mathematics

MDPI

Article

Optimal Robot Pose Estimation Using Scan Matching by Turning Function †

Bahram Sadeghi Bigham [1,*], Omid Abbaszadeh [2], Mazyar Zahedi-Seresht [3], Shahrzad Khosravi [3] and Elham Zarezadeh [4]

[1] Department of Computer Science, Faculty of Mathematical Sciences, Alzahra University, Tehran 1993893973, Iran
[2] Department of Computer Engineering, University of Zanjan, Zanjan 4537138791, Iran
[3] Department of Quantitative Studies, University Canada West, Vancouver, BC V6Z 0E5, Canada
[4] Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan 4513766731, Iran
* Correspondence: b_sadeghi_b@alzahra.ac.ir
† This research has been facilitated by Data Science Lab at the Department of Computer Science, Alzahra University.

Abstract: The turning function is a tool in image processing that measures the difference between two polygonal shapes. We propose a localization algorithm for the optimal pose estimation of autonomous mobile robots using the scan-matching method based on the turning function algorithm. There are several methodologies aimed at moving the robots in the right way and carrying out their missions well, which involves the integration of localization and control. In the proposed method, the localization problem is implemented in the form of an optimization problem. Afterwards, the turning function algorithm and the simplex method are applied to estimate the localization and orientation of the robot. The proposed algorithm first receives the polygons extracted from two sensors' data and then allocates a histogram to each sensor scan. This algorithm attempts to maximize the similarity of the two histograms by converting them to a unified coordinate system. In this way, the estimate of the difference between the two situations is calculated. In more detail, the main objective of this study is to provide an algorithm aimed at reducing errors in the localization and orientation of mobile robots. The simulation results indicate the great performance of this algorithm. Experimental results on simulated and real datasets show that the proposed algorithms achieve better results in terms of both position and orientation metrics.

Keywords: localization; optimization; SLAM; image processing autonomous vehicle; robot; pose estimation

MSC: 65D18; 68U05; 90C23; 68W01; 70B15

Citation: Sadeghi Bigham, B.; Abbaszadeh, O.; Zahedi-Seresht, M.; Khosravi, S.; Zarezadeh, E. Optimal Pose Estimation for Robot Using Scan Matching based on Turning Function. *Mathematics* **2023**, *11*, 1449. https://doi.org/10.3390/math11061449

Academic Editor: Konstantin Kozlov

Received: 10 February 2023
Revised: 11 March 2023
Accepted: 14 March 2023
Published: 16 March 2023

1. Introduction

The most essential goal of robotics is to develop autonomous mobile robots that behave, move, think, and communicate like people. Mobile robots are now widely employed in daily life, and they are becoming increasingly intelligent to assist people in tough and dangerous activities. Since mobile robots are utilized in a variety of settings, one of the most important criteria for mobile robot applications is that the robot knows its present position related to a fixed and reference coordinate. For example, in situations where the robot must move to attain a goal, an accurate estimation of its current position is required. Similarly, accurate information on the robot's location is necessary for exploratory applications so that the robot can avoid duplicate paths [1].

In mobile robotics, there are four main elements: localization, map building, collision avoidance, and path planning. One of the most fundamental requirements in robotics

is localization. Localization refers to the robot's ability to calculate its correct position in the environment relative to a reference point at any given time and to determine its direction [2,3]. If the robot's localization is incorrect, it will have difficulty performing other tasks that require precise location data [4].

The use of robot sensors is critical in solving the localization problem. However, these sensors are often unreliable, making it challenging to determine the robot's exact location. Mobile robots can use geometric tools to guess their location on a map and eliminate localization errors as much as feasible. As a result, many efforts have been made in the previous decade among scholars to find effective answers to this problem, and several methods have been proposed [5,6]. Scan matching is a strategy that has received considerable attention in recent years among scholars working in the field of improving robot location estimates. It is a method that compares two consecutive measurements received from robot sensor data and an environment map until the two measurements are the most closely aligned. This can be accomplished by equipping the robot with sensors and employing techniques such as particle filter estimation, Kalman filter estimation, and scan matching, among others. With the sort of environment in which the robot operates, several methods and algorithms have been presented for this goal. First, a spatial description of the robot's expected position is simulated on a complete and accurate map of the environment using the suggested algorithm. After that, the simulated model is compared to a spatial description derived from laser distance sensor data.

The remainder of this paper is organized as follows. Section 2 of this paper presents a review of the relevant literature. We present the key concepts and definitions that are necessary to discuss our methods in Section 3. Section 4 introduces our algorithm in detail. Section 5 presents the experimental results of the proposed algorithms on the simulated environment. Section 6 concludes with the discussion and points to future work.

2. Related Works

In [7], one of the most important strategies based on the scan matching of environmental geometry features is presented. The algorithm attempts to match data from the laser distance sensor, such as line segments and break points, to calculate the robot's position. In Banerji et al. [8], a method is presented for precisely determining the position of a mobile robot using laser distance sensor data. In this study, the authors use the closest neighbor technique to compare the two images formed by the robot's observations and the environment map and then determine the robot's position. In [9], a topological model-based scan-based approach to mobile robot localization is presented. Topological nodes are used as reference points in this strategy. The robot now calculates its location using the RANSAC approach by matching the geometric relationships between the points collected from the robot sensors and the reference scan.

In [10], a scan-matching technique based on a laser distance sensor for approximating the location of a mobile robot is presented. This technique always attempts to find the correct matches between the two sets by finding correct matches and performing sequential transformations, which considers a simulated polyline from the robot's expected location in the map as a prediction model and a real polyline extracted from robot sensor data. This allows for an accurate estimation of the robot's likely position on the map. The technique given in [11] is another effort on mobile robot localization that uses scan matching. The laser distance sensor is used to retrieve data, and the geometric aspects of the environment are used to localize the robot. In [12], a robotic navigation approach is described in which the points received from each scan are first imaged in networked map cells, and then scan matching is used to determine the exact location. Li et al. [13] developed a novel repetitive motion control approach to handle the limitations imposed by external noises by considering the inherent noise suppression capability with the formulation of a dynamical quadratic program. In [1], a method is proposed to determine the 3D relative pose of pairs of communicating robots by using human pose-based key points as correspondences.

