GSR: Geometrical Scan Registration algorithm for robust and fast robot pose estimation

Farhad Shamsfakhr · Bahram Sadeghi Bigham*

Received: date / Accepted: date

Abstract Undoubtedly robot localization in indoors environment where environmental information is obtained by laser rangefinders, which provide partial and simple information about surrounding of robots, is a challenging task. In this paper, an attempt has been made to develop an algorithm equipped with geometric pattern registration techniques to perform exact, robust, and fast robot localization purely based on laser range data. The expected pose of the robot on a pre-calculated map is in the form of simulated sensor readings. In order to obtain the exact pose of the robot, segmentation of both real laser range and simulated laser range readings is performed. Critical points on two scan sets are extracted from the segmented range data, and thereby the pose difference is computed by matching similar parts of the scans and calculating the relative translation. In contrast to other self-localization algorithms based on particle filters and scan matching, our proposed method, in common positioning scenarios, provides a linear cost with respect to the number of sensor particles, making it applicable to real-time resource-limited embedded robots. The proposed method is able to obtain a sensibly accurate estimate of the relative pose of the robot even in non-occluded, but partially visible segments conditions. A comparison of State-of-the-Art localization techniques has shown that GSR algorithm is superior to the other localization methods based on scan matching in accuracy, processing speed, and robustness to large positioning errors. Effectiveness of the proposed method has been demonstrated by conducting a series of simulations.

Keywords Autonomous Robots · Robot Pose Estimation · Laser Range Data · Corner Detection · Feature Matching · Scan Registration

1 Introduction

Autonomous navigation of a mobile robot is generally defined as control of robot motion to arrive at a given position in its environment without human intervention. This navigation task can be decomposed into the various aspects such as robot pose estimation (localization) and path planning. To generate a path from an initial position to a target position, the localization is vital which frequently provides and updates a reliable estimate of the position and orientation of the robot given a global coordinate system in a structured or semi-structured environment. Maintaining an estimate of the subsequent location of a robot using odometry, while the position and orientation of the robot is known at any time, is a common solution. But this method is sensitive to errors and the outcome is not reliable because of the integration of errors over time. Thus, the need arises for developing an algorithm to overcome the challenge. Many approaches have been proposed in the literature [1–5] to address the self-localization problem. One possible strategy to tackle the problem of pose estimation is to use scan matching, in which the geometric
representation of the current scan is frequently compa-
ted to a reference scan until an optimal geometric overlap with the reference scan is obtained. A robust and accurate localization approach using Kalman filtering, which is based on scan matching is discussed by [6]. The earliest efficient methods based on scan matching, which try to match sensor readings to the line segments extracted from set of scanned points, and then reconstruct the map at the same time, are proposed by [7,8]. Another scan matching method is proposed by [9], in which at first a histogram is assigned to each of the scans and then a histogram registration algorithm transforms (shift and rotate) the pair of histograms into one coordinate system. A point-to-point scan matching algorithm which takes two scans as an input and then using a heuristic algorithm tries to compute the transformation between the two scans has been presented by [10]. An efficient multisensory approach which uses the geometric features from a laser rangefinder and a CCD camera to pose estimation is proposed by [11]. The method has proven to be successful in environments with more uncertainty, where a laser rangefinder, by itself, cannot visualize all the peripheral edges facing the robot (e.g. long corridors). An algorithm for self-localization in polygonal environments is suggested by [12]. The method is shown to be efficient, fast, and flexible in dynamic and polygonal small environments such as robotic soccer. [13] use the principal components of range data sets to form the structure of robot environment, and then tries to infer the most probable pose of robot in the Eigen space of the range data sets. A particle filtering pose estimation algorithm is proposed by [14]. Although the method is accurate, a set of assumptions and limitations have to be defined to make it work well. For example, when the robot has a high pose error, and where the robot environment is partly cluttered the method is useless.

The most famous position estimation algorithm based on Bayesian estimation that works through a non-Gaussian distribution of particles in a previously mapped environment is Monte Carlo Localization (MCL) [15–17]. Each sample (particle) is weighted using a reference map represented by vectors, and when the robot is confronted with the changes in the environment, the set of weighted samples are updated. This method highly relies on particles sampling. A large number of samples are required to achieve the desired result; Otherwise the result becomes inaccurate. Hence, to achieve an appropriate degree of accuracy this method requires a high computational resource.

One of the most popular methods for dealing with the pose estimation problem is based on the study of techniques which predominantly try to use exterocep-
tive sensors’ readings to localize the robot in a map and to obtain a representation of the map at the same time; like the known Simultaneous Localization and Mapping problem (SLAM) [18,19]. Another self-localization strategies which use Kalman filters to perform sensor fusion are introduced by [20,21]. Pose estimation problem can also be addressed by frequently recognizing landmarks, in which in order to determine the probable position of the landmarks, robot observation, in the form of laser range reading, continually is matched with the features of the landmarks. A position estimation algorithm based on recognition of landmarks is proposed by [22]. Probabilistic approaches such as Bayesian methods [23–25], and Markov localization [26–30] are another major strategies to solve robot localization problem.

It turned out that all existing positioning strategies do not meet the key property of robustness, accuracy, and being fast. For this reason, a new method, which is able to correct large position and orientation errors of a robot in a liner time complexity has been developed. According to the reviews of literatures, scan matching, which is concerned with matching sensed data against map information, is an obvious choice in dealing with the self-localization problems. In efficient model-based approaches equipped with scan matching algorithms, by computing the best similarity between the robot observation, and an accurately-known or reconstructed map (i.e. the model) of the environment, it is possible to obtain a sensibly accurate estimate of the relative position and orientation of robot. Hence, in our proposed algorithm, firstly, a spatial description of the expected pose of the robot on a totally known or reconstructed environmental map is simulated, and then the simulated model is matched to the spatial description from laser range data. The approach proposed in this study is inspired from scan matching approaches [31–33] and particle filter approaches [14,34,35].

Various types of sensors such as laser rangefinders, sonars, Kinect, and vision can be used to address the self-localization problem in indoors environments. Although laser rangefinder suffers from poor calibration, but in the study it would be preferred, since it is much better than other types of exteroceptive sensors in case of speed of distance traveled and angular resolution, also provides an accurate data, which can be interpreted faster, and with much less computational effort than a vision system. The rest of the paper is organized as follows: In Section 2 the proposed method has been briefly described. The mathematical statement of the problem, also a general overview of our algorithm are provided in Section 3. In Section 4, the proposed method, including five different stages, as well as complexity analysis of the
algorithm are explained. In Section 5, the experimental results of our algorithm are presented. In Section 6, a comparison between the proposed method and some state-of-the-art localization methods is performed, and finally our conclusions are presented in Section 7.

