LINE SEGMENT BASED RANGE SCAN MATCHING WITHOUT POSE INFORMATION FOR INDOOR ENVIRONMENTS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

by

İskender Yakın

July, 2008

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Uluç Saranlı (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Ali Aydın Selçuk

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Osman Abul

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

LINE SEGMENT BASED RANGE SCAN MATCHING WITHOUT POSE INFORMATION FOR INDOOR ENVIRONMENTS

İskender Yakın
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. Uluç Saranlı
July, 2008

A mobile robot exploring an unknown environment often needs to keep track of its pose through its sensors. Range scan matching is a way of computing the pose difference of a robot at two different locations on the navigation path by finding common features observed in range sensor readings recorded at these locations. In this thesis, we introduce a new algorithm which computes this pose difference by matching common line segments extracted from two laser range scans taken from two different but unknown poses. In this algorithm, matching is performed by exploiting invariant geometric relations among line segments. The use of line segments instead of range points also reduces the computational complexity of determining the pose difference between two distinct scans. Compared to other scan matching algorithms, our method presents a powerful means for global scan matching, map building, place recognition, loop closing and multirobot mapping, all in real-time.

Keywords: Scan Matching, feature extraction, mapping, localization, geometric relations, laser scan processing.

ÖZET

İÇ MEKANLAR İÇİN DOĞRU PARÇASI TABANLI MESAFE TARAMALARININ EŞLENMESİ

İskender Yakın
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Uluç Saranlı
Temmuz, 2008

Bilinmeyen bir ortamda keşif yapan seyyar bir robot, almaçları vazıtasıyla konumunu takip etmek durumunda kalabilir. Mesafe taramalarının eşlenmesi, robotun geçtiği seyir yolu üzerindeki iki farklı mevkide kaydedilen mesafe almacı kayıtlarında ortak olan özniteliklerin bulunmasıyla, bu mevkiler arasındaki konum farkının hesaplanmasıdır. Bu tezde, bilinmeyen ve farklı konumlarda kaydedilmiş lazer mesafe taramalarından çıkartılan, ortak doğru parçalarını eşleyerek konum farkını hesaplayan, doğru parçası tabanlı bir mesafe taraması eşleme algoritması sunulmaktadır. Bu algoritmada eşleme işlemi, doğru parçaları arasındaki, geometrik ilişkiler olarak adlandırdığımız, değişmez geometrik öznitelikler kullanılarak gerçekleştirilmektedir. Mesafe noktaları yerine bu noktalara oturtulan doğru parçalarının kullanılması iki farklı tarama arasındaki konum farkını kestirmek için yapılan hesaplamaların karmaşıklığını azaltmaktadır. Diğer mesafe taraması eşleme algoritmalarıyla kıyaslandığında, bizim metodumuzun küresel tarama eşleme, harita oluşturma, yer tanıma, döngü kapatma ve çoklu robot ile haritalama problemlerinin gerçek zamanlı çözümleri için etkili bir altyapı sunduğu görülmektedir.

Keywords: Tarama eşleme, öznitelik çıkartma, haritalama, konumlanma, geometrik ilişkiler, lazer taramsı işleme.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An autonomous mobile robot is a system which perceives its environment in order to use acquired information for solving a given task. For a mobile robot, autonomous navigation in its environment is one of the most important of all tasks. Robot navigation means the ability of a robot to determine its own pose (position and orientation) in its frame of reference and then to plan a path toward some goal location. As a result, pose estimation is a fundamental problem for autonomous navigation of most mobile robots.

In order to estimate pose, researchers and engineers have developed a variety of systems, sensors, and techniques. These can be categorized into two groups: relative (dead reckoning) and absolute pose estimation (reference-based systems) [7]. The fundamental idea behind relative pose estimation is the integration of incremental motion information over time, inevitably leading to unbounded accumulation of errors, so that the reliability of pose estimation decreases over distance [25]. Among absolute pose estimation techniques, *Map Based Positioning* is the only one which does not require the installation of a positioning aid such as magnetic compass, or the deployment of external references such as active beacons. Map based pose estimation can be accomplished by the use of active (laser scanners, ultrasonic or infrared sensor rings) or passive (stereo vision, binocular vision cameras) range sensors that may already be installed on a mobile robot platform for environmental sensing tasks such as obstacle detection and avoidance. By interpreting data acquired from these sensors, natural landmarks (walls, corners, corridors, etc.) present in an indoor environment can be identified and used as external positioning references.

Generally two methods, one from each pose estimation category, is combined due to the lack of a single good method. Map based positioning techniques are mostly coupled with odometry which is among most widely used relative pose estimation method. It provides good short-term accuracy, is inexpensive, and allows very high sampling rates. When using odometry as the basis for this combined system, the maintenance of accurate pose estimation

over time and distance depends on the accuracy, reliability and sampling speed of the range sensor along with the robustness and running-time of the chosen map matching technique.

Robots frequently use active sensors for more reliable range sensing since passive sensors suffer from image intensity variation due to illumination noise, insufficient feature information on environment composed of plain surfaces, and correspondence problem between multiple images. In many approaches to indoor robot applications, laser scanners have been preferred for detailed sensing and object modeling, due to better range accuracy, denser range data and very high sampling rates compared to other active range sensors.

