world.

III. Scan matchers descriptions

A. Monte-Carlo scan matcher

This approach and its application to a SLAM problem were described in [3]. The main idea is based on a iterative random search of the most suitable robot offset. This loop takes some constant numbers of steps, for example there are 100 tries to find ”good” robot position. The value of robots' pose goodness is estimated with a scan cost function that estimates the similarity of scans. And if a good position is found then a tries counter is flushed and random search loop starts again in a narrowed area.

One approach to define the cost function bases on building a grid of cells where each cell presents the weight of a local area containing some points from the scan. So laser scan points are transformed to cells of a grid. After building the map from the first scan, the second scan is superimposed on it. So the scan cost function is defined as the sum of all cell weights where the points from the second scan are located.

The described approach is presented on Algorithm 1.

Algorithm 1 Monte-Carlo scan matching

```
pose_delta ← 0;
i ← 0;
total_count ← 0;
area ← U_σ(robot_pose)
while i < 100 && total_count < 1000 do
    observation ← shift laser_scan by pose_delta;
    scan_cost ← estimate observation with map;
    if scan_cost < min_cost then
        min_cost ← scan_cost;
        out_delta ← pose_delta;
        area ← U_σ(robot_pose + pose_delta);
        i ← 0;
    end if
    pose_delta ← random(area);
    i ← i + 1;
    total_count ← total_count + 1;
end while
```

Finally, the algorithm complexity is required to be estimated. This algorithm has a limit of step amount and for each step there is an operation **estimate** that complexity equals

$O(n)$, where n – a range of the laser scan. So the total algorithm complexity is estimated as $O(n \cdot n_{total\_count})$.

B. Olson scan matcher

In this approach it is required to find the greatest value of $p(x_i|x_{i-1}, u, m, z)$ – the probability of a robot position on step i using the known pose on a previous step x_{i-1}, the motion u, the environment model m and the observation z.

In the original paper [4] this probability is calculated in the following way:

$$p(x_i|x_{i-1}, u, m, z) = p(z|x_i, m) \cdot p(x_i|x_{i-1}, u)$$

where

- $p(z|x_i, m)$ – is the observation model that shows how likely is an observation if the previous position and the model are known.
- $p(x_i|x_{i-1}, u)$ – is the motion of robot that depends on odometry of robot.

The second term could be calculated taking into account information about typical odometry's noise and the most efforts are required to calculate the first term.

There are three steps of scan matching. The first step is to build a so-called lookup table using a previous scan – a grid map where each cell contains a probability of being occupied by any point of scan. This is pretty similar to Monte-Carlo approach. But a cost function is calculated as a multiplication of probability of all suitable cells. The example of lookup table is presented on Fig. 3

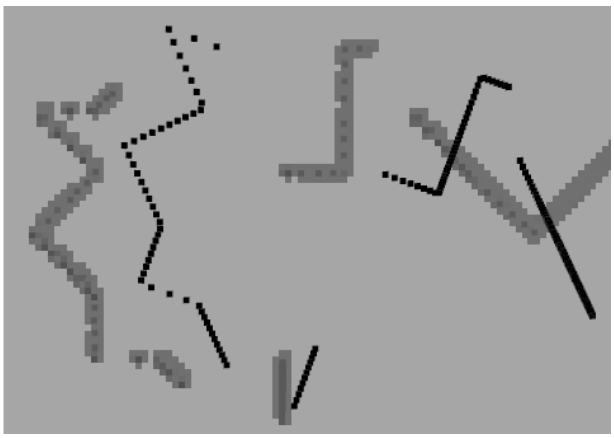


Fig. 3. A lookup table(gray) and a new scan(black)

The second step is to superimpose a new scan on a built lookup table and to calculate a probability of coincidence of two scans. The greatest probability $p(z|x_i, m)$ can be found by changing position x_i.

Then it is required to repeat the previous step for different x_i, calculate $p(x_i|x_{i-1}, u)$ for each robot pose and to find out the greatest probability $p(x_i|x_{i-1}, u, m, z)$ of the robot position.