3. Problem Statement

A robot's environment is supposed to be static and polygonal, with a limited number of obstacles and access to a map of the environment at all times to characterize the robot localization problem. Table 1 provides a list of abbreviations used in this paper, with each abbreviation accompanied by its corresponding full term. On a known map, Figure 1 depicts the virtual and real sensor concepts.

Table 1. List of abbreviations.

Abbreviation	Definition	Abbreviation	Definition
RP	Robot Position	VP	Virtual Position
RSD	Robot Sensor Data	VSD	Virtual Sensor Data
TF	Turning Function	SS	Shortstraw algorithm

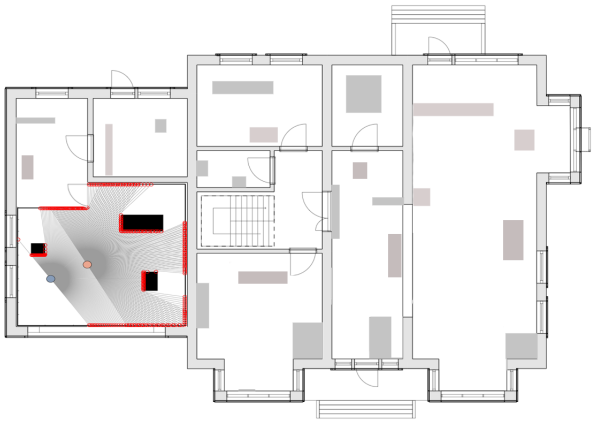


Figure 1. Schematic map of the office environment used for localization in this study.

Real spatial descriptions and virtual spatial descriptions are two types of shapes that may be produced from two-dimensional laser distance sensor data. Virtual and real sensors both employ the same sort of sensor (Sick LMS200-30106), denoted as $g(m, f, q)$, where m stands for the sensor's maximum range, f stands for its field of view, and q stands for its angular resolution. Figure 2 depicts the mobile robot configuration in two-dimensional Cartesian coordinate space. The O point, denoted by X_r and Y_r for robot localization in the plane, is assumed to be the robot position's origin. As a result, the following is the definition of the robot's and virtual sensor's positions in Figure 2:

$$RP = \begin{pmatrix} x_r \\ y_r \\ \theta_r \end{pmatrix}, \qquad VP = \begin{pmatrix} x_v \\ y_v \\ \theta_v \end{pmatrix}. \tag{1}$$

The position difference values, denoted by $\hat{T}$, are as follows:

$$\hat{T} = RP - VP = \begin{pmatrix} \widehat{\Delta_x} = \sum_{i=1}^{I} \delta_{x_i} \\ \widehat{\Delta_y} = \sum_{j=1}^{J} \delta_{y_j} \\ \widehat{\Delta_\theta} = \sum_{k=1}^{K} \delta_{\theta_k} \end{pmatrix} \tag{2}$$

where δ_{θ_k} is the difference of orientation values at time k, and δ_{y_j} and δ_{y_j} are the difference values of position at time i and j. $D^r = [d_1^r, d_2^r, \ldots, d_n^r]$ is the set of measured distance by real sensors, and $D^v = [d_1^v, d_2^v, \ldots, d_n^v]$ is the set of measured distances by virtual sensors. $\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$ is the set of laser angles, where $n = (f/q) + 1$ is the number of measured distances. The angle of ϕ_i for distance d_i is calculated as follows [14]:

$$\phi_i = \phi_{i-1} + q, \qquad \phi_1 = 0. \tag{3}$$

In the local coordinate system, Figure 3 displays the virtual sensor and robot sensor data from Figure 2. The following linear equation is used to display any RSD and VSD point in a local coordinate system.

$$\begin{aligned} RSD_i &= \left(cos(\phi_i)d_i^r, \quad sin(\phi_i)d_i^r\right) \\ VSD_j &= \left(cos(\phi_j)d_j^v, \quad sin(\phi_j)d_j^v\right). \end{aligned} \tag{4}$$

Thus, the problem of robot pose estimation is solved if $\delta_{\theta K}$, δ_{yJ}, and δ_{xI} are optimized.

$$\widehat{RP} = \{VP + \widehat{T} | \delta_{x_i} \leq \epsilon_x, \delta_{y_j} \leq \epsilon_y, \delta_{\theta_k} \leq \epsilon_\theta\} \tag{5}$$

and

$$\varepsilon_x = |x_r - \hat{x}_r|, \qquad \varepsilon_y = |y_r - \hat{y}_r|, \qquad \varepsilon_\theta = angdiff(\theta_r, \hat{\theta}_r) \tag{6}$$

where $-\pi \leq angdiff(a, b) \leq \pi$ is the difference between two angles, a and b.

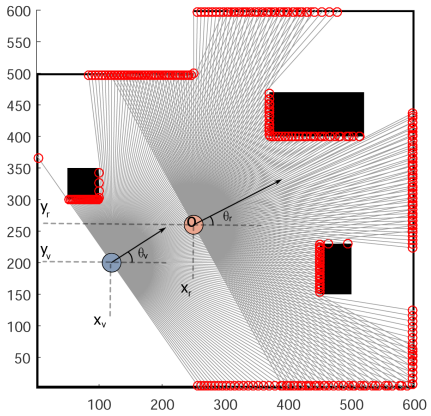


Figure 2. Robot configuration in two-dimensional space.