Robustness of a map matching method employing a laser range scanner is dependent on the robustness of the underlying range scan matching algorithm. A *range scan* (or simply a scan) is a finite sequence of numbers, where each element is a number representing the distance to the nearest obstacle in the direction associated with this element. The assignment of angles to elements in this sequence is in a consecutive manner and evenly spaced. *Scan matching* is the estimation of a robot's pose by matching a pair or range scans. The first scan, S_r, serves as the reference scan whereas the second scan, S_c, is called the current scan. S_c is matched against S_r in order to find the pose of S_c relative to S_r. The result of the match is a pose correction to the current robot pose. Furthermore, once S_c is aligned over S_r, it is merged with the map.

The correctness of this scan alignment determines how precisely the pose difference is estimated and also depends on the representation of range scans. Instead of points, representing a range scan with fitted line segments improves the precision of the alignment by reducing the drift of points from ideal line segments. A line segment is a simple feature. Consequently, maps based on line segments represent a middle ground between highly reduced feature maps and massively redundant raw sensor-data maps. Clearly, line segment based maps are most suited for indoor applications, or structured outdoor applications, where objects with straight surfaces comprise many of the environmental features. Relatively simple representation of line segments also reduces the computational complexity of associated scan matching algorithms.

This thesis introduces a new method for robust global range scan matching by using geometric relations derived from line segments fitted to range scan data. The basic idea behind our method is the observation that, if common line segments corresponding to static structures in the environment exist in both line segment sets extracted from two distinct range scans recorded at different poses, then the relative geometry between those line segments must remain the same in both observations. Naturally, there will be noise and dynamic obstacles in each observation, which may result in false line segments. There may also be valid line segments which are not common to both observations due to the different viewpoints from which they were taken. The aim, therefore, is to find a one-to-one mapping of line segments common to both scans. This is done by selecting the largest subset of line segments where geometric constraints between line segments are mutually satisfied. Our method is

also capable of matching scans in real time without any pose information.

Even though odometric information is often available, one of the reasons for focusing on pure scan matching methods is that we want to be able to use the same or a modified version of our method for different tasks such as global scan matching, map building, place recognition, loop closing and multirobot mapping. Another reason is that, sometimes it may be desirable to interrupt one of these tasks and resume it at a later time without having to reset the initial pose(s) of the robot(s). This provides a solution to the so-called kidnapped robot problem [11].

Chapter 2

Overview of Scan Matching Techniques

In a pure geometric sense, scan matching is the process of finding a rotation θ and a translation T maximizing the overlapping of two groups of two dimensional data sets. Following this interpretation, scan matching approaches can be classified according to methods used to find the maximum overlap between the two scans. These methods can be further classified with respect to their use of odometry.

2.1 Scan Matching with Odometry

Scan matching methods relying on relationships between feature sets of current and reference scans (or map), require an accurate initial estimate for the displacement provided by odometry, because as the displacement between scans increases, the accuracy of feature relationships decreases. This results in incorrect correspondences between features which means an erroneous matching of scans.

2.1.1 Iterative Approaches

A well-known scan matching method is the iterative method presented in [8] for matching range scans to an *a priori* map of line segments. This method depends on odometry for estimating the initial alignment of the current scan. The current scan is matched to the map iteratively by finding the correspondences between scan points and line segments in the map. In each iteration, the translation and rotation that minimizes the total squared point to line segment distances are computed based on these correspondences. These two steps are

repeated until the procedure converges. This approach was extended in [12]. Instead of using an *a priori* map, scan points in the current scan are matched to line segments extracted from previous scans. The major limitation of these methods is that they can only be applied only to polygonal environments.

The method proposed in [18] also matches the current and reference scans iteratively by using a least squares method similar to [8]. This method iteratively minimizes an error measure by first finding a correspondence between points in the reference scan and points in the current scan, and then doing a least squares minimization of all point-to-point distances to determine the best pose difference. An initial pose estimate is provided through odometry to avoid erroneous alignments. The computation cost of IDC is high and the method does not seem to be suited for polygonal environments. This method is extended in [6] by reducing noise sensitivity of original IDC and by refining it to cope with dynamic environments.

2.1.2 Histogram Matching Approaches

The method proposed in [28] uses points to represent range scans. This method first creates a histogram of angles between consecutive point pairs. Rotational difference between the scans is computed by correlating angle histograms across two scans through a cross correlation function. For computing translation, x and y coordinate histograms of points are compared. This method requires a good initial position estimate since the cross correlation function tends to produce incorrect results in the presence of large displacements between scans. The major drawback of this method is that the algorithm performs well only in environments that consist of straight perpendicular walls. The other drawback is that it only allows for minor changes in the environment.

The improvement in [22] deals with non-perpendicular walls, even though it still assumes straight walls and shows poor performance in scattered environments. In [23], an extension to the method is proposed. Instead of matching two complete scans, a *projection filter* [17] is first applied two both scans. Based on an estimated offset between both scan poses, this new method removes all points from one scan that result from surfaces that cannot be seen from the recording position of the other scan and vice versa. Then, instead of only using neighboring scan points, line segmentation is applied to determine the orientations of the surfaces. The resulting lines are used to calculate the histograms. This method relies only on odometry for initial position estimate and runs in real-time.