2 Algorithm Description

This paper presents a new scan matching method to yield a robust and fast pose estimation algorithm. The GSR algorithm proposed in this paper takes two visualizations extracted from 2D laser range data, namely RSD (real spatial description), and VSD (virtual spatial description), and then tries to maximize the similarity between the two visualizations by transforming them (shift and rotate) into one coordinate system, and in this way, calculates actual difference between the two poses. VSD is a simulated visualization of the expected pose of the robot on a pre-calculated environmental map or, say, a simulation of sensor particles, and RSD is a visualization of the real pose of the robot from laser range data. The GSR algorithm proposed in this paper assumes that the environmental map is known at any time. Figure 1 illustrates the concept of virtual sensor and real robot on a known-map.

3 Problem statement

Figure 2 shows the configuration of a mobile robot in a two-dimensional Cartesian space. In order to specify the position of the robot, we select a point $O$, denoted by $x_r, y_r$, on the rigid body of the robot as the position reference point. The angle between the line which passes through this point and the positive x-axis specifies the robot orientation $\theta_r$. In the same way as real robot, the pose of the virtual sensor is defined. Hence, the virtual sensor and the real robot poses illustrated in Figure 2 are defined as:

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

$\widehat{T}$, the estimation of the relative translation (i.e. $\widehat{T} = RP - VP$), which holds the values of the pose difference is defined as:

$$\widehat{T} = \widehat{RP} - VP = \begin{pmatrix} \Delta x = \sum_{i=1}^{I} \delta x_i \\ \Delta y = \sum_{j=1}^{J} \delta y_j \\ \Delta \theta = \sum_{k=1}^{K} \delta \theta_k \end{pmatrix}$$  \tag{2}

where $\delta x_i$ and $\delta y_j$ are the two position difference components at times $i$ and $j$, respectively, and $\delta \theta_k$ is the orientation difference component at time $k$.

The type of the sensor which is used for both virtual sensor and robot sensor is the same (i.e. Sick laser scanner LMS200-30106), and is defined as $q(m, f, q)$, where $m$ represents the maximum range, $f$ is the field of view, and $q$ is the angular resolution of the sensor.

$D^R = [d^R_1, d^R_2, \ldots, d^R_n]$ is the set of distances measured by the robot sensor, $D^V = [d^V_1, d^V_2, \ldots, d^V_n]$ is the set of distances measured by the virtual sensor, and $\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$ is the set of beams angles, where $n = (f/q) + 1$ is the number of distances measured by the sensor, and beam angle $\phi_i$ of distance $d_i$ is calculated as follows:

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

Each point of the relative visualizations, RSD and VSD, is obtained by the following linear transformation:

$$\begin{align*}
RSD_i &= (\cos(\phi_i) \times d^i_r, \sin(\phi_i) \times d^i_r) \\
VSD_j &= (\cos(\phi_j) \times d^j_v, \sin(\phi_j) \times d^j_v).
\end{align*}$$  \tag{4}

Figure 3 depicts RSD and VSD for the robot sensor and the virtual sensor illustrated in Figure 2. RSD and VSD are used in next stages to perform scans registration calculations.

Given the $D^R$, $D^V$, and $VP$, the problem to be solved is to determine $\widehat{T}$ so that the final components of the pose difference, $\delta x, \delta y, \delta \theta$ are below some threshold. In other words, the robot pose is estimated as:

$$\rho P = \{VP + \widehat{T}|\delta x, \delta y, \delta \theta \leq \epsilon\}.$$  \tag{5}

Therefore, the three components of the pose estimation error are defined as follows:

$$\begin{align*}
\delta x &= |x_r - \bar{x_r}| \\
\delta y &= |y_r - \bar{y_r}| \\
\delta \theta &= angdiff(\theta_r, \hat{\theta}_r)
\end{align*}$$  \tag{6}

where $angdiff(a, b)$ returns the difference between the angles $a$ and $b$ so that the result is wrapped on the interval $[-\pi, \pi]$.

4 GSR algorithm

Figure 4 shows an overview of the proposed algorithm. The pose estimation algorithm proposed in this paper which determine an exact value of $\widehat{T}$ runs in an iterative process in four different stages: corners detection, corners matching, calculating orientation difference and position difference. Once the corners of both scans are detected, the registration cycle starts by matching corner points on the two sets and finding similar parts of the scans, and thereby computing the orientation difference $\delta \theta$. The virtual sensor is then simulated for the
In this study, the registration is carried out using a geometrical polyline matching method. Corner points as the most widely used critical points in shape matching problems are important features that provide remarkable information about the global structure of a shape. Corner points are points, at which the angle between the two intersecting line segments is above/below a threshold. Thus, the problem to be solved in this stage is to find all corner points of \(RSD\) and \(VSD\).

4.1 Corners detection

The first section of GSR algorithm is to detect and extract all corners from the shape of the scans. A significant number of algorithms are developed to perform polyline corner finding [36–39], but all the algorithms rely on more advanced statistical and mathematical formulation which makes them computationally intensive. In this paper ISS algorithm (Improved ShortStraw), a slightly modified version of the known ShortStraw algorithm [40] which is a robust, fast and highly accurate polyline corner finder founded on a simple concept is introduced. The first step in ShortStraw algorithm is to resample the points of a shape. The goal of resampling is to redrawing original points of a polyline so that the new points are placed equidistant from each other as much as possible and in this way can fully describe the shape. The resampled \(RSD\) and \(VSD\) shown in Figure 3 are illustrated in Figure 5.

The strength of ShortStraw is greatly dependent on resampling stage. In our ISS algorithm, a new method for point resampling is proposed, which has improved the effectiveness of the ShortStraw algorithm for using in scan matching. This method ensures that all the resampled points of the shape of a scan are placed based on an absolutely equal distance with a constant length from each other. This constant length is named interspacing distance \(S\), and for our positioning problem is set to 1. The main idea in the resampling procedure is illustrated in Algorithm 2. After resampling the relative visualizations, ShortStraw detects corners using length of the straws. A straw for a resampled point at \(p_i\) is calculated as:

\[
\text{straw}_i = |P_{i-\omega}, P_{i+\omega}|
\]

where \(\omega\) is a constant window, and in proposed ISS algorithm we set \(\omega = 1\). All the straws are first computed...
Algorithm 1 GSR\((D^r,D^v,VP)\)