The disadvantages of this approach are the fixed step resolution and time consuming brute-force method of searching x_i, however in the paper [4] there are some advices how to decrease the computation time including usage of GPU.

The algorithm complexity could be obviously estimated as a brute-force search complexity. So, if n_x, n_y, n_θ are amounts of steps for corresponding variable and on every step the current scan is required to superimpose on the look-up table with complexity $O(n)$ (where n – a range of the laser scan), the total algorithm complexity is estimated as $O(n \cdot n_x \cdot n_y \cdot n_{theta})$.

C. Hough scan matcher

This approach of localization is described in [5] and presents a point to point type of scan matchers. Every point from the each scan is transformed in special Hough coordinates (ϱ, θ). These coordinates present a distance and an angle, but they should not confused with the polar ones. For every point in Euclidean coordinates (x_0, y_0) there is a set of lines that contain this point. Line coefficients presented in the (k, b) notation from an equation $y_0 = kx_0 + b$ are not good enough, as there could be vertical lines like $x = 2$ which can not be defined using the notation. It is more useful to store information about each line using a distance ϱ from the zero point $(0, 0)$ to this line, and an angle θ between the OX axis and a line that includes a normal to this line (Fig. 4).

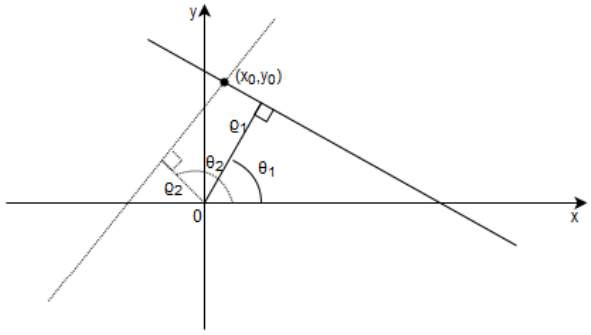


Fig. 4. The example of two possible lines in (ϱ, θ) notation contained a point (x_0, y_0)

And it can be shown that a set of lines crossing a point (x, y) could be built with the following rule:

$$\varrho = x \cdot \cos(\theta) + y \cdot \sin(\theta) \quad (2)$$

The expression in the right part of (2) could be simplified to $\varrho_0 \cdot cos(\theta + \theta_0)$ and it presents a shifted sine function. So the set of points in Euclidean coordinates $\{(x_i, y_i)\}_{i=1}^n$ will be transformed in a set of sets of points $\left\{\{(\varrho_j, \theta_j) : \varrho_j = x_i \cdot \cos\theta_j + y_i \cdot \sin\theta_j\}_{j=1}^m\right\}_{i=1}^n$ under the Hough transform.

For example the Hough transform for one of the scans (black) from the Fig. 2 is presented on the Fig. 5. As it was mentioned above the Hough domain consist of a sine functions set. The brightness of point (ϱ, θ) depends on an amount of laser scan points are contained in the line (ϱ, θ). For example bright point locates at $\theta \approx \pi/3$ and small ϱ value correspond to the right line of black scan from Fig. 2

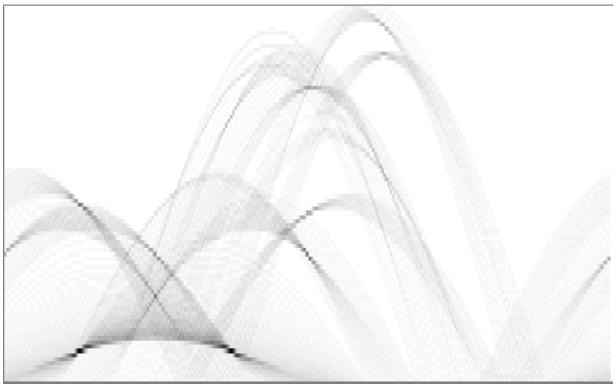


Fig. 5. Hough domain for the one of the scans from Fig. 2

Hough domain is built for the current scan and the previous one. And the following equation connects this two Hough domains with offset$(\Delta\rho, \Delta\varphi)$ between two scans:

$$HD_{prev}(\varrho, \theta) = HD_{curr}(\varrho + \Delta\rho, \theta + \Delta\varphi)$$

where HD_i – the 2D array that presents Hough domain like on the Fig. 5.