2.1.3 Closest-Feature Matching Approaches

The closest feature matching approach presented in [31] matches two sets of line segments corresponding to the current scan and a global map, respectively. In order to find corresponding line segment pairs, line segments in the current scan are first updated with respect to odometry. Subsequently, a matching check is performed for each current line segment against lines in the global map, based on the directions and distance between center points of line segments. Once matching line segments are found, rotational difference is computed by averaging angular differences between matching line segments. The Weighted Least Squares method is used to find the translational difference. A special center of gravity representation is used to describe the uncertainty of line segments and variances on the center of gravity are used as weighting factors. The method proposed in [27] refines the alignment of scans by using the partial Hausdorff distance, computed on the original laser data and finds the best alignment between the global map and the current scan. This method also requires an initial estimate of the pose of the scans.

Similar to [31], the method proposed in [4] matches two sets of line segments corresponding to the current scan and the global map by correlating closest line segments with respect to their midpoints, assuming that the pose estimate of the current scan is close enough to the real pose such that new line segments match up with their counterparts in the map. The relative orientation of the two maps is determined by computing a histogram of angle differences and then the translation is adjusted by overlapping the midpoints of line segments using least square minimization. The method works for linear and static environments and for very small displacements.

2.1.4 Probabilistic Approaches

The probabilistic line segment matching method presented in [9] depends on odometry for initial alignment of current scan over reference scan assuming that range data is obtained in small displacements and the odometry error is small. After the initial alignment step, the total probability of pairing two segments is computed. Pairing probability is the product of probabilities of three different characteristic factors: parallelism, parallel distance and overlapping length of line segments. This method produces a probability table from computed pairing probabilities and selects line segment pairs with higher probabilities as correct line segment matches.

2.2 Scan Matching without Odometry

There are also several attempts to match scans in the absence of any pose information. All scan matching methods which do not require an initial pose estimate rely on relative feature relationships defined within the same feature sets of current and reference scans. Matching is done by correlating two sets of relative feature relationships, which in turn enables correlating features between current and reference feature sets.

2.2.1 Pattern Recognition Approaches

The method proposed in [10] uses a panorama laser range finder and identifies line segments representing linear structures in the environment. A line segment map of the environment is created by matching two sets of line segments without any additional data about the poses of corresponding range scans. This is accomplished by pattern matching and pattern recognition on line segment sets through a dynamic programming algorithm. In this context the term *pattern* denotes the set of line segments. The matching of two patterns is done by finding the optimal path through a matrix of grid points which is spanned by the similarity measures between line segments sets as a cost function of Hesse normal representation parameters. The method operates in polygonal or rectilinear environments, but does not work well in scattered environments. It also relies on small displacements of the robot.

2.2.2 Shape Matching Approaches

In [15], a comprehensive geometric model for robot mapping based on shape information is presented. Polygonal lines, called *polylines*, serve as the basic representation of shape as a structure of boundaries. Matching two shapes means matching two ordered sets of polylines againts each other according to their similarity. The similarity measure utilized in this approach is based on a measure introduced in [16]. To compute the basic similarity measure between two polygonal curves, the best possible correspondence of maximal left or right arcs are established. Computing the actual matching of two structural shape representations extracted from scan and map is done by finding the best correspondence of polylines respecting a cyclic order. This method is also capable of matching polylines in the absence of odometry by means of the distinctive property of shape similarity.

2.2.3 Graph Theoretic Approaches

The data association algorithm presented in [5] operates purely through the matching of relative constraints and feature types (points and line segments), having the effect of enabling

batch data association without a priori knowledge of the relative pose between data sets. This method is valid when features are observed as a batch observation such that they have accurate relative geometric information. The mapping of common features between two feature sets is transformed into the graph theoretic problem of finding the maximum common subgraph (MCS) which, in turn, can be represented as a maximum clique problem.

Another graph theoretic approach presented in [14] matches current scan with one of the reference scans by identifying the maximum matching subgraphs in the set of all reference graphs. Graphs are constructed by anchor points, which are feature positions corresponding to edges in the environment. This method defines three types of anchor points: *jump*, *angle* and *virtual edge* anchor points. Anchor points are detected through angle histograms as described in Section 2.1.2. Distances between anchor points form the edges of the corresponding graph. In environments which do not provide a sufficient number of anchor points, alignments cannot be determined.

2.2.4 Relative-Geometry Matching Approaches

The scan matching method proposed in [30] matches two scans without odometry by using geometric features based on line segments, also called Complete Line Segment (CLS) relationships. The method singles out complete line segments that represent complete linear structures in the environment and uses them to match between the local and global maps. Line segments are sorted in a counterclockwise fashion in both maps in order to improve search efficiency. Matching between the current range scan and the global map is based on relative position relationships of line segments in both maps. Relative position relation of a CLS to another CLS consists of three parts: relative position of center point to line segment, relative orientation of line to line segment and relative length of line segment to line segment. For each line segment in the local map, a consistent line segment in terms of its length in the global map is selected as a candidate match and the likelihood of trial localization is computed by testing whether other line segments in the local map has corresponding line segments in global map based on this trial localization. Finally, the trial localization with the maximum likelihood is singled out as the best matching between the current scan and the environment map. The position of the current scan is computed based on the maximum likelihood. The method has been shown to be fast and accurate. However, it cannot handle partially visible line segments and causes a significant amount of data loss in environments with occluded objects. This method cannot be extended to multirobot map building with unknown poses of the robots, since it is based on a sorting order of line segments to improve its search efficiency. Sorting order also eliminates the potential use of line segments in closed line segments cluster such as linear columns present in the environment, since such clusters change the sorting order of line segments in local map depending of the pose of the current scan.