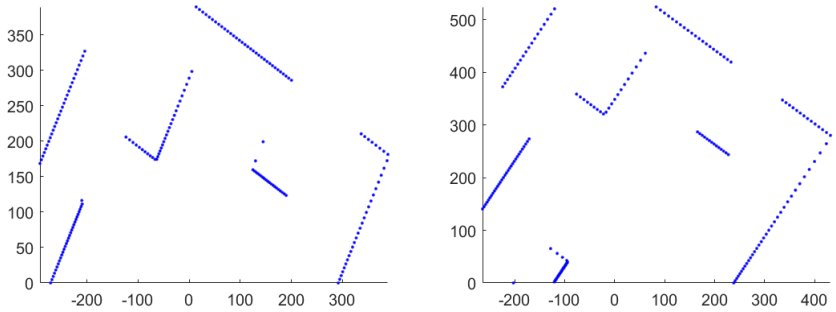


Figure 3. Visualization of laser range data for both real sensor RSD (**left**) and virtual sensor VSD (**right**).

4. Turning Function Algorithm

Turning function is a function that assigns a real value to each polygon/chain, which can be used to measure the similarity or difference between two polygons/chains. This function has already been used in several applications, the most recent one being used by Padkan et al. [15] to match fingerprints. The proposed turning function (TF) algorithm starts by receiving polygons extracted from RSD and VSD data from two-dimensional laser distance sensors. Then, for each sensor scan, a histogram is generated. The algorithm attempts to maximize the similarity between paired histograms by converting them into a

single coordinate system, enabling the accurate estimation of the true difference between the two desired positions. Figure 4 represents the TF algorithm's steps.

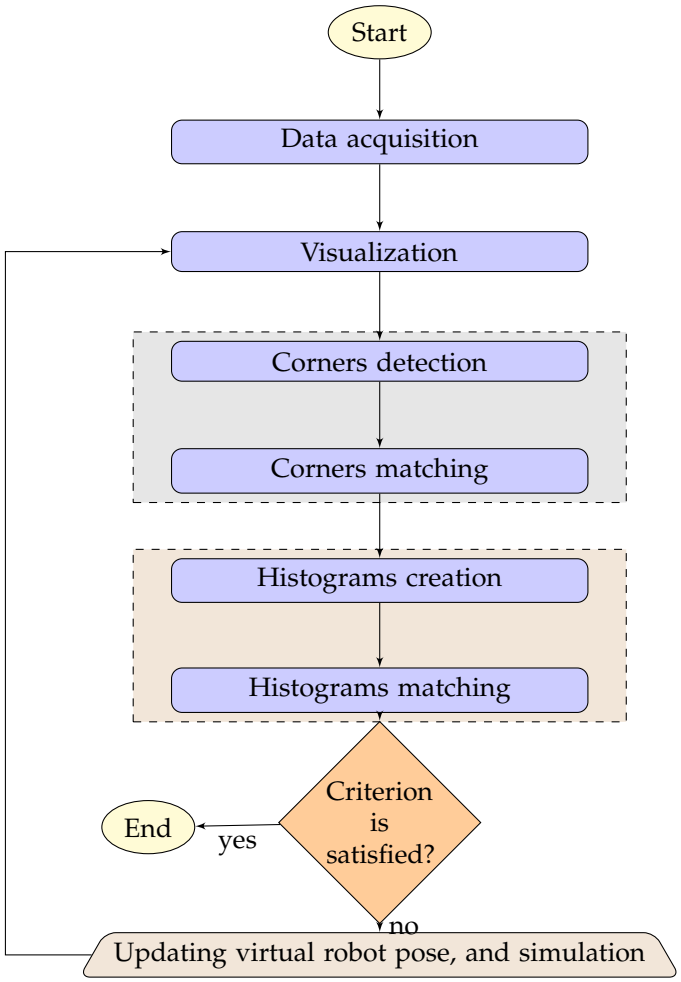


Figure 4. A brief overview of the proposed algorithm.

Corner detection, histogram creation, histogram matching, position calculation, and orientation difference are all steps in the proposed location estimation algorithm. The algorithm starts each cycle by matching paired histograms and calculating the distance function in both sets, with angle detections of both images and histogram creation for each scanned form. This reduces the difference between the two positions by calculating the robot's position and orientation difference. This process is repeated until the algorithm converges or meets a stopping criterion.

4.1. Corner Detection

The corners, which are commonly used as critical points for localization problems, are one of the most important features that provide information about the overall structure of the environment. Corner points are points where the angle formed by two intersecting line segments passing through them is greater than or equal to a predetermined value. As a result, this section aims to separate corner points from VSD and RSD datasets. Many algorithms for corner detection have been proposed in recent years. In this paper, we use the shortstraw (SS) algorithm proposed by Wolin et al. [16] to extract corner points from the laser distance sensor data set in this study. The data collected by the robot sensor is re-sampled and redrawn as the first step in this algorithm. The procedure for reproducing points from RSD and VSD datasets is shown in Algorithm 1. The goal is to create new points

that are evenly spaced from one another. The format of each scan remains unchanged, with only points that are equidistant from other remaining ones.

Algorithm 1 Point resampling

Input:
Output:
$i \leftarrow 2$
$S \leftarrow 1$
$P_1 \leftarrow points_i$
while $i \leq length(points)$ **do**
    $P_2 \leftarrow points_i$
    $D \leftarrow ||P_1 - P_2||^2$
    if $D > S$ **then**
        $P_{3,x} \leftarrow P_{1,x} + (P_{2,x} - P_{1,x})/D * S$
        $P_{3,y} \leftarrow P_{1,y} + (P_{2,y} - P_{1,y})/D * S$
        P_3 is located exactly S Euclidian distance away from the P_1 in the slop direction of the straight line from P_{i-1} to P_i
        Append P_3 to the resampled array
        $P_1 \leftarrow P_3$
    else
        $i \leftarrow i + 1$
    end if
end while

The length of an interspacing distance is used to find corner points in the second step of the SS algorithm. This interspacing distance is determined as follows for the re-sampled P_i point:

$$straw_i = |P_{i-\omega}, P_{i+\omega}|. \quad (7)$$

When using the SS algorithm, ω is a static window with a length of one ($\omega = 1$). To compute the set of virtual straws for the points $P_{\omega+1}$ to $P_{N-\omega}$, where N is the number of re-sampled points, we first calculate the set of virtual straws for the points $P_{\omega+1}$ to $P_{N-\omega}$. As they approach a curve in the illustration, the length of these virtual straws shortens. As a result, corner points are sites where the length of the related straws is smaller than a specified value. The corners acquired from the recreated sets of actual and virtual sensors using the SS method are shown in Figure 5. The method of extracting corners from new sets, RSD, and VSD, is shown in Algorithm 2 [10].