After the Hough domain was built the rotation angle can be defined and then it is possible to estimate an offset. The rotation angle could be estimated using Hough spectrum that is built by the following rule:

$$HS(\theta) = \int_0^\infty HD^2(\varrho, \theta) d\varrho; \tag{3}$$

The formula (3) means that the Hough spectrum is defined as a ϱ energy for every angle θ. So if there is a small offset error, the two Hough spectrums for the previous and current scans will be shifted by θ with some value $\Delta\varphi$:

$$HS_{prev}(\vartheta) = HS_{curr}(\vartheta + \Delta\varphi) \tag{4}$$

And it is possible to match this value of shift using a correlation function.

Unfortunately, the same approach to find the offset $(\Delta x, \Delta y)$ is not useful as it is possible to find $\Delta\rho$ but not separately Δx or Δy. And the found angle $\Delta\varphi$ can not help to compute it. This is not a polar angle between two scans but an angle of robot orientation in space. So the authors implementation has no ability to find $(\Delta x, \Delta y)$ and it should be found using another approaches.

The algorithm complexity is defined as a sum of complexities of building Hough domain, computing Hough spectrum and computing correlation function. The Hough domain is built using $O(n)$ operations, where n – a range of the laser scan. The Hough spectrum building needs every cell in Hough domain to be executed. There are $n_\varrho \cdot n_\theta$ cells in Hough domain, so the Hough spectrum calculation complexity defines as $O(n_\varrho \cdot n_\theta)$. So the total complexity is approximated with $O(n + n_\varrho \cdot n_\theta)$.

IV. Testing and results

Scan matchers were tested on both simulated data and real dataset from MIT [7]. Simulated data could be divided in two parts: the environments that mostly consist of straight lines, and the other. Moreover when an environment is simulated there is an option to add some noise to odometry data to scan matchers either with some error or without.

Testing on a real data is more complex because there is no knowledge about accurate odometry and therefore there is no way to determine if a scan matcher works well. Thus all considered scan matchers were integrated in a SLAM algorithm. For this purpose the tinySLAM [8] was chosen. The accuracy of a scan matcher was estimated with a map and robot trajectory that had been built during a SLAM process. Visual comparison of built maps and trajectories is supposed to be enough to understand whether a scan matcher works well or not.

As it was mention above the time execution is computed for every test too. This time depends on algorithms complexity that were described above. All this complexities are presented in the Table I

TABLE I. Complexity of scan matching algorithms

	Monte-Carlo	**Olson**	**Hough**
Complexity	$O(n \cdot n_{count})$	$O(n \cdot n_x \cdot n_y \cdot n_\theta)$	$O(n + n_\varrho \cdot n_\theta)$

The tests presented below were executed on Intel® Core™ i7-860 4x2.8GHz with DDR3 8GB, Ubuntu Xenial x64.

A. Testing in simulated straight world without an error

Firstly, all scan matchers were tested in a computer simulated worlds without noise in odometry. In this test two scans observed from two different places in the world with robot offset $(\Delta x, \Delta y, \Delta\theta)$ are used as input to scan matchers. This offset is provided to the scan matchers too. So these two laser scans contain some similar points with similar coordinates. The difference is in ranges of each scans. In fact these two scans present data from different points in the space, so there are points in the one scan that are not observable from the second point of view. So sets of points $\{p_i\}$ and $\{q_i\}$ contain subsets of corresponding points but there definitely are points which are contained in the one scan and are not contained in another. So scan matchers should recognize that the error between first scan and shifted second scan equals to zero.