Another geometric approach proposed in [3] only uses angles between line segment pairs to match the current and reference scans. This method computes relative angles between line segments within the same set corresponding to either the current and the reference scan. A possible transformation is determined for the current scan for each equal relative angle and the total length of overlapping line segments between line segment sets is computed in order to evaluate the correctness of the transform. The method is extended in [1] to build a global map of an environment. A further extension in [2] is capable of building global maps with multiple robots without using any knowledge about relative poses of robots.

2.2.5 Geometric Hashing Approaches

The method presented in [26] extends the geometric hashing technique of [29], originally developed for computer vision to match geometric features against a prior database. The main idea is a signature representation of the local region around each point in the scan. The search for the best alignment between two scans is performed with a voting system in the Hough space containing all the signatures. Even though this method does not require an initial pose estimate, it is implicitly based on the assumption of small pose difference between two scans.

Chapter 3

Extraction of Geometrical Primitives

As described in Section 2.2, pure scan matching can be used to find the difference between two distinct robot poses as in Figure 3.1 without any other pose information. Matching of two range scans recorded at different poses requires the identification of geometrical primitives common to both scans. Once relative poses of common geometrical primitives are found, it is easy to find the pose difference between the two scans.

Line segments and edges are among the most basic geometrical primitives that can be extracted from a range scan. While finding common line segments is enough to determine the rotational difference, common edges help compute the translational difference as well.

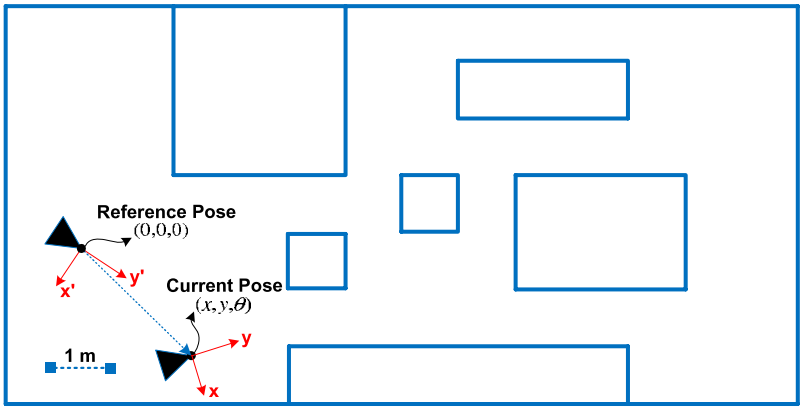


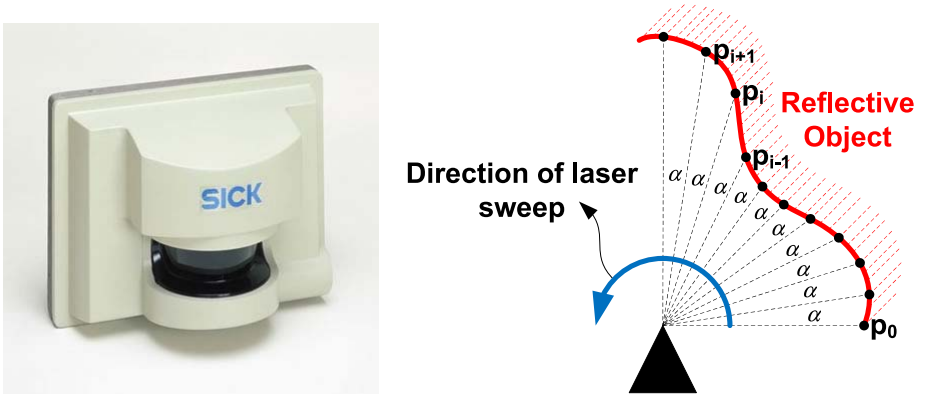
Figure 3.1: Two distinct robot poses on a 2D map. The *reference pose* stands for the first location visited by the mobile robot. The *current pose* is the current location of the robot. If the reference pose $(x, y, \theta)_r$ is known, the current pose $(x, y, \theta)_c$ can be computed by updating $(x, y, \theta)_r$ with the pose difference (x, y, θ). (x, y, θ) is also the absolute pose difference between two poses assuming that $(x, y, \theta)_r$ is $(0, 0, 0)$.

3.1 Sensing The Environment

Active range measurement is one of the most common sensory modalities available to mobile robots. Laser range-scanner is a popular active range sensor which produces range scans consisting of a set of points expressed in polar coordinates. In order to extract geometrical primitives from such a range scan, it should first be transformed to points on the cartesian (x, y) coordinate plane. The following sections describe all the steps starting from how a laser range scanner sweeps the environment, to how to get the points in (x, y) coordinates.