Algorithm 2 Corner detection (points)

Input:
Output:
Corners $\leftarrow 1$
Straws $\leftarrow \{\}$
$t \leftarrow 0.99$
for $i \leftarrow w + 1$, number of $points - w$ **do**
    $Straws(i - w) \leftarrow ||points(i - w) - points(i + w)||^2$
end for
$threshold \leftarrow median(straws) \times t$
for $i \leftarrow w + 1$, number of $points - w$ **do**
    if $straws(i - w) < threshold$ **then**
        Append($corners, i$)
    end if
end for
Append($corners$, number of $points$)

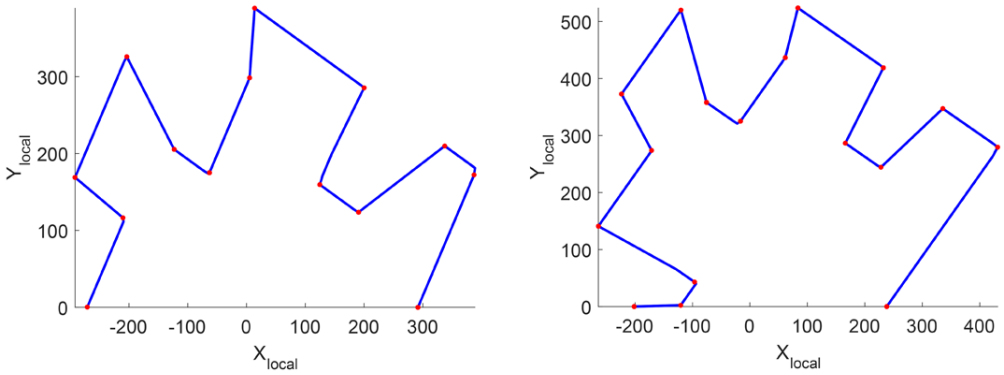


Figure 5. Correctly detected corners of (**left**) VSD and (**right**) RSD, which are marked with red points.

We then proceed to eliminate the additional corners from the set after the corner detection. All three corners are continuously examined to ensure that they are in all directions for this purpose. If the Is-Line function returns True for each line between c_l and c_n, then c_l, c_m, and c_n are considered parallel. In this case, the set of corners is reduced by removing the middle corner, i.e., c_m. Algorithm 3 shows how to use this function. In the following steps, the algorithm will be fed a set of corner points extracted from laser distance sensor data.

Algorithm 3 Is-Line (points, a, b)

Input:
Output:

$threshold \leftarrow 0.95$
$D \leftarrow ||points_a - points_b||$
$PathD \leftarrow 0$
for $i \leftarrow a, b-1$ **do**
    $PathD \leftarrow PathD + ||points_i - points_{i+1}||^2$
end for
if $D/PathD > threshold$ **then**
return True
else
return False
end if

4.2. Histogram Creation

The histogram creation process is described in Algorithm 4. The function may become non-smooth as a result of scaling, making it difficult to compare the two shapes' similarities. As a result, the first step in the algorithm is to maintain the scale, which includes normalizing the figure's total length to one. The polygon's length will eventually be between 0 and 1 due to the normalization. At a certain point on the shape boundary, the TF algorithm begins. The angle between the edge and the horizontal axis (X-axis) in the counterclockwise direction is calculated to determine the starting point angle. However, the angle between the previous and the next edge of the point is calculated for the following points. For counterclockwise concave vertices, all considered angles are positive; for clockwise concave vertices, all considered angles are negative. By rotating to the left, the angle becomes bigger, and by rotating to the right, the angle becomes smaller. The maximum sum of angles in the rotation function is v if the angle created by the first line segment in the positive direction of the X axis is $\Theta_A(L) = 2\pi + v$. The process of creating a histogram from a set of RSD and VSD corner points is illustrated in Figure 6.

Algorithm 4 TF (corners)

Input:

Output:

$total\_length \leftarrow 0$
for $i \leftarrow 1$, number of corners **do**
  $Vector(i) \leftarrow Corners(i+1) - Corners(i)$ ▷ Vectors are create in counterclockwise direction
  if $i = Corners$ **then**
    $Vector(i) \leftarrow Corners(1) - Corners(i)$
  end if
  $total\_length = total\_length + \sqrt{Vector(i,1)^2 + Vector(i,2)^2}$
end for
for $i \leftarrow 1$, number of Vectors **do**
  $Vector(i,1) \leftarrow vector(i,1)/total\_length$
  $Vector(i,2) \leftarrow vector(i,2)/total\_length$
end for
$V \leftarrow (1,0,0)$
for $i \leftarrow 1$, number of Vector **do**
  $h(i,1) \leftarrow |Vector(i).|$
  if $i = 1$ **then**
    $h(i,2) \leftarrow Vector(i).V$
    $(0,0,,Z) \leftarrow Vector(i) \times V$
  else
    $h(i,2) \leftarrow Vector(i).Vector(i+1)$
    $(0,0,,Z) \leftarrow Vector(i) \times Vector(i+1)$
  end if
  if $Z < 0$ **then**
    $h(i,2) \leftarrow h(i,2) \times -1$
  end if
end for

The set of corner point coordinates and the VSD, respectively, are denoted by: $C^R = [c_1^r, c_2^r, \ldots, c_n^r]$ and $C^V = [c_1^v, c_2^v, \ldots, c_m^v]$. Then, between the corner points of each set, a vector is created sequentially (counterclockwise). Next, by utilizing the Euclidean distance (for two consecutive points c_i and c_{i+1}, for example), the generated vector length is calculated as follows:

$$\begin{aligned} Vectore(X,Y) &= (X_{c_i} - X_{c_{i+1}}, Y_{c_i} - Y_{c_{i+1}}) \\ distance(c_i, c_{i+1}) &= \sqrt{(X_{c_i} - X_{c_{i+1}})^2 + (Y_{c_i} - Y_{c_{i+1}})^2}. \end{aligned} \tag{8}$$