1:  \(i \leftarrow 1\)
2:  \(j \leftarrow 1\)
3:  \(k \leftarrow 1\)
4: repeat
5:  \([RSD,VSD] \leftarrow \text{Visualization}(D^r,D^v,VP)\)
6:  \(\text{CornersSet} \leftarrow \text{CornersDetection}(RSD,VSD)\)
7:  \(\delta \theta \leftarrow \text{CalculatingOrientationDifference} (\text{MatchedCorners})\)
8:  if \(\delta \theta > \epsilon \theta\) then
9:     \(\theta_v \leftarrow \theta_v + \delta \theta;\)
10:    \(k \leftarrow k + 1;\)
11: Repeat the lines 5-7;
12: \([\delta x_i,\delta y_j] \leftarrow \text{CalculatingPositionDifference}(\text{MatchedCorners})\)
13: if \(\delta x_i > \epsilon x\) then
14:     \(x_v \leftarrow x_v + \delta x_i;\)
15:     \(i \leftarrow i + 1;\)
16: if \(\delta y_j > \epsilon y\) then
17:     \(y_v \leftarrow y_v + \delta y_j;\)
18:     \(j \leftarrow j + 1;\)
19: until \((\delta x_i \leq \epsilon x) \land (\delta y_j \leq \epsilon y) \land (\delta \theta_k \leq \epsilon \theta)\)

for points \(P_{\omega+1}\) to \(P_{N-\omega}\), where \(N\) is the number of resampled points. As we reach to a curvature in the shape, the length of the straws will begin to shorten. Thus, corner points are points at which the length of the associated straws are below a threshold. Algorithm 3 illustrates the process of finding corners set of resampled \(RSD\) and \(VSD\). The quality of the process of finding corners is dependent on the value of the threshold constant factor \(Z\). In addition to \(Z\), the value of the interspacing distance \(S\) affects the strength of the shortstraw. For example, the less the value of \(S\) is, the more accurate visualization is gained, but smaller values of \(S\) cause too much noise, whereas greater values of \(S\) create oversmoothed strokes. However, this is simply possible to experimentally determine appropriate values of \(S\) and \(Z\). In our experiment by setting the values of \(S\) and \(Z\) to 0.99 and 1, respectively, the best possible trade-off has been achieved. Figure 6 shows an example of correctly segmented strokes of resampled \(RSD\) and \(VSD\) by the proposed ISS algorithm.

4.2 Similarity Measures

After the set of corners for both \(RSD\) and \(VSD\) are computed in the previews stage, some processes are used to obtain a good matching between the two sets. Let \(C^R = [c^R_1,c^R_2,\ldots,c^R_n]\) be the set of corner points of \(RSD\), and \(C^V = [c^V_1,c^V_2,\ldots,c^V_m]\) be the set of corner points of \(VSD\), both in a counterclockwise direction. \(CC^R = [cc^R_1,cc^R_2,\ldots,cc^R_n]\), and \(CC^V = [cc^V_1,cc^V_2,\ldots,cc^V_m]\) are then defined to be the sets of corner coordinates from \(RSD\) and \(VSD\), respectively. Before going into the mathematical description of the algorithm, follow-
Algorithm 2 Point resampling(points)

1: \( i \leftarrow 2 \)
2: \( S \leftarrow 1 \)
3: \( P_1 \leftarrow points_{i-1} \)
4: \( \text{while } i < \text{length(points)} \text{ do} \)
5: \( P_2 \leftarrow points_i \)
6: \( D \leftarrow \|P_1 - P_2\|^2 \)
7: \( \text{if } D > S \text{ then} \)
8: \( P_{3x} = P_{2x} + (P_{2x} - P_{1x}) / D \times S \)
9: \( P_{3y} = P_{2y} + (P_{2y} - P_{1y}) / D \times S \)
\( \) \( P_3 \) is located exactly \( S \) Euclidean distance away from the \( P_1 \) in the slope direction of the straight line from \( P_{i-1} \) to \( P_i \).
10: \( \text{Append } P_3 \text{ to the resampled array} \)
11: \( P_3 = P_3 \)
12: \( \text{else} \)
13: \( i \leftarrow i + 1 \)

Algorithm 3 Corner detection(points)

1: \( corners \leftarrow 1 \)
2: \( straws \leftarrow \{ \} \)
3: \( Z \leftarrow 0.99 \)
4: \( \omega \leftarrow 1 \)
5: \( \text{for } i \leftarrow \omega + 1, \text{number of points} - \omega \text{ do} \)
6: \( straws(i - \omega) \leftarrow \|\text{points}(i - \omega) - \text{points}(i + \omega)\|^2 \)
7: \( \text{threshold} \leftarrow \text{median}(straws) \times Z \)
8: \( \text{for } i \leftarrow \omega + 1, \text{number of points} - \omega \text{ do} \)
9: \( \text{if } straws(i - \omega) < \text{threshold} \text{ then Append(corners, } i) \)
\( \) \( \text{Append(corners, number of points) } \)

and VSD, respectively. Figure 7 illustrates the concept of CA and edge. As shown in the figure, each \( ca_i \) has two edges as follows:

\[
ca_i^r = \begin{pmatrix} E1_{i}^r \\ E2_{i}^r \end{pmatrix}, \quad ca_i^v = \begin{pmatrix} E1_{i}^v \\ E2_{i}^v \end{pmatrix} \tag{8}
\]

where

\[
\begin{align*}
E1_{i}^r &= (cc_{i}^r - (cc_{i+1}^r)) \), \quad E2_{i}^r = (cc_{i+2}^r - (cc_{i+1}^r)) \), \\
E1_{i}^v &= (cc_{i}^v - (cc_{i+1}^v)) \), \quad E2_{i}^v = (cc_{i+2}^v - (cc_{i+1}^v)) \).
\end{align*} \tag{9}
\]

The task to do in this stage is to detect all correct correspondences between the sets of CAs of RSD and VSD using a similarity function. The method is to take each CA from RSD set and compare it with each of the CAs in the VSD set. Each comparison yields a score, a similarity measure. The best similarities are assigned to the pair of CAs with the highest similarity score. From the correspondences between the similar CAs of RSD and VSD, the set of matched CAs are obtained, and then using a robust matching technique an estimate of the difference between the two poses is archived. Therefore, given the sets of CAs and corners, the similarity matrix is defined as:

\[
\text{SimilarityMatrix}(i,j) = \frac{f(i,j) \times g(i,j) \times s(i,j)}{h(i,j)}.
\]

Fig. 4 A brief overview of GSR algorithm.
Fig. 5 Resampled points of: (a) VSD, and (b) RSD.

Fig. 6 Correctly detected corners of: (a) VSD, and (b) RSD, which are marked with red points.

Fig. 7 An example of a CA and its edges.

The function $f$ measures the length similarity of $RSD$ edges and $VSD$ edges:

$$f(i,j) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{E_{l_{i}}^r - E_{l_{j}}^v}{\sigma_i} \right)^2} + \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{E_{l_{i+1}}^r - E_{l_{j+1}}^v}{\sigma_i} \right)^2}.$$  \(11\)

where $E_{l_{i}}^r (i = 1, 2, 3, \ldots, n)$ and $E_{l_{j}}^v (j = 1, 2, 3, \ldots, m)$ are the length values of edges of $RSD$ and $VSD$, respectively. It must be noted that we do not use the Euclidean distance between the end points of an edge to compute its length, but rather simply calculate the difference between the number of the two end points (e.g. $E_{l_{i}}^r = c_{i+1}^r - c_i^r$). Using Euclidean distance, in addition to increase the algorithm’s computation time to perform the square root calculations, leads to impossibility of measuring the length of curved edges.