3.1.1 Laser Range Scanners

A *laser range-scanner* is a sensor which uses a laser beam in order to determine the distance to a reflective object. It operates on the time of flight principle by sending a laser pulse in a narrow beam toward the object and measuring the time taken by the pulse to be reflected off the target and returned back to the sender. In our experimental setup, we use a SICK LMS 221 range finder (shown in Figure 3.2(a)) mounted on a Pioneer 3AT mobile robot platform at a height of approximatively 100 cm.



(a) (b)

Figure 3.2: (a) SICK LMS 221 2D Laser Range-Scanner. (b) A *range scan* is the raw output of a laser range scanner consisting of a finite sequence of numbers representing the distance to the nearest obstacle in a particular direction.

3.1.2 Range Scans

A range scan is the raw output of a laser range scanner. As described in chapter 1, it is a finite sequence of numbers, where each number represents the distance to the nearest obstacle in the associated direction. The assignment of angles α to elements in this sequence is illustrated in Figure 3.2(b). Range values correspond to distances measured by a laser beam sweeping a 180° angular area in the counterclockwise direction at 1° intervals. A set

of points expressed in polar coordinates is the result of a complete laser sweep. The origin of the coordinate frame is usually the range finder itself.

3.1.3 Transforming Range Data to Points on the Plane

A 2D laser range finder sweeps the environment in the counterclockwise direction. Each sweep is called a *range scan* and consists of a list of n range values $\{r_0, r_1, r_2, ..., r_n\}$. Every range value r_i corresponds to the distance to an obstacle hit by the laser beam shot at an angle $i.\alpha$ where α is the constant angle between two laser shots as in Figure 3.2(b). Thus, a range scan describes a 2D planar slice of a 3D environment. In order to get a good computational and visual representation, each range value r_i is transformed to a point p_i on the (x, y) coordinate plane as in Figure 3.3 where we define

$$p_i = r_i . \begin{bmatrix} cos(i.\alpha) \\ sin(i.\alpha) \end{bmatrix}.$$

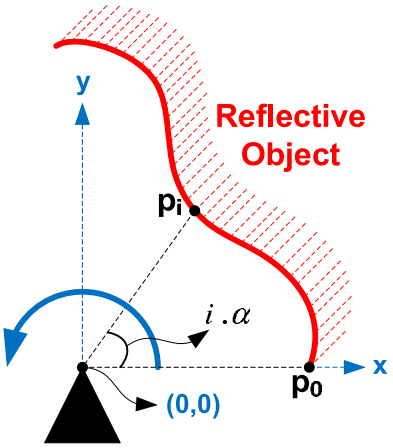
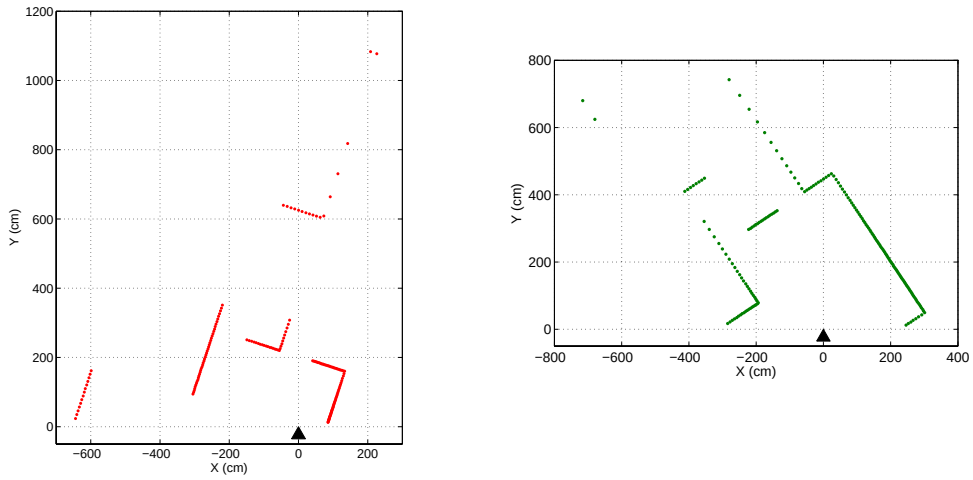


Figure 3.3: A point p_i is composed of x and y components computed according to the associated angle $\alpha.i$.

At the end of this phase, we get a list of points $P = \{p_0, p_1, p_2, ..., p_n\}$, corresponding to range values (as shown in Figure 3.4(a) for S_c). Figure 3.4(b) shows the points transformed from S_r. Note that the laser range finder is at the center of the (x, y) plane and oriented towards the positive y axis as illustrated in Figure 3.3.

3.2 Extraction of Line Segments

The distribution of points obtained from a range scan reflect the structure of the environment in which the corresponding range scan was recorded. If the environment is structured (with



(a) (b)

Figure 3.4: (a) Points transformed from S_c and (b) points transformed from S_r.

walls, doors, etc.) and points are dense enough to support assumptions about the geometry of the structure, then the range scan can be represented by higher level primitives such as line segments.