As a result, the sum of all the vector lengths yields the total length of each scan.

$$D = \sum_{i=1}^{N} distance(c_i, c_{i+1}). \tag{9}$$

Each set has a total of N corner points. The external angle between the two vectors is calculated using interior and exterior products after each vector's length (the length of each side of the figure) has been calculated. If we have two vectors V and $\hat{V}$ that are adjacent, we can calculate their interior product using the formula:

$$V \cdot \hat{V} = (V_x \times \hat{V}_x) + (V_y \times \hat{V}_y). \tag{10}$$

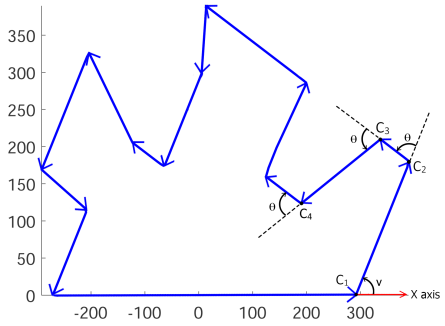
The TF algorithm then calculates the angle between two neighbor vectors as follows:

$$\theta = \arccos(V \cdot \hat{V}). \tag{11}$$

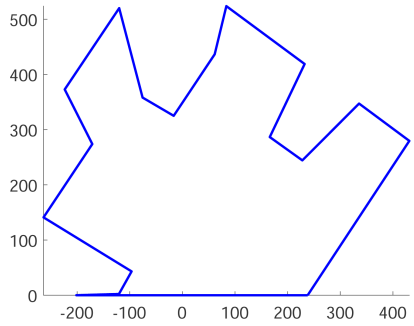
As angle signs are critical in histogram creation, we use the TF algorithm's exterior product to determine them. Because vectors are two-dimensional, the Z component ($Z = 0$) must also be included when calculating the exterior product. The exterior product's output will be a vector in the end. The angle sign is then determined by the sign of the Z component:

$$\begin{aligned} V \times \hat{V} &= (m_x, m_y, m_z) \\ m_x &= V_y \hat{V}_z - V_z \hat{V}_y \\ m_y &= V_z \hat{V}_x - V_x \hat{V}_z \\ m_z &= V_x \hat{V}_y - V_y \hat{V}_x. \end{aligned} \tag{12}$$

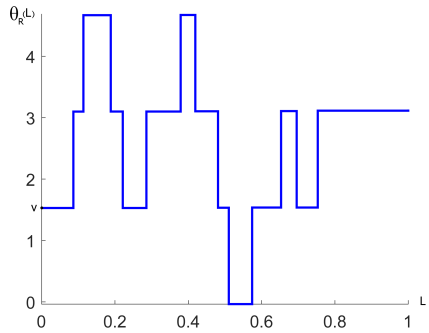
The angle sign will be negative if the Z-component sign is negative; otherwise, it will be positive. We will obtain a histogram for each scan form after calculating the length and angle of the vectors.



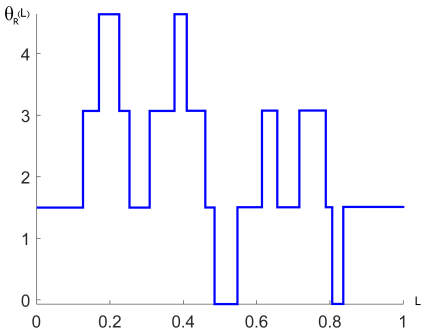
(**a**) RSD data



(**b**) VSD data



(**c**) Created histogram based on RSD



(**d**) Created histogram based on VSD

Figure 6. Histogram creation steps.

4.3. Histogram Matching

A histogram is generated for the RSD set after the length and angle of the vectors are calculated. However, we will have a histogram for each point in the RSD set because each point in the dataset will have a different graph when we begin drawing the histogram. As a result, we have an equal number of points in the C_v set of the histogram, all of which match the RSD set's histogram. To match two histograms, they must first be transferred to the same coordinate system. The area between the two graphs is then computed. Rectangles are created between two graphs because they are in the form of histograms. Each rectangle's area is determined, and the area between the two graphs is calculated by adding the total areas together. The more similar the two graphs are, the smaller the area between them becomes. Therefore, a graph from the VSD set is chosen that has a lower distance function

value than the others. The pairing of histograms made from RSD and VSD datasets is shown in Figure 7.

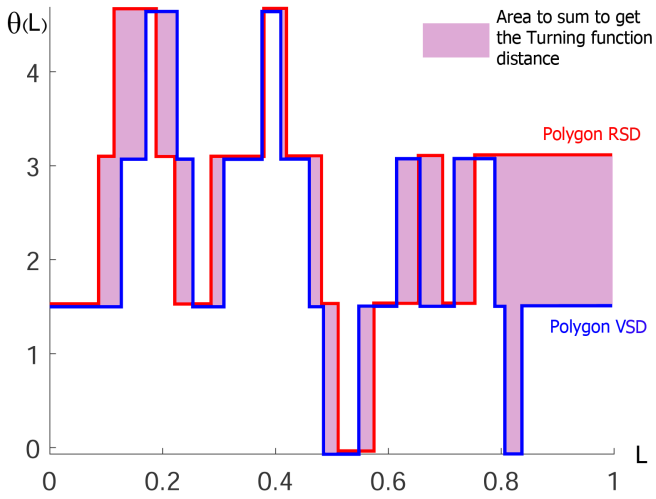


Figure 7. Histogram matching on RSD and VSD data.