Moreover, in this case scan matchers are tested in the room-like worlds which contain many straight lines passed for long distances. In this case scans are not very different if the offset is small. So this kind of test could be used as a some of sanity check.

The results of testing in this case are presented in the Table II.

TABLE II. Results of test *A*

Scan Matcher	**Time**	Δx	Δy	$\Delta\theta$
Monte-Carlo	0.02 sec	0	0	0
Olson	0.3 sec	0	0	0
Hough	0.2 sec	-	-	0

This result shows that all scan matchers do not introduce any error if there is no mistake in odometry. Monte-Carlo is faster in 10 times than other.

In this table symbol "-" in a row of Hough scan matcher shows that author's implementation does not allow to calculate offset $\Delta x, \Delta y$. Taking into account deltas, calculated by this scan matcher, the output becomes absolutely incorrect, so this part of Hough scan matcher should be researched with more attention in the future.

Up to this test it seems that Monte-Carlo scan matcher is the best.

B. Testing in simulated straight world with an error

In this test case there is some odometry error: a noise error added to robot offset $(\Delta x, \Delta y, \Delta\theta)$. So these sets of points have no points with equal coordinates. All coordinates are shifted and rotated by some random values. This test is closer to real case as there is always some odometry error in the real data sequence. In this case scan matchers should determine this noise. As the error was generated manually so it is possible to estimate whether the scan matchers localize and estimate the noise correctly.

In this test the world consists of straight lines, and it is easy to match the error with the naked eye. The difficulties appear in a fractal world when it is impossible to define whether there are any rotation or not. Also tests in corridor-like world will not define correct (ξ_x, ξ_y) offset errors, but scan matchers have to define correct a ξ_θ rotate error.

The result of testing in this case with odometry noise about 0.2 meters and 0.2 rad is presented in the Table III.

TABLE III. RESULTS OF TEST *B*

Scan Matcher	Time	Δx	Δy	Δθ
Monte-Carlo	0.02 sec	0.011	0.009	0.02
Olson	0.3 sec	0.001	0.001	0.001
Hough	0.1 sec	-	-	0.008

In this test the Monte-Carlo scan matcher shows the best time again, but now there is an error that is greater than the error of other scan matchers. It can be explained again by random behavior of Monte-Carlo scan matcher.

The error in Olson scan matcher can be explained by the value of search step in its implementation. Decreasing this step value makes scan matcher work more accurate, but slower.

Talking about Hough scan matcher it's step of discretization should be mentioned – it is an increment of Hough spectrum i.e. the field of search. The lower this step the lower an error of output.

C. Testing in simulated random world without an error

In this case there is one extreme condition – a random world. This means that the input environment presents set of linked chaotic points. These points are very close to each other so this world is an implementation of 2D random process. So it is impossible to align scans to each other when there is a huge robot offset $(\Delta x, \Delta y, \Delta\theta)$. There could be extremely different view areas from observation points of this two scans. So in this case the scan matchers should find the zero offset between two scans, but it depends on the random world.

The result of testing in this case is presented in the Table IV.

TABLE IV. RESULTS OF TEST *C*

Scan Matcher	Time	Δx	Δy	Δθ
Monte-Carlo	0.3 sec	0	0	0
Olson	2.7 sec	0	0	0
Hough	0.8 sec	-	-	0.008

This result demonstrate the restriction for the Hough scan matcher - because of an absence of straight lines or appearance of some fractal areas on scan this scan matcher may calculate the result with some error. However this error is about a searching delta in implementation of algorithm and it can be reduced as it was mentioned in a previous test.

Monte-Carlo and Olson scan matchers have performed the same result as in the test A, so they are not sensitive to the nature of the world if there is no error in odometry.

D. Testing in simulated random world with an error

This case is the closest to real world. There is a random environment and some odometry error. That means that the input scans are so different that it is impossible to compare each point of one scan to each point of another. Moreover, the odometry error is added and two input scans become absolutely different. Because of this difference a little error in scan matcher's output may be presented. It is impossible to get rid of this error but the focus is to minimize it.