Indoor and structured outdoor environments are usually rich in linear structures. Scanned by a 2D range finder, these linear structures can be extracted by detecting sets of consecutive, collinear points. Fitting a line to each of these point sets yields a set of line segments in the range scan. In order to detect points corresponding to line segments, we use the *Split and Merge* algorithm given in [19], also shown below in Algorithm 1. This is the most popular line extraction algorithm, first introduced in 1974 in the context of computer vision [20]. This algorithm detects line segments in a range scan by first finding their endpoints. Points that are farthest from the line currently being fitted are assumed to be endpoints. Once a set of points that belong to a line is identified, the least-squares method is used for determining the associated line. After all line segments are extracted, collinear line segments are merged. Algorithm 1 shows the main steps of the algorithm.

Algorithm 1 Split-and-Merge

1: Initial: set A consists of n points. Put A in a list L.
2: Fit a line to the next unprocessed set A_i in L.
3: Detect point p_j with maximum distance d_{p_j} to the line
4: If d_{p_j} is less than a threshold t, go to 2
5: Otherwise, split A_i at p_j into A_{i1} and A_{i2}, replace A_i in L by A_{i1} and A_{i2}, go to 2
6: When all line segments have been cheched, merge collinear line segments.

Figure 3.5(a) and Figure 3.5(b) show line segments extracted from S_c and S_r respectively.

Sym.	Description
S_x	Scan x, where $x \in \{r, c\}$ denotes the scan type.
r_i	i^{th} range value in a scan
P	Point list extracted from a single scan
p_i	Point transformed from r_i
L	Line segment list extracted from a single scan
l_k	k^{th} line segment in a scan
$i_{(k,m)}$	Intersection of line segments l_k and l_m
G_x	Geometrical relation set of scan x
g_x	A geometrical relation in G_x

Table 3.1: Notational definitions. r and c denote reference and current scans respectively. $i \in [0, n]$ denotes range value index in a scan. k is the extraction number of a line segment in the counterclockwise direction.

After fitting to the range scan data, every line segment l_i (colected in a line segment list L) within a single range scan, is identified by its extraction number i, its start point s_i and end point e_i as illustrated in Figures 3.5(a) and 3.5(b).

We identify s_i and e_i of l_i to be either edge points or interior points based on their structural relation to the range scan. An *edge point* of a line segment stands for a visible corner in the environment formed by the intersection of two linear structures one of which is represented by that line segment. If both start and end points of a line segment are edge points, then this line segment is a complete line segment with length ℓ corresponding to a complete linear structure shown in Figure 3.6. In contrast, an *interior point* is any point of a line segment which does not represent an actual edge. The start and end points of a line segment can be interior points if the corresponding linear structure in the environment was only partially seen by the sensor as a result of occlusion caused by closer objects. If, at least, one of the start and end points of a line segment is an interior point, then the length of the whole linear structure represented by that line segment cannot be determined.

3.3 Extraction of Edges

In the context of line segment based representation, an *edge* is an endpoint of a 2D linear structure corresponding to a corner of a 3D flat object such as a wall. Edges are among the geometrical primitives used in our method. An edge extracted from line segments can be an *Angle Edge*, a *Jump Edge* or a *Virtual Edge* as shown in Figure 3.7. While angle edges and jump edges correspond to real structures such as corners of objects, a virtual edge stands for a virtual corner as the intersection of line segments corresponding to the linear structures in the environment.

- An **Angle Edge** appears in a scan when both enclosing planar object surfaces are

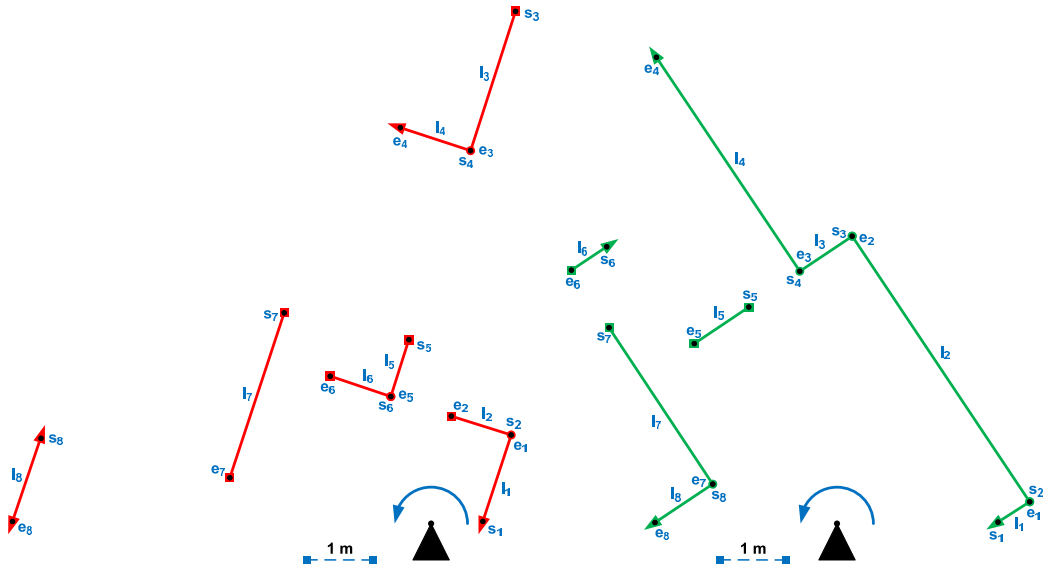


Figure 3.5: Line segments extracted from (a) S_c (b) and S_r. A line segment l_i is uniquely identified by its extraction number, its start point s_i and end point e_i. Line segments are numbered according to the order of extraction in the counterclockwise direction.