4.4. Pose and Orientation Estimation

The robot localization problem is expressed as an optimization problem to determine the correct position and orientation. The simplex algorithm is proposed as a solution to this problem.

$$\begin{array}{lll} find & \delta_x, \delta_y, \delta_\theta & \\ subject & to & Err \leq Treshhold \\ where & Err = (\int_0^1 |\Theta_R(L) - \Theta_V(L)|^2 \, dL)^1/2. & \end{array} \tag{13}$$

The rotation functions of the real and virtual sensor data sets are denoted by $\Theta_R(L)$ and $\Theta_V(L)$, respectively, in this problem. The distance between these two functions is indicated by Err. The goal is to determine the difference between real and virtual sensor positions.

The three vertices of a triangle are the three candidates for the robot's expected position on a predefined peripheral map in location estimation. The algorithm returns some of the position minimization parameters of the error minimizer as a solution, described in the Algorithm 5, by shifting the direction and coordinates of the vertices of this triangle and moving them to the local or global minima. The expected position angle of the robot on the map is assumed to be the viewing angle of each of these vertices. To improve the angle and reduce the difference between the real and virtual sensors' positions and orientations, the angle of each of these three vertices is examined locally in seven orientations to find the best angle that minimizes the difference in orientation and position. Algorithm 5 describes the histogram matching process used to calculate the best orientation and least error from the RSD and VSD sets for the seven virtual space coordinates (area between two graphs). Three points are chosen at random in an arbitrary interval in the beginning. The maximum value of the robot position difference should be covered by this interval. The δ_x, δ_y, and δ_θ values are thus appropriately approximated each time the algorithm is run using the simplex optimization algorithm. New approximations of the robot's position in space must be considered until new values for δ_x and δ_y can be estimated. The virtual sensor is simulated for new values after the final values for δ_x, δ_y, and δ_θ have been set. This operation is iterated by the algorithm until a suitable fixation between the two positions is found.

Algorithm 5 Simplex(x_v, y_v, θ_v, D^r)

Input:

Output:

$v \leftarrow [x_v, y_v]$
Select three random points with an arbitrary x and y and call them $P1, P2, P3$
$Err_{P1} \leftarrow Function(P1, \theta_v, D^r)$ ▷ Err_{P1} is the amount of error Err after simulation for P1
$Err_{P2} \leftarrow Function1(P2, \theta_v, D^r)$
$Err_{P3} \leftarrow Function1(P3, \theta_v, D^r)$
while $(Err_{P1} > Threshold)\&(Err_{P2} > Threshold)\&(Err_{P3} > Threshold)$ **do**
    $B \leftarrow$find the point among P1, P2, and P3 with minimum error
    $NB \leftarrow$find the point among P1, P2, and P3 with 2nd minimum error
    $W \leftarrow$find the point among P1, P2, and P3 with 2nd maximum error
    $CEN \leftarrow [(B_x + NB_x)/2, (B_y + NB_y)/2]$
    $R \leftarrow [(CEN_x + (CEN_x - W_x)), (CEN_y + (CEN_y - W_y))]$
    $Err_B \leftarrow Function1(B, \theta_v, D^r)$
    $Err_{NB} \leftarrow Function1(NB, \theta_v, D^r)$
    $Err_W \leftarrow Function1(W, \theta_v, D^r)$
    $Err_R \leftarrow Function1(R, \theta_v, D^r)$
    if $Err_R < Err_B$ **then**
        $E \leftarrow [(R_x + (R_x - CEN_x)), (R_y + (R_y - CEN_y))]$
        $Err_E \leftarrow Function1(E, \theta_v, D^r)$
        if $Err_R < Err_E$ **then**
            $W \leftarrow R$
        else
            $W \leftarrow E$
        end if
    else
        if $Err_R \geq Err_B \& Err_R < Err_{NB}$ **then**
            $W \leftarrow R$
        else
            if $Err_R \geq Err_{NB} \& Err_R < Err_W$ **then**
                $C_R \leftarrow [(CEN_x + R_x)/2, (CEN_y + R_y)/2]$
                $Err_{C_R} \leftarrow Function1(C_R, \theta_v, D^r)$
                $W \leftarrow C_R$
            else
                $C_W \leftarrow [(CEN_x + W_x)/2, (CEN_y + W_y)/2]$
                $Err_{C_W} \leftarrow Function1(C_W, \theta_v, D^r)$
                $W \leftarrow C_W$
            end if
        end if
        $Err_{P1} \leftarrow Function1(B, \theta_v, D^r)$
        $Err_{P2} \leftarrow Function1(NB, \theta_v, D^r)$
        $Err_{P3} \leftarrow Function1(W, \theta_v, D^r)$
    end if
end while
$B \leftarrow$ find the point among B, NB and W with minimum error
$x_v \leftarrow B_x, y_v \leftarrow B_y$

5. Results and Discussion

This section evaluates the performance of the proposed algorithm. The current paper compares the performance of its turning function algorithm in three different positioning scenarios: (I) the effect of environmental details on the position and orientation of the robot, (II) the effect of angle difference on the position and orientation of the robot, and (III) the effect of the distance difference on the position and orientation of the robot. In this experiment, the environment map used in Figure 1 is made up of 1620 data points collected from various observations with various robot viewing angles. The SICKLMS200-30106

laser distance scans with a range of 10 m, an angular resolution of 1 degree, and a domain of view of 180 degrees were used to gather all of these observations. The data in this set were divided into three scenarios based on the influence of environmental factors, distance, and angle to evaluate the algorithm's performance. The difference in the position and orientation of these data ranges from 0 to 300 cm and 0 to 150 degrees, respectively.

5.1. First Scenario

Table 2 shows how environmental details affect the robot's position and orientation improvement process. We attempt to measure the effect of the incremental process of the obstacles in the simulated environment of the proposed algorithm in this experiment. One of the main challenges for our proposed algorithm is the limited number of obstacles in the environment, as it relies on robot observations (privacy). As a result, our algorithm appears to be very efficient in such situations based on the results. Since the robot is too close to the obstacles, finding the right position and orientation becomes more difficult as the number of obstacles increases. Due to the fact that the robot may not always be able to see the correct features and may find itself in situations where obstacles block its vision, or where the same features of the obstacles cause the histogram produced by the turning function to overlap, our proposed algorithm has demonstrated acceptable performance in crowded environments.