(10)
$g$ is the similarity measure between interior angles of two different CAs in RSD and VSD, and it is formulated as:

$$g(i,j) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{\varphi_i^r - \varphi_j^r}{\sigma_i} \right)^2}. \tag{12}$$

where

$$\varphi_i^r = \cos^{-1}\left( \frac{\|E1_i^r \times E2_i^r\|}{\|E1_i^r\| \|E2_i^r\|} \right),$$

$$\varphi_j^v = \cos^{-1}\left( \frac{\|E1_j^v \times E2_j^v\|}{\|E1_j^v\| \|E2_j^v\|} \right). \tag{13}$$

$S$ is the attitude similarity function which computes the angle between an edge in $ca_i^r$ and its corresponding edge in $ca_j^v$, and in this way measures the difference between the orientation or angular position of the two CAs. Figure 9 illustrates the concepts of the interior angle and the attitude angle. The function $S$ is formulated as:

$$s(i,j) = \left( \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{\rho_{i,j}^a - \rho_{i,j}^b}{\sigma_i} \right)^2} \right). \tag{14}$$

where

$$\rho_{i,j}^a = \cos^{-1}\left( \frac{\|E1_i^r \times E1_j^v\|}{\|E1_i^r\| \|E1_j^v\|} \right),$$

$$\rho_{i,j}^b = \cos^{-1}\left( \frac{\|E2_i^r \times E2_j^v\|}{\|E2_i^r\| \|E2_j^v\|} \right). \tag{15}$$

$h(i,j)$ is the position similarity function which simply specifies the horizontal area (the area along the x-axis) occupied by $ca_i^r$ and $ca_j^v$, and can be computed as:

$$h(i,j) = (|\alpha_{i,1} - \beta_{i,1}| + |\alpha_{i,2} - \beta_{i,2}|). \tag{16}$$

4.3 Correspondence matching

After the similarity matrix is computed in the previews stage, the similarity scores are used to match the CAs. As the number of matched points increases, the estimation of the difference between the two poses becomes more reliable. Here, in an ideal case, all the CA points of the set with fewer member should be matched with the CA points of the other set. But, as it will be further
discussed, in some cases it could be impossible. However, a lower bound must be determined for the number of CA points that should be corresponded between the RSD and VSD sets. Through experiment, we came to believe that in order to get a good enough estimate of the difference between the two poses at least three consecutive CA points of RSD must be matched with three consecutive CA points of VSD. Algorithm 4 illustrates the process of matching CAs.

Table 1 (i.e. a similarity matrix) illustrates the result of the matching procedure for RSD and VSD shown in Figure 3. In each row of the similarity matrix, the matching is assigned to the pair of CAs with the highest similarity score. The correspondence matching between the CAs of VSD and RSD is illustrated in Figure 10.

4.4 Orientation Estimation

Given the set of matching sequences $M = \{m_1, \ldots, m_N\}$, the orientation difference between the two poses is estimated as:

$$\delta_\theta = \frac{1}{N} \sum_{i=1}^{N} \text{angleDiff}(\theta_i^{Rr}, \theta_i^{Rv})$$  \hspace{1cm} (18)

where

$$\begin{cases} 
\theta_i^{Rr} = 2 \times \arctan\left(\frac{\lambda_y^c}{\sqrt{\lambda_x^c + \lambda_y^c}}\right) \\
\theta_i^{Rv} = 2 \times \arctan\left(\frac{\lambda_y^c}{\sqrt{\lambda_x^c + \lambda_y^c}}\right)
\end{cases}$$  \hspace{1cm} (19)

and $m_i$ is the subsequence $i$ from the set of matching sequences $M$ which consists of $n$ consecutive matched CA pairs ($n$ is the number of CAs of a subsequence).

$\theta_i^{Rr}$, the rotation angle of subsequence $i$ from RSD set, is the angle between the line from the first CA to the last CA of the subsequence $i$ and positive x-axis. Figure 11 illustrates the concept.

4.5 Position estimation

The problem of estimating position difference between the robot and our expectation can be considered as an optimization problem as:

$$\begin{cases} 
\text{find } \delta_x, \delta_y \in \mathbb{R} \\
\text{subject to } \text{Err} \leq \text{Treshhold} \\
\text{where } \text{Err} = \sum_{i=1}^{N} |d_i^r - d_i^v|
\end{cases}$$  \hspace{1cm} (21)

In our experiments, the simplex algorithm[41] has been utilized in order to make an estimate of relative position of the robot. Simplex algorithm is expected to converge to a local minimum in a finite number of iterations and a linear time complexity [42]. As shown in Algorithm 5, Simplex algorithm starts with three random points in a confidence interval. As it will be further illustrated by some experiments, the confidence interval for position difference in GSR algorithm for a laser range finder with 10 meters sensor range is $[-300, 300]$. Hence, using Simplex algorithm, in each iteration of GSR algorithm an appropriate estimate of the relative position of robot is obtained. While the values of $\delta_x$ and $\delta_y$ are estimated,
Algorithm 4 Correspondence matching \((RSD)\)

1: for \(i \leftarrow 1, \text{number of CAs in RSD set}\) do
2: \(\text{indx} \leftarrow \text{find index of the maximum value in } i^{th} \text{ raw of the similarity matrix}\)
3: \(\text{Append}(\text{Temp}, [i, \text{indx}])\)
4: \(M \leftarrow \{\}\) \(\triangleright \) \(M\) is the matching sequence
5: \(\text{Stack} \leftarrow \text{Temp}(1)\)
6: \(i \leftarrow 2\)

7: while \(i \leq \text{length}(\text{Temp})\) do
8: \(\text{if } (\text{Temp}(i, 2) = \text{Temp}(i - 1, 2) + 1) \text{ and } (\text{Temp}(i, 1) = \text{Temp}(i - 1, 1) + 1) \text{ then}\)
9: \(\text{Append}(\text{Stack}, (\text{Temp}(i)))\)
10: \(i \leftarrow i + 1\)
11: \(\text{else}\)
12: \(\text{if } \text{length}(\text{stack}) \geq 3 \text{ then}\)
13: \(\text{if } M \text{ is not empty} \text{ then}\)
14: \(\text{if } \text{the first value of the current Stack does not exist in } M\) then
15: \(\text{Append}(M, \text{Stack})\)
16: \(\text{if } \text{the first value of the current Stack does not exist in } M\) then
17: \(\text{Append}(M, \text{Stack})\)
18: \(\text{Stack} \leftarrow \text{Temp}(i)\)
19: \(i \leftarrow i + 1\)
20: \(\text{if } \text{length}(\text{stack}) \geq 3 \text{ then}\)
21: \(\text{Append}(M, \text{Stack})\)