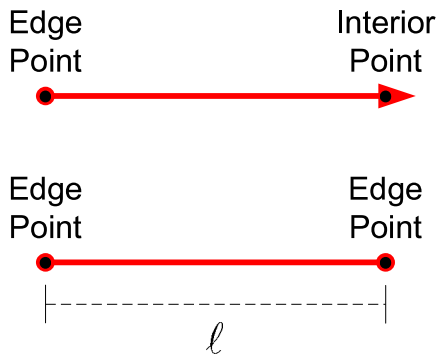


Figure 3.6: Start or end point of a line segment is either an edge point or an interior point.

visible. It is the intersection $i_{(k,k+1)}$ of two consecutive line segments l_k and l_{k+1} such that the end point of l_k and the start point of l_{k+1} are consecutive points p_j and p_{j+1} respectively, and $i_{(k,k+1)}$ lies between two laser beams corresponding to points p_j and p_{j+1}.

- A **Jump Edge** represents a corner in a linear structure that causes an occlusion in the visible sensor range and as a result, creates a jump in distance between two consecutive raw range values. It can be detected by looking at the points, p_{k-1} just before the start point p_k, and p_{m+1} just after the end point p_m of a line segment l_i. If either p_{k-1} or p_{m+1} is further from the origin than their projections p'_{k-1} or p'_{m+1} on l_i, then p_k or p_m are jump edges of l_i. Detection of jump edges help find line segments that have interior points as start or end points. If p_{k-1} is the end point of l_{i-1} or p_{m+1} is the start point of l_{i+1}, then these points are interior points as a result of the occlusion

caused by l_i

- A **Virtual Edge** is the intersection $i_{(k,m)}$ of two line segments l_k and l_m which do not otherwise create an angle edge with each other.

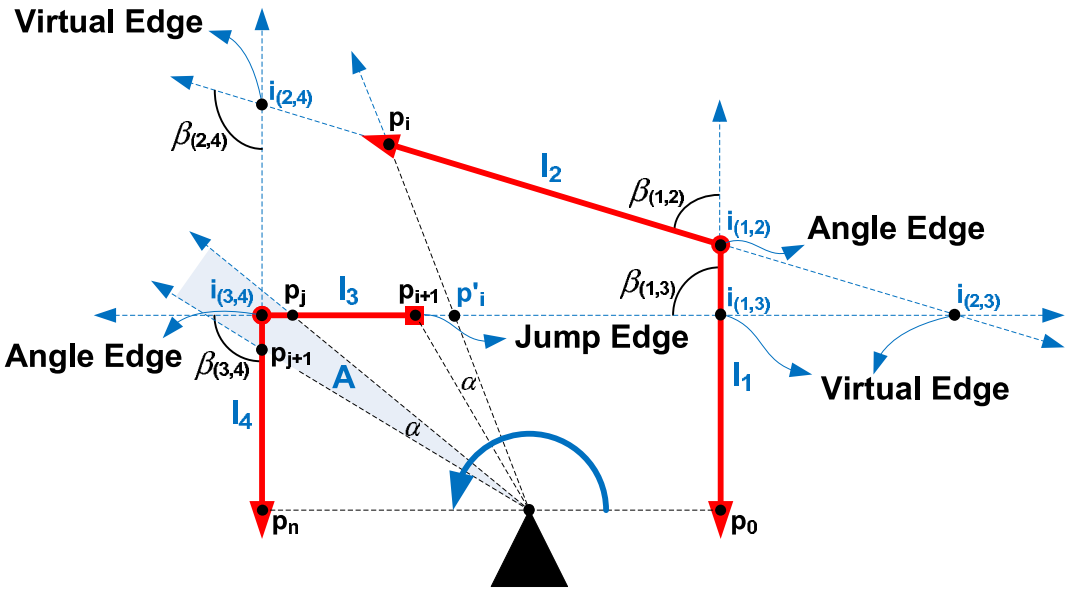


Figure 3.7: $i_{(3,4)}$ is an angle edge formed by the intersection of two consecutive line segments l_3 and l_4 such that the end point of l_3 and the start point of l_4 are consecutive points p_j and p_{j+1} respectively, and $i_{(3,4)}$ is within the area A between two laser beams which hit points p_j and p_{j+1}. The jump edge p_{i+1} can be detected by looking at the point p_i just before itself. p_i is further from the origin than its projection p_i' which is the intersection of l_3 and the laser beam which hit p_i, so p_{i+1} is the jump edge of l_3. Virtual edges $i_{(1,3)}$, $i_{(2,3)}$ and $i_{(2,4)}$ are the intersection of line segment pairs (l_1, l_3), (l_2, l_3) and (l_2, l_4) which are not consecutive.

Chapter 4

Extraction and Comparison of Geometrical Relations

We define a *geometrical relation* as either a property of a geometrical primitive or the relative geometry among several primitives extracted from a single scan. Defining geometrical relations based on relative geometry provides independence from pose, forming the basis for scan matching without explicit pose information. Examples of pose independent geometrical relations are length of a line segment, angle between two line segments, parallel line segments and distance between two edges. Assuming that the geometry of the environment at least partially stays the same, a sufficient number of geometrical relations are expected to remain invariant in both scans.