The result of testing in this case is presented in the Table V.

TABLE V. RESULTS OF TEST *D*

Scan Matcher	Time	Δx	Δy	Δθ
Monte-Carlo	0.3 sec	0.08	0.09	0.05
Olson	2.7 sec	0.0005	0.0004	0.001
Hough	0.8 sec	-	-	0.016

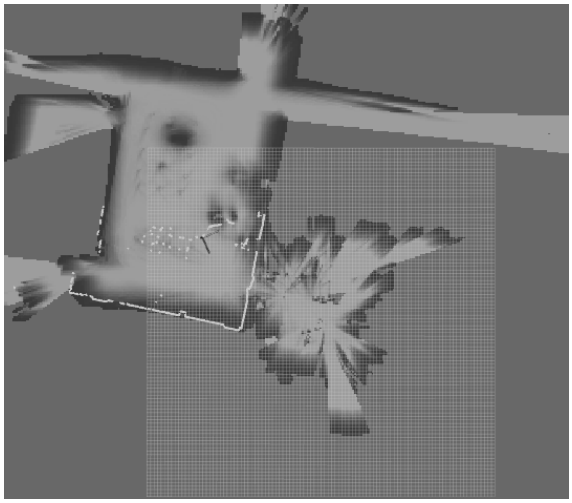
This result shows the main difference between random search of Monte-Carlo, brute force of Olson and searching in another space of Hough scan matchers. If the amount of steps in Monte-Carlo is not enough, the best position will not probably be found.

In this test the best result is performed by Olson scan matcher, but the cost of this profit is considerable. In SLAM algorithm it is undesirable to spend so much time on localization, despite the fact that the accuracy is very high.

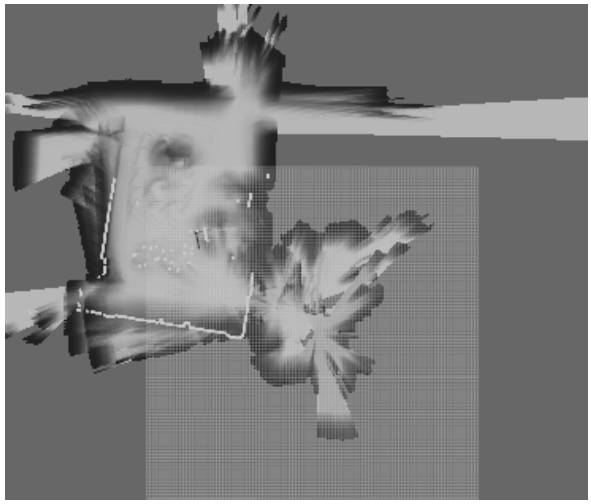
E. Testing on real datasets

There is no opportunity to get quality estimation of scan matching performance, as there is no information about error $(\xi_x, \xi_y, \xi_\theta)$. So there is a test on a real data sequence got from [7]. And as the result of scan matcher work there are the map built in SLAM task. If a map has the sharp bound, it means that the scan matcher completed its execution correctly without mistakes. The more blur is visible on a picture the more mistakes have been done by scan matcher.

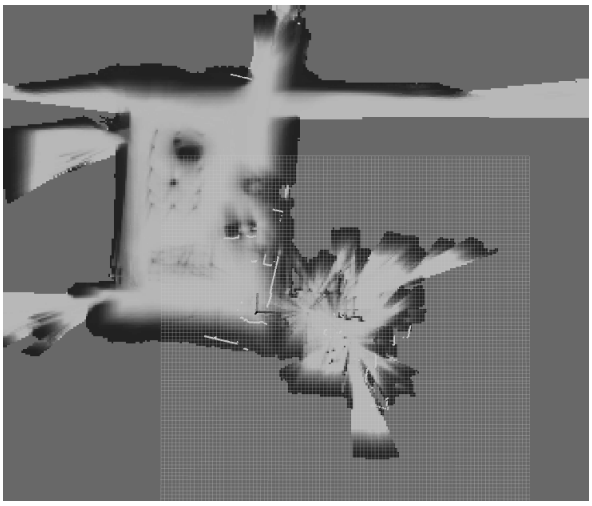
There are four maps below. On Fig. 6a you can see the result of building map during tinySLAM performance without