---

**Table 1** Similarity matrix of the two \(RSD\) and \(VSD\) sets

| \(\text{ca}_v^1\) | \(\text{ca}_v^2\) | \(\text{ca}_v^3\) | \(\text{ca}_v^4\) | \(\text{ca}_v^5\) | \(\text{ca}_v^6\) | \(\text{ca}_v^7\) | \(\text{ca}_v^8\) | \(\text{ca}_v^9\) | \(\text{ca}_v^{10}\) | \(\text{ca}_v^{11}\) | \(\text{ca}_v^{12}\) | \(\text{ca}_v^{13}\) |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| 0.60             | 0.18             | 1.09             | 0.55             | 0.17             | 0.06             | 0.12             | 0.16             | 0.02             | 0                | 0                | 0.02             | 0.04             |
| 0.06             | 0.44             | 0.73             | 0.90             | 0.07             | 0.01             | 0.12             | 0.20             | 0.07             | 0                | 0                | 0.04             | 0.02             |
| 0.21             | 0.12             | 0.68             | 1.60             | 0.14             | 0.15             | 0.10             | 0.19             | 0.04             | 0                | 0                | 0.05             | 0.03             |
| 2.88             | 0                | 0.12             | 0.64             | 13.9             | 0                | 0.13             | 1.33             | 3.55             | 0.02             | 0                | 2.92             |
| 0                | 119               | 0.58             | 0.49             | 0                | 4.46             | 1.20             | 0.12             | 0                | 0.02             | 0.03             | 1.15             |
| 0.01             | 0.58             | 3.76             | 3.46             | 1.18             | 0.27             | 4.06             | 0.15             | 0.12             | 0                | 0                | 0.03             | 0.06             |
| 0.25             | 0.09             | 0.31             | 1.75             | 0.13             | 0.84             | 1.20             | 4.15             | 0.10             | 0                | 0                | 0.12             | 0.04             |
| 0.11             | 0                | 0.01             | 0.04             | 23.8             | 0                | 0.13             | 0.07             | 71.8             | 0.01             | 0.01             | 0                | 13.8             |
| 0.04             | 0.04             | 0                | 0                | 0.14             | 0.20             | 0                | 0                | 0.23             | 2.78             | 0.31             | 0.31             | 0.20             |
| 0.04             | 0.10             | 0                | 0                | 0.17             | 0.48             | 0                | 0                | 0.60             | 0.64             | 3.23             | 0.58             | 0.34             |

---

![Fig. 11](image) **Fig. 11** The orientation angle of: (a) \(ca_v^i\), and (b) \(ca_r^i\).

we must check whether the new estimated pose of the virtual sensor is in a free space or not. In cases the new estimated pose makes the virtual sensor locate inside an obstacle then we can simply increase/decrease the
values of $\delta_x$ and $\delta_y$ by an arbitrary amount as:

$$
\begin{align*}
\delta_x &= \{\delta_x \pm \gamma|d'_i| \neq 0, (i = 1, 2, 3, \ldots, n)\} \\
\delta_y &= \{\delta_y \pm \gamma|d'_i| \neq 0, (i = 1, 2, 3, \ldots, n)\}.
\end{align*}
$$

(22)

After the final values of $\delta_0, \delta_x$, and $\delta_y$ are determined then we use them to perform a new simulation of virtual sensor, and the algorithm is terminated when an accurate enough registration between the two scans is achieved.

4.6 Computational cost

With a quick analysis it can be shown that the proposed ISS method costs $O(\ell n / S)$ in time and space to resample the points, where $n$ is the number of distances measured by sensors, or original points of the scans, $S$ is the interspacing distance, and $\ell$ is the maximum gap between two consecutive observed points in relative visualizations. In the best-case scenario $\ell$ equals 1, and in the average-case scenario, which is the common case, the value of $\ell$ is much less than $n$. However, in the worst-case scenario, which is exceedingly unusual and virtually impossible to happen, $\ell$ could be equals the maximum distance measured by the sensors (i.e. $\ell = \text{max}(|d'_1, d'_2, \ldots, d'_n|)$). The algorithm follows with the similarity function which is $O(c^2)$, where $c$ is the number of corners. Although it is not a common input but in the worst case which each point is a corner, that is a very unlikely case, the similarity function takes $O(n^2)$ time. Since the correspondence matching procedure, in the worst case, has the same computational time as the similarity function, and the rest of the algorithm runs in $O(1)$ we can conclude that the computational complexity of the one iteration step of the algorithm is $O(\ell n / S)$. 

---

Algorithm 5 Simplex($x_v, y_v, \theta_v, D'$)

1: $v \leftarrow [x_v, y_v]$
2: Select three random points with an arbitrary $x$ and $y$ and call them $P1, P2, P3$
3: $Err_{P1} \leftarrow \text{simm}(P1 + v, \theta_v, D')$
4: $Err_{P2} \leftarrow \text{simm}(P2 + v, \theta_v, D')$
5: $Err_{P3} \leftarrow \text{simm}(P3 + v, \theta_v, D')$
6: while $(Err_{P1} > \text{Treshold}) \& (Err_{P2} > \text{Treshold}) \& (Err_{P3} > \text{Treshold})$ do
7: $B \leftarrow$ find the point among $P1, P2$ and $P3$ with minimum error
8: $NB \leftarrow$ find the point among $P1, P2$ and $P3$ with 2nd minimum error
9: $W \leftarrow$ find the point among $P1, P2$ and $P3$ with maximum error
10: $\text{CEN} \leftarrow [(B_x + NB_x)/2, \ (B_y + NB_y)/2]$
11: $R \leftarrow [(\text{CEN}_x + (\text{CEN}_x - W_x)), \ (\text{CEN}_y + (\text{CEN}_y - W_y))]$
12: $RR \leftarrow \text{simm}(B + v, \theta_v, D')$
13: $RW \leftarrow \text{simm}(W + v, \theta_v, D')$
14: $RR \leftarrow \text{simm}(R + v, \theta_v, D')$
15: if $RR < RW$ then
16: if $RR > RB$ then
17: $W \leftarrow R$
18: else
19: $E \leftarrow [(R_x + (R_x - \text{CEN}_x)), \ (R_y + (R_y - \text{CEN}_y))]$
20: $RE \leftarrow \text{simm}(E + v, \theta_v, D')$
21: if $RR < RE$ then
22: $W \leftarrow R$
23: else
24: $W \leftarrow E$
25: else
26: $C \leftarrow [(W_x + (\text{CEN}_x - W_x)/2, \ W_y + (\text{CEN}_y - W_y)/2]$
27: $W \leftarrow C$
28: $Err_{P1} \leftarrow \text{simm}(B + v, \theta_v, D')$
29: $Err_{P2} \leftarrow \text{simm}(NB + v, \theta_v, D')$
30: $Err_{P3} \leftarrow \text{simm}(W + v, \theta_v, D')$
31: $B \leftarrow$ find the point among $B, NB$ and $W$ with minimum error
32: $\delta_x \leftarrow B_x$
33: $\delta_y \leftarrow B_y$
34: $\delta_\theta \leftarrow B_\theta$
35: function $Err = \text{simm}(v_{\text{new}}, \theta_v, D')$
2: $Err \leftarrow \Sigma_{i=1}^N |d'_i - d''_i|$
3: return $Err$
GSR algorithm is $O(c \max(c/S, t/S))$ that in the worst case, in which $c$ equals $n$, is $O(n \max(n/S, t/S))$.