Table 2. Effect of environmental details on improving the position and orientation of the robot.

Scenario	Mean Distance Error	Mean Orientation Error	Distance Improvement	Orientation Improvement
1	7.7878	1.0526	94.8979	82.8125
2	7.9858	0.6315	64.3136	90.6250
3	9.3984	0.8421	93.0935	88.2353

5.2. Second Scenario

In the second scenario, we attempt to figure out how the angle affects the robot's observations. The effect of angle difference on the process of improving the robot's position and orientation is shown in Table 3. Based on the angle difference between real and virtual sensors, the datasets are divided into three classes, as shown in Table 2: (1) data with an angle difference between the two sets of RSD and VSD located in the range of 0 to 5 degrees, (2) located in the range of 5 to 10 degrees, and (3) located in the range of 10 to 15 degrees. Each category contains 540 data points. The distance difference in all three sections ranges from 0 to 300 cm. The greater the angle difference, the more obvious the differences between real and virtual robot observations. As a result of the rotation function, the distance difference between the two histograms will increase. There are sometimes large and unusual differences between the two sets of RSD and VSD, causing both histograms to deform fundamentally. The algorithm's performance will undoubtedly be harmed in this case. It should be noted that, as with any other algorithm, some histograms may have multiple matching candidates. Because our proposed algorithm is based on observations, there may be points that the rotation function can view and process, resulting in a histogram that looks very similar to the RSD set's histogram. In this case, our proposed algorithm will incorrectly interpret that point as the robot's true coordinates. The algorithm comes to a halt at a local optimum in this situation. The greater the angle difference, the greater the angle error, as shown by the results of this experiment.

Table 3. Effect of angle on improving the position and orientation of the robot.

Scenario	Sample	Mean Distance Error	Mean Orientation Error	Distance Improvement	Orientation Improvement
1	540	8.5752	0.4368	94.2968	77.4194
2	540	6.5290	0.7371	95.3235	86.2746
3	540	10.0688	1.0536	92.7747	90.5661

5.3. Third Scenario

The effect of the distance between the real and virtual sensors is investigated in the third scenario. Table 4 represents the results in this scenario. Based on the distance difference between the RSD and VSD data sets, the data were divided into three categories in this section: (1) data with a distance difference between 0 and 100 cm, (2) data with a distance difference between 100 and 200 cm, and (3) data with a distance difference between 200 to 300 cm. The angle difference in this scenario ranges from 0 to 15 degrees. Our proposed algorithm performs better in reducing angle error when the position difference between RSD and VSD is small, as can be seen. The virtual and real sensors' positions are more similar to the details observed in their surroundings the closer their positions are. The robot's histogram will be different than RSD in this case because of the angle change. As a result, our proposed algorithm can better orient itself at shorter distances. The distance error grows in proportion to the difference in position between the RSD and VSD. However, given the initial position difference, this increase is negligible, and due to the distance improvement rate, our proposed algorithm has been successful over long distances. Figures 8 and 9 represent the results in these scenarios. In both figures red, blue, and green lines represent first, second, third scenarios respectively.

Table 4. Effect of distance on improving the position and orientation of the robot.

Scenario	Sample	Mean Distance Error	Mean Orientation Error	Distance Improvement	Orientation Improvement
1	540	6.0639	0.4210	89.8533	93.7500
2	540	8.6486	1.0538	93.9186	84.6154
3	540	10.4595	1.0526	95.5024	85.5073

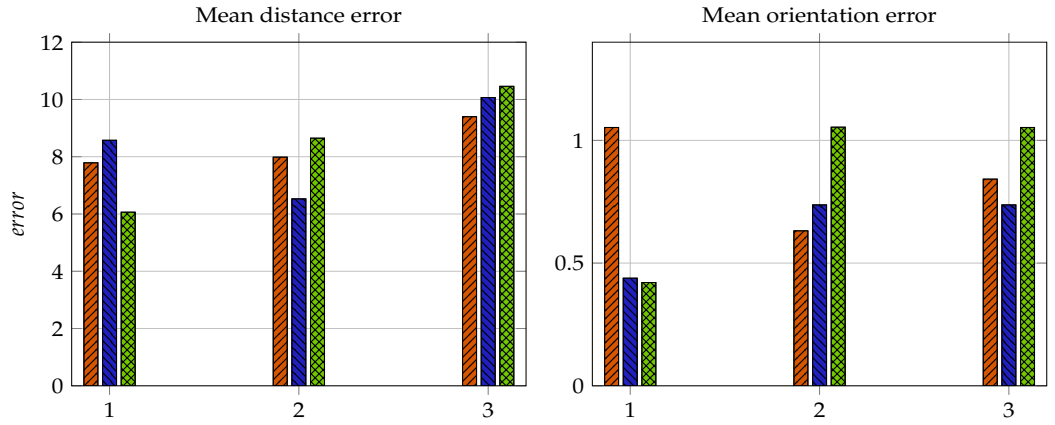


Figure 8. Mean distance and orientation error in different scenarios.

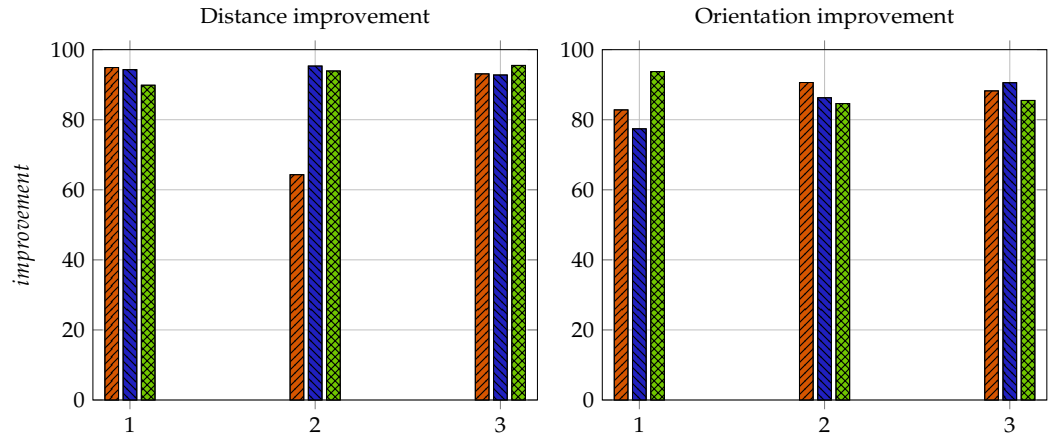


Figure 9. Distance and Orientation improvement in different scenarios.