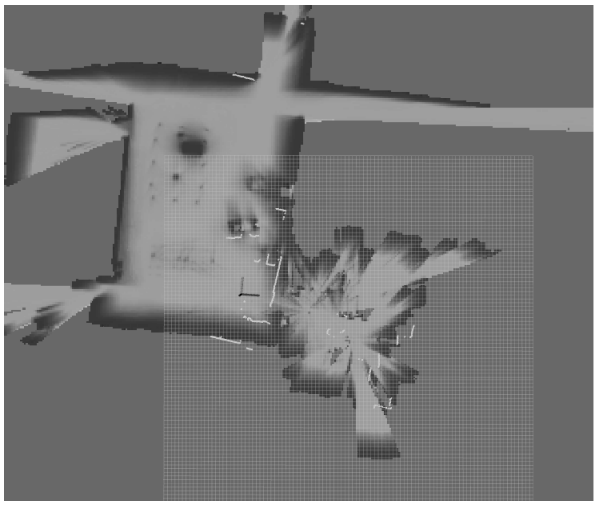
(a) The map built using only odometry data



(b) The map built using Hough scan matcher



(c) The map built using Olson scan matcher



(d) The map built using Monte-Carlo scan matcher

Fig. 6. Maps built with different scan matchers

any scan matcher. That means that instead of localization the robot pose was calculated using only odometry. As you can see a map is pretty well, so the odometry has not many mistakes. But in the top of a picture there is a split of lines, that appeared falsely.

On Fig 6b there is a map built after including the Hough scan matcher in SLAM process. Mistakes in building this map can be explained by appearance of oscillation in angle correction during scan matching. It is hard to explain or to predict this oscillation.

The map that was build after including Olson scan matcher in SLAM process is presented on a Fig. 6c. As the scan matching takes a lot of time in this case, the solution to decrease the quality was taken. This helped to finish a process in the appropriate time.

Finally on a Fig 6d there is the map built using Monte-Carlo scan matcher. The result of this SLAM is the most accurate and it can be explained by the fact that the odometry error was little. That is the reason why SLAM with Mote-Carlo scan matcher completed both fast and accurate.

On Fig. 7 - 9 the robot trajectories are presented depending on the used scan matcher. Every trajectory was superimposed on the odometry, so the differences could be matched. Monte-Carlo scan matcher shows the best result, because on a Fig 9 it is shown that the begging and the ending of robot moving coincide.

V. Conclusion

Scan matching is an important part of Simultaneous Localization and Mapping process and a choice of scan matcher should be very elaborate. In case of real time performance the step of localization have to proceed quickly and accurate. And there is a classic discussion between a high speed of taking a decision and an absence of errors.

In this paper three scan matchers are considered from both of these points of views. The result of this researched shows that the choice of scan matcher depends in conditions of usage.