5 Experiments

In this section, the performance of GSR algorithm is evaluated by conducting two experiments, both carried out in MATLAB environment. In the first experiment the robustness of the matching procedure is illustrated by a series of different simulations, and in the second experiment the performance of the localization algorithm is evaluated, also the impact of outliers on the performance of the algorithm is quantified.

5.1 Experiment 1

Significant pose differences between RSD and VSD, which lead to a large deformation between the CAs of the two scans, certainly, will affect the matching performance. However, it should be noted that, like any other point matching approach, it is possible to exists several matching candidates for some CAs, also possible to exists some CAs which cannot be matched. All of these problems lead to outliers in the set of matches. This is an inevitable phenomenon, which is always likely to exist and affects the process of CA matching. In spite of the fact that complete elimination of outliers is virtually impossible, it will be further discussed in detail, and followed by some examples that even despite the existence of outliers in the set of matched CAs, when there is no large depth pose difference between RSD and VSD, the algorithm is able to correctly estimate the difference between the two poses.

Through Experiment 1, 2704 various robot observations of the environment shown in Figure 1 have been collected from 100 different spots on the map. All the scans are captured with the 2D-laser scanner SICK LMS200-30106 with 10 meters sensor range and 180 degrees field of view when the Pioneer P3-DX robot was guided through the environment. It should be noted that the average number of CAs in each scan is 14 (i.e. scatter constant-1). By running the GSR algorithm for only one iteration step, the proportion of correctly matched CAs of two RSD and VSD in different situations has been measured. In order to better evaluate the performance of the matching procedure, also optimal readability, this experiment has been divided into four different scenarios, considering 676 samples, collected by the robot sensor and virtual sensor in different situations. In all scenarios, the $x$ and $y$ position differences are modified and range from 0 to 300 centimeters. In the first scenario, the orientation difference is fixed and equals $0^\circ$. Figure 12 is the error surface for the relative correspondence error in the first scenario. It can be seen from the figure that even in situations in which the position difference is too large (e.g. $\Delta_x = 250$ and $\Delta_y = 250$), the outlier rate is significantly low. In a wide range of error surface the correspondence rate is more than 90%. The simulation result for the first scenario shows that, the minimum correct matching rate is about 45%.

The second scenario evaluates the performance of the matching procedure under the conditions in which, the orientation difference equals $15^\circ$. The correspondence error surface in the second scenario is illustrated in Figure 13. As it has almost behaved the same on previous scenario, the result shows a significantly high performance of the method. In this scenario the minimum correct matching rate is about 35%.

The robustness of the matching procedure is more evident in the third scenario where the orientation difference equals $30^\circ$. Despite the high orientation difference, the method is still able to match corners to an acceptable rate. In this scenario the minimum correct matching rate is about 30%. Figure 14 shows the result in the third scenario.

And finally the last scenario evaluates the performance of the method in conditions of large depth pose difference. In this case the orientation difference equals $45^\circ$. It can be seen from the error surface showed in Figure 15 that even in spite of such a high orientation error the average correspondence rate is relatively high which shows the efficiency of the matching procedure. In this scenario the minimum correct matching rate is about 10% which has allocated a very small region of the error surface to itself.

5.2 Experiment 2

In the second experiment the performance of GSR algorithm will be evaluated by several sets of simulations, which have been conducted under different positioning error conditions. The simulation results have been obtained by running the algorithm in four different scenarios. The number of iterations, total elapsed time and the

| scenario | Iterations | Time (ms) | $\epsilon_x$ (cm) | $\epsilon_y$ (cm) | $\epsilon_\theta$ (deg) |
|----------|------------|-----------|-------------------|-------------------|---------------------|
| 1        | 5          | 98        | 0.69              | 0.64              | 0.70                |
| 2        | 5          | 114       | 0.76              | 0.53              | 0.58                |
| 3        | 4          | 115       | 0.13              | 0.13              | 0.11                |
| 4        | 5          | 86        | 0.13              | 0.30              | 0.29                |
GSR: Geometrical Scan Registration algorithm for robust and fast robot pose estimation

$\Delta \theta = 0^\circ$

Fig. 12 Correspondence error surface in the first scenario.

$\Delta \theta = 15^\circ$

Fig. 13 Correspondence error surface in the second scenario.

$\Delta \theta = 30^\circ$

Fig. 14 Correspondence error surface in the third scenario.
$\Delta v = 45^\circ$

Fig. 15 Correspondence error surface in the fourth scenario.

Fig. 16 The first simulation scenario.

Fig. 17 The second simulation scenario.

Fig. 18 The third simulation scenario.

Fig. 19 The forth simulation scenario.
final pose estimation errors in each scenario are demonstrated in Table 2. In all scenarios of the experiment all the three pose error thresholds, $\epsilon_x$, $\epsilon_y$, and $\epsilon_\theta$ is set to 1. The first scenario evaluates the performance of GSR algorithm in conditions in which we have curved edges. The actual pose difference between the real robot and the virtual sensor in this scenario is as follows:

$$\Delta_\theta = -20^\circ, \quad \Delta_X = 100\text{cm}, \quad \Delta_Y = 140\text{cm} \quad (23)$$

The simulation results of the first scenario are illustrated in Figure 16. The algorithm has successfully corrected the pose error between the $RSD$ and $VSD$ in four iterations. In the second scenario the actual pose correctly matches $CAs$ of $RSD$ and $VSD$ after the first iteration step of running GSR algorithm. The process of pose error correction is evident from Figure 16(b). In this figure the virtual sensor readings are depicted after each iteration of running the algorithm.

The actual pose difference between the real robot and the virtual sensor in the second scenario is as follows:

$$\Delta_\theta = 20^\circ, \quad \Delta_X = 180\text{cm}, \quad \Delta_Y = 80\text{cm} \quad (24)$$

The simulation results of the second scenario are illustrated in Figure 17. GSR algorithm has corrected the pose error between the $RSD$ and $VSD$ in five iterations.