6. Conclusions

In this article, a new matching algorithm for estimating a robot's position with the ability to correct the error in the orientation and position of a moving robot is presented. In the presented algorithm, first, a spatial description of the expected position of the robot is simulated on a complete and accurate map of the environment, and then the simulated model is adapted to a spatial description obtained from the laser rangefinder sensor data. The proposed *TF* algorithm first receives the polygons extracted from the data received from both sensors and then assigns a histogram to each sensor scan and tries to transform the pair of histograms into a single coordinate system. In this research, a simulation has been carried out to evaluate the performance of the *TF* algorithm. The simulation results show that our proposed algorithm is able to perform well in quiet, crowded environments and in different conditions due to the use of a suitable matching strategy. Other features of this algorithm include its low computational complexity and high speed and the simplicity of its implementation. Our future work in this field will involve determining the appropriate time step to estimate the position of the moving robot using the *TF* algorithm. If we run the rotation function algorithm after every hundredth of a second, we can calculate the exact position of the robot. Although we will have an accurate position of the robot, we will bear a significant calculation burden. If we run the *TF* algorithm after traveling a long distance and time step, the position of the robot may be estimated incorrectly, which causes the error to accumulate and the robot to wander in the environment. Therefore, one of our future tasks will be finding the best time to estimate the position of a moving robot using the rotation function.

Author Contributions: Conceptualization, B.S.B.; Methodology, B.S.B. and E.Z.; Software, E.Z.; Validation, M.Z.-S.; Formal analysis, S.K.; Resources, M.Z.-S.; Writing—original draft, O.A.; Supervision, B.S.B.; Funding acquisition, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The corresponding author can provide access to the source code and data used in this study upon request.

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

References

1. Islam, M.J.; Mo, J.; Sattar, J. Robot-to-robot relative pose estimation using humans as markers. *Auton. Robot.* **2021**, *45*, 579–593. [CrossRef]
2. Zhang, C.; Yang, Z.; Liao, L.; You, Y.; Sui, Y.; Zhu, T. RPEOD: A Real-Time Pose Estimation and Object Detection System for Aerial Robot Target Tracking. *Machines* **2022**, *10*, 181. [CrossRef]
3. Shamsfakhr, F.; Sadeghi Bigham, B.; Mohammadi, A. Indoor mobile robot localization in dynamic and cluttered environments using artificial landmarks. *Eng. Comput.* **2019**, *36*, 400–419. [CrossRef]
4. Kozák, V.; Sushkov, R.; Kulich, M.; Přeučil, L. Data-Driven Object Pose Estimation in a Practical Bin-Picking Application. *Sensors* **2021**, *21*, 6093. [CrossRef] [PubMed]
5. Liao, B.; Liu, W. Pseudoinverse-type bi-criteria minimization scheme for redundancy resolution of robot manipulators. *Robotica* **2015**, *33*, 2100–2113. [CrossRef]
6. Jin, L.; Li, S.; Xiao, L.; Lu, R.; Liao, B. Cooperative motion generation in a distributed network of redundant robot manipulators with noises. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *48*, 1715–1724. [CrossRef]
7. Xiaowei, Z.; Khing, H.Y.; Seng, C.C.; Yi, Z. The localization of mobile robot based on laser scanner. In Proceedings of the 2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No. 00TH8492), Halifax, NS, Canada, 7–10 May 2000; Volume 2, pp. 841–845.
8. Banerji, D.; Ray, R.; Basu, J.; Basak, I. Autonomous navigation by robust scan matching technique. *arXiv* **2012**, arXiv:1212.1313.
9. Park, S.; Park, S.K. Global localization for mobile robots using reference scan matching. *Int. J. Control Autom. Syst.* **2014**, *12*, 156–168. [CrossRef]
10. Shamsfakhr, F.; Bigham, B.S. GSR: Geometrical scan registration algorithm for robust and fast robot pose estimation. *Assem. Autom.* **2020**, *40*, 801–817. [CrossRef]
11. Friedman, C.; Chopra, I.; Rand, O. Perimeter-based polar scan matching (PB-PSM) for 2D laser odometry. *J. Intell. Robot. Syst.* **2015**, *80*, 231–254. [CrossRef]

12. Furukawa, T.; Dantanarayana, L.; Ziglar, J.; Ranasinghe, R.; Dissanayake, G. Fast global scan matching for high-speed vehicle navigation. In Proceedings of the 2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), San Diego, CA, USA, 14–16 September 2015 ; pp. 37–42.
13. Li, Z.; Liao, B.; Xu, F.; Guo, D. A new repetitive motion planning scheme with noise suppression capability for redundant robot manipulators. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *50*, 5244–5254. [CrossRef]
14. Pozna, C.; Precup, R.E.; Földesi, P. A novel pose estimation algorithm for robotic navigation. *Robot. Auton. Syst.* **2015**, *63*, 10–21. [CrossRef]
15. Padkan, N.; Sadeghi Bigham, B.; Faraji, M.R. Fingerprint matching using the onion peeling approach and turning function. *Gene Expr. Patterns* **2023**, *47*, 119299. [CrossRef] [PubMed]
16. Wolin, A.; Eoff, B.; Hammond, T. ShortStraw: A Simple and Effective Corner Finder for Polylines. *SBIM* **2008**, *8*, 33–40.