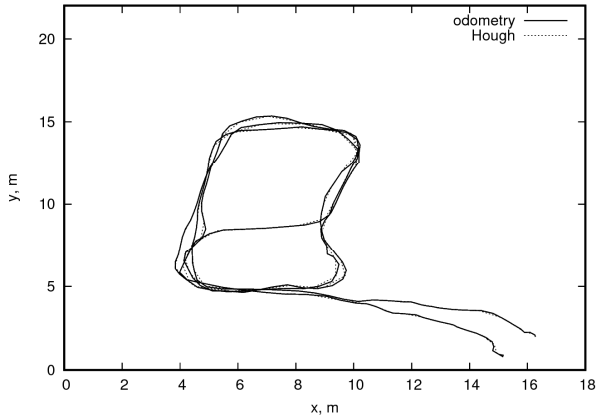


Fig. 7. Groundtruth built by odometry or with usage Hough scan matcher

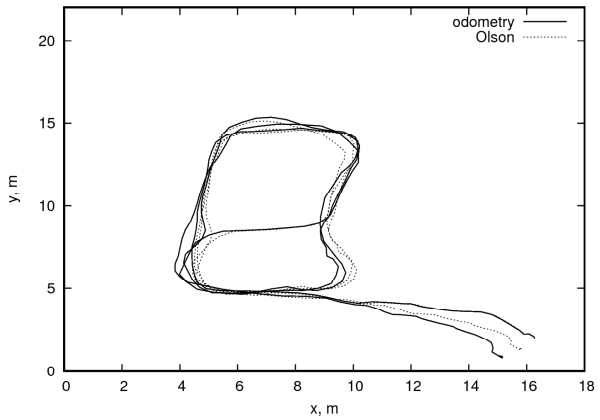


Fig. 8. Groundtruth built by odometry or with usage Olson scan matcher

For example if it is known that the world consists of long straight lines the best idea is to choose Hough scan matcher. If nothing is known about the world and a lidar scan except of an approximate value of an error, it is possible to use Olson scan matcher.

But it would be honest to admit that Monte-Carlo scan matcher shows the best result when it is included in SLAM. That means that on real data where an error of lidar scan or odomerty is unknown, Monte-Carlo is the most preferred among tested scan matchers. However the gotten results depends on MIT datasets and some other datasets may be not such accurate. For these cases the only opportunity to get a correct result is to choose another scan matcher and spend more time on localization.

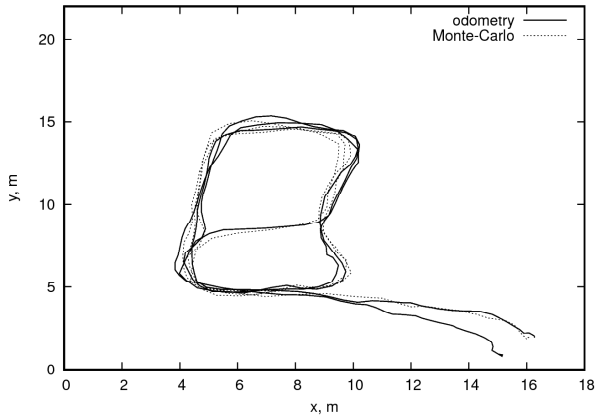


Fig. 9. Groundtruth built by odometry or with usage Monte-Carlo scan matcher

VI. Acknowledgment

Authors would like to thank JetBrains Research for provided support and materials for working on this research. Some parts of the paper has been prepared within the scope of project part of the state plan of the Board of Education of Russia (task # 2.136.2014/K).

References

[1] J.J. Leonard and H.F. Durrant-Whyte, "Mobile robot localization by tracking geometry beacons", *IEEE Trans. Robotics and Automation*, vol. 7, no. 3, Jun. 1991, pp. 376-382.

[2] H.F. Durrant-Whyte and T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms", *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, 2006, pp. 99-110.

[3] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo Localization for Mobile Robots", *In Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1975-1979.

[4] E. Olson, "Real-time correlative scan matching", *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 4387-4393.

[5] A. Censi, L. Iocchi and G. Grisetti, "Scan matching in the Hough domain", *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 2739-2744.

[6] Q. Li, F. Muller, A. Wenzel and T. Rauschenbach, "Simulation-based comparison of 2D scan matching algorithms for different rangefinders", *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, Sep. 2016, pp. 924-929

[7] MIT Stata Center Data Set,
Web: http://projects.csail.mit.edu/stata/downloads.php

[8] tinySLAM package for ROS, Web: http://wiki.ros.org/tiny_slam

[9] Source code SLAM framework package for ROS, Web: https://github.com/OSLL/slam-constructor