In the third scenario the actual pose difference between the real robot and the virtual sensor is as follows:

$$\Delta_\theta = 45^\circ, \quad \Delta_X = 200\text{cm}, \quad \Delta_Y = 50\text{cm} \quad (25)$$

The simulation results of the third scenario are illustrated in Figure 18. In spite of a large pose difference between the $RSD$ and $VSD$ sets, after the first iteration step of running the algorithm, it correctly matches six $CAs$ from $RSD$ to six $CAs$ from $VSD$. The algorithm has successfully corrected the pose error between the $RSD$ and $VSD$ in four iterations. In the fourth scenario the actual pose different between the real robot and the virtual sensor is as follows:

$$\Delta_\theta = 25^\circ, \quad \Delta_X = -100\text{cm}, \quad \Delta_Y = 100\text{cm} \quad (26)$$

The algorithm has successfully corrected the pose error between $RSD$ and $VSD$ in five iterations. The simulation results are illustrated in Figure 19.

6 Comparison

Since some of the outstanding localization techniques (e.g. MCL, PF, and etc.), in order to achieve the desired accuracy in pose estimation, highly rely on a large number of particles, it can be concluded that the proposed GSR algorithm outperforms these techniques in case of computational efficiency. However, in order to demonstrate the precision and convergence speed of GSR algorithm, it needs to be compared with the other group of the state-of-the-art localization methods under different positioning conditions. Hence, in this section a comparison between the proposed solution and some important solutions to the localization problem proposed in the recent past which are mainly based on landmarks recognition or scan registration is performed. For the reasons mentioned earlier we only concentrate on the methods that basically are based on laser rang data. For comparing the different methods we implemented each one, under 676 random positioning error conditions, on the environment shown in Figure 1. To better demonstrate the errors of GSR algorithm, Figure 20 illustrates the final pose estimations occurred in each simulation. The desired target location is centered at zero and the positions approached by the robot is marked by the solid circles. The results derived from this plot for the other localization methods are depicted in Figure 21. The figure summarizes the results of 676 positioning actions for each method in terms of average error, standard deviation, and maximum error in position and orientation estimation that occurred in each simulation. As can be seen from the figure, the error mean and variance associated with GSR algorithm is relatively low compared to the other methods of localization.

The result of the comparison of four terms, minimum and maximum $APE$ (i.e. accuracy in position estimation), minimum and maximum $AOE$ (i.e. accuracy in orientation estimation), and $AET$ (i.e. average elapsed time in each pose estimation operation), for the seven localization methods are demonstrated in Table 3.

These results are obtained through 4732 (i.e. $676 \times 7$) simulations on the proposed map for a laser range sensor with 10 meters maximum range and 180 degrees field of view. The simulation results show that even in large depth pose error conditions the error of the GSR algorithm is small and never exceeds 12 mm/0.83 deg in any of our simulations. As shown in Table 3, the minimum accuracy in the estimated robot pose obtained through the simulations for GSR algorithm is 1.2 cm in position and 0.83 degree in orientation, and the maximum accuracy is 0.1 cm in position and 00.08 degree in orientation, whereas the average elapsed time for correcting the robot pose error in each pose estimation operation is approximately 75 millisecond making it applicable to real-time resource-limited embedded robots that require high precision.
Fig. 20 Plots of the final pose estimation errors occurred through 676 different positioning actions. The desired target location is centered at zero and the positions approached by the robot is marked by the solid circles.

(a) Position estimation error.

(b) Orientation estimation error.

(c) Orientation estimation errors ($\varepsilon_\theta$ respect to $\varepsilon_x$).

(d) Orientation estimation errors ($\varepsilon_\theta$ respect to $\varepsilon_y$).

Fig. 21 Error distribution for different methods. The average and standard deviation of error in position and orientation estimation are indicated by boxes and bars, respectively. The maximum error in position and orientation estimation that occurred in each simulation is marked by cross.

(a) Position estimation error.

(b) Orientation estimation error.
Table 3 Comparison of robot localization methods

| Method  | MinAPE(cm) | MaxAPE(cm) | MinAOE(°) | MaxAOE(°) | AET (ms) |
|---------|------------|------------|------------|------------|----------|
| [43]    | 2.05       | 0.1        | 0.98       | 0.13       | 240      |
| [44]    | 4.5        | 1.9        | 3.8        | 0.9        | 603      |
| [45]    | 3.7        | 0.8        | 3          | 1          | 413      |
| [46]    | 2.7        | 0.9        | 2          | 0.4        | 240      |
| [47]    | 3.8        | 1.7        | 1.8        | 0.15       | 288      |
| [48]    | 5.9        | 3.5        | 2.3        | 0.56       | 134      |
| GSR     | 1.2        | 0.1        | 0.83       | 0.08       | 75.7     |

7 Conclusions

This paper presents a new approach to pose estimation for a mobile robot based on geometrical and topological analysis of the shape of laser range scans. The problem of robot positioning has been stated as finding matchings between two visualizations extracted from 2D laser range data. The method, considering a simulated visualization of the expected pose of the robot on a known map as the prediction, continually searches for pairings between the features of the prediction and observation (i.e. corner points), and tries to maximize the similarity between the two sets by finding the relative translation in an iterative process, and in this way obtains a sensibly accurate estimate of the robot pose. Experimental results have been described to confirm the appropriateness of the matching procedure in finding the most similar parts of two scans even in large depth pose error conditions. Experiments also show that even in the presence of outliers in the set of matched corners, using an appropriate matching strategy, the algorithm is able to correctly estimate the difference between the two poses.

The main advantages of GSR algorithm are the significantly low number of iterations to converge to a minimum error, computational efficiency, and robustness to large positioning errors. The minimum accuracy, obtained through the experiments, in the estimated robot pose for a laser range sensor with 10 meters maximum range and 180 degrees field of view was 1.2 cm in position and 0.83 degree in orientation, and the maximum accuracy was 0.1 cm in position and 0.08 degree in orientation, which demonstrates robustness and precision of the proposed algorithm.

References

1. Castellanos JA, Tardos JD (2000) Mobile robot localization and map building: a multisensor fusion approach. Springer, Kluwer, New York.
2. Dulimart HS, Jain AK (1997) Mobile robot localization in indoor environment. Pattern Recognition 30:99-111.
3. Gutmann J, Weigel T, Nebel B (2000) Fast Accurate and Robust Self-Localization in the RoboCup Environment 16:304-317.
4. Se S, Lowe D,Little J (2002) Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. The international Journal of robotics Research 21(8):735-758.
5. Drumheller M (1987) Mobile robot localization using sensor. IEEE transactions on pattern analysis and machine intelligence 9:325-332.
6. Gutmann JS, Burgard W, Fox D, et al (1998) An experimental comparison of localization methods. Proceedings 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems Innovations in Theory, Practice and Applications (Cat No98CH36190) 2:736-743.
7. Cox IJ (1991) An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle. IEEE Transactions on Robotics and Automation 7:193-204.
8. Gutmann JS, Schlegel C (1996) AMOS: comparison of scan matching approaches for self-localization in indoor environments. Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT’96) 6:61-67.
9. Weiss G, Puttkamer E, et al (1995) A Map Based on Laser-scans Without Geometric Interpretation. Proceedings of Intelligent Autonomous Systems 4 (IAS-4) 4 :403-407.
10. Lu F, Milios E (1994) Robot pose estimation in unknown environments by matching 2D range scans. IEEE Conference on Computer Vision and Pattern Recognition 2 :249-275.
11. Arras K, Tomatis N (1999) Improving robustness and precision in mobile robot localization by using laser range finding and monocular vision. European Workshop on Advanced Mobile Robots 7 :177-185.
12. Gutmann JS, Weigel T, Nebel B (2001) A fast, accurate, and robust method for self-localization in polygonal environments using laser-range-finders. Advanced Robotics 14:651-668.
13. Crowley JL, Wallner F, Schiele B (1998) Position Estimation Using Principal Components of Range Data. Proc of the IEEE International Conference on Robotics and Automation (ICRA’98) 3 :3121-3128.
14. Pozna C, Precup RE, Földesi P (2015) A novel pose estimation algorithm for robotic navigation. Robotics and Autonomous Systems, 63:10-21.
15. Jensfelt P, Wijk O, Austin D, et al (2000) Experiments on augmenting CONDENSATION for mobile robot localization. Proceedings 2000 ICRA Millennium Conference IEEE International Conference on Robotics and Automation Symposium Proceedings (Cat No00CH37085) 3:2518-2524.
16. Dellaert F, Fox D, Burgard W, et al (1998) Monte Carlo localization for mobile robots. Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat No99CH36288C) 2:1322-1328.
17. Thrun S, Fox D, Burgard W, et al. (2000) Robust Monte Carlo Localization for Mobile Robots. Artificial Intelligence 128:99-141.
18. Durrant-Whyte H, Bailey T. (2006) Simultaneous localization and mapping (SLAM): Part I. IEEE Robotics and Automation Magazine 13:99-110.
19. Bailey T, Durrant-Whyte H. (2006) Simultaneous localization and mapping (SLAM): Part II. IEEE Robotics and Automation Magazine 13:108-117.
20. Grewal M, Andrews A. (2001) Kalman Filtering: Theory and Practice Using MATLAB. California State University at Fullerton Vol. 5.
21. Welch G, Bishop G. (2006) An Introduction to the Kalman Filter. In Practice 7:1-16.
22. Mu noz AJ, Gonzalez J. (1998) Two-dimensional landmark-based position estimation from a single image. IEEE International Conference on Robotics and Automation, 1998 Proceedings 4:3709-3714.
23. Thrun S, Fox D, Burgard W. (1999) Markov localization for mobile robots in dynamic environments. Journal of Artificial Intelligence Research 11:391-427.
24. Burgard W, Fox D. (1996) Estimating the absolute position of a mobile robot using position probability grids. Proceedings of the national conference on Artificial intelligence 6:896-901.
25. Reuter J. (2000) Mobile robot localization using PDAB. IEEE Intl Conf on Robotics and Automation (ICRA).
26. Illah Nourbakhsh, Rob Powers, Birchfield S. DERVISH (1995) An Office-Navigating Robot. AI Magazine 4: 16-53.
27. Thrun S, Fox D, Burgard W. (2000) Monte carlo localization with mixture proposal distribution. Proceedings of the National Conference on Artificial intelligence 8:859-865.
28. Fox D, Burgard W, Thrun S. (1998) Active Markov localization for mobile robots. Robotics and Autonomous Systems 25:195-207.
29. Simmons R, Koenig S. (1995) Probabilistic Robot Navigation in Partially Observable Environments. Proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI) 3:1080-1087.
30. aR Cassandra, Kaelbling L, Kurien J. (1996) Acting under uncertainty: discrete Bayesian models for mobile robot navigation. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS ’96 2:963-972.
31. Bengtsson O, Baerveldt AJ. (2003) Robot localization based on scan-matching - Estimating the covariance matrix for the IDC algorithm. Robotics and Autonomous Systems 44:29-40.
32. Nieto J, Bailey T, Nebot E. (2007) Recursive scan-matching SLAM. Robotics and Autonomous Systems 55:39-49.
33. Tomomo M. (2004) A scan matching method using Euclidean invariant signature for global localization and map building. IEEE International Conference on Robotics and Automation, 2004 Proceedings ICRA’04 1:866-871 Vol.1.
34. Rosencrantz M, Gordon G, Thrun S. (2003) Decentralized Sensor Fusion with Distributed Particle Filters. Proceedings of Conference on Uncertainty in Artificial Intelligence 4:493-500.
35. Rigatos GG. (2010) Extended Kalman and Particle Filtering for sensor fusion in motion control of mobile robots. Mathematics and Computers in Simulation 81:590-607.
36. Lee JS, Sun YN, Chen CH. (1995) Multiscale corner detection by using wavelet transform. IEEE Transactions on Image Processing 4:100-104.
37. Teh CH, Chin RT. (1989) On the Detection of Dominant Points on Digital Curve. Pami 11:859-872.
38. Rattarangsi A, Chin RT. (1990) Scale-based detection of corners of planar curves. 10th International Conference on Pattern Recognition 1990 Proceedings 1:923-930.
39. Sezgin T, Davis R. (2000) Sketch Based Interfaces: Early Processing for Sketch. ACM SIGGRAPH 2006 Courses 13:22-29.
40. Wolin A, Eoff B, Hammond T. (2008) ShortStraw: A Simple and Effective Corner Finder for Polylines. 5th Annual Workshop on SketchBased Interfaces and Modeling 5:33-40.
41. Nelder JA, Mead R. (1965) A simplex method for function minimization. The computer journal 7:308-313.
42. Spielman DA, Teng SH. (2004) Smoothed analysis of algorithms. Why the simplex algorithm usually takes polynomial time. Journal of the ACM (JACM) 51:385-463.
43. Lu F, et al. (1994) Robot pose estimation in unknown environments by matching 2d range scans. In: Computer Vision and Pattern Recognition, 1994. Proceedings CVPR’94., 1994 IEEE Computer Society Conference on. IEEE 9:935-938.
44. Xiaowei Z, Khing HY, Seng CC, et al. (2000) The localization of mobile robot based on laser scanner. In: Electrical and Computer Engineering, 2000 Canadian Conference on; Vol. 2. IEEE 3:841-845.
45. Rofer T. (2002) Using histogram correlation to create consistent laser scan maps. In: Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on; Vol. 1. IEEE 6:625-630.
46. Diosi A, Kleeman L. (2007) Fast laser scan matching using polar coordinates. The International Journal of Robotics Research 26:1125-1153.
47. Núñez P, Vazquez-Martín R, Bandera A, et al. (2009) Fast laser scan matching approach based on adaptive curvature estimation for mobile robots. Robotica 27:469-479.
48. Friedman C, Chopra I, Rand O. (2015) Perimeter-based polar scan matching (pb-psm) for 2d laser odometry. Journal of Intelligent & Robotic Systems 80:231-254.