Extraction of geometrical relations forms two sets G_c and G_r, corresponding to S_c and S_r, respectively. If similar geometrical relations exist in G_r and G_c, geometrical primitives in S_c can be matched with geometrical primitives in S_c as will be explained in Chapter 5. Two geometrical relations match if their parameters are compatible and corresponding primitives are consistent.

4.1 Consistency of Geometrical Primitives

One of the preconditions for two geometrical relations to match is the consistency of their associated geometrical primitives. If a primitive of a geometrical relation in G_c is not consistent with its corresponding primitive belonging to a relation in G_r, then the two associated relations are determined to be not similar.

4.1.1 Consistency of Line Segments

Consistency of two line segments can be checked by comparing the type of their start and end points according to the following criteria.

- If one line segment is shorter than the other and start and end points of the shorter line segment are edge points, it is evident that these line segments cannot match.
- If at least one end point of the shorter line segment is an interior point, then these line segments can match.
- If all start and end points of both line segments are edge points, then these line segments may match provided their lengths are sufficiently close to each other. Otherwise, they cannot match.

Two line segments l_k in S_c and l_m in S_r are *consistent* if the criteria given above are satisfied. l_k and l_m can match only if they are consistent with each other. Consider the example, in Figure 4.1. It is evident that l_2 in S_c is not consistent with l_2, l_4, l_7, and l_8 in S_r according to the first criterion, because it is a complete line segment (start and end points are edge points) and is shorter than l_2, l_4, l_7, and l_8 in S_r. However, it is consistent with l_1 and l_6 in S_r according to the second criterion, because start points of l_1 and l_6 are interior points. Since l_2 in S_c and l_3, l_5 in S_r are complete line segments and their lengths are equal, l_2 in S_c is also consistent with l_3 and l_5 according to the last criterion.

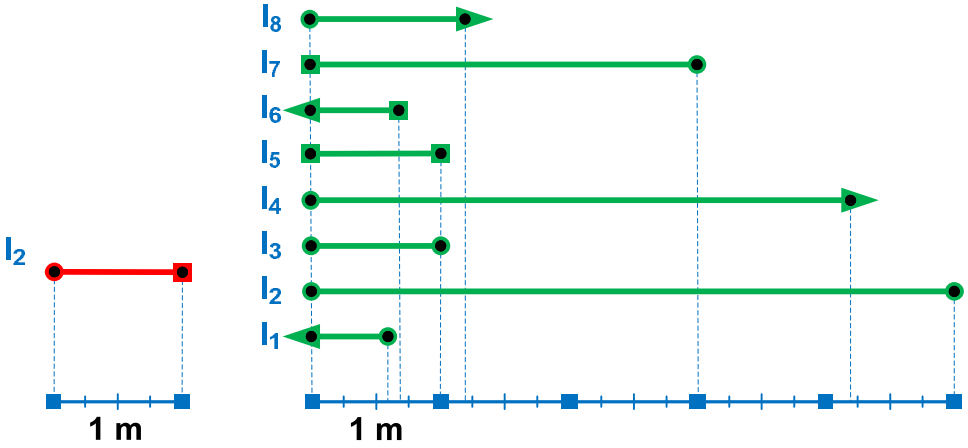


Figure 4.1: Line segment l_2 on the left is in S_c and other eight line segments are in S_r. l_2 in S_c is consistent only with l_1, l_3, l_5, and l_6 according to the line segment consistency criteria. l_2 in S_c can only match with these line segments in S_r.

4.1.2 Consistency of Edges

Consistency of an edge $i_{(k,m)}$ formed by l_k and l_m in S_c with an edge $i_{(u,v)}$ formed by l_u and l_v in S_r can be determined by checking the consistency of,

- The edge types. An angle edge can be compared with an angle or a jump edge and a virtual edge can be compared only with an edge of the same type,
- The angles $\beta_{(k,m)}$, $\beta_{(u,v)}$ between line segment pairs (l_k, l_m) and (l_u, l_v),
- Line segment pairs (l_k, l_u) and (l_m, l_v).

In order for an edge $i_{(k,m)}$ in S_c to be consistent with another edge $i_{(u,v)}$ in S_r, all criteria given above must be satisfied. If edges $i_{(k,m)}$ and $i_{(u,v)}$ are consistent, then l_k and l_m can match with l_u and l_v respectively as a result of the preservation of extraction order. Otherwise, l_k cannot match with l_u and and l_m cannot match with l_v. As an example, look at Figure 4.2(a) including an edge $i_{(1,2)}$ formed by (l_1, l_2) in S_c and Figure 4.2(b) including edges $i_{(2,3)}$, $i_{(6,-)}$, and $i_{(7,8)}$ formed by (l_2, l_3), l_6, and (l_7, l_8) in S_r. All edges except the jump edge $i_{(6,-)}$ are angle edges and are consistent according to edge type condition. Comparison of angles reduces the number of consistent edges to one. Only $i_{(2,3)}$ is consistent with $i_{(1,2)}$ according to the first two conditions. Even if $i_{(6,-)}$ is formed by a single line segment, the laser beam passing through e_6 during the scan process ensures that if there exists a line segment starting at $i_{(6,-)}$, it does not create an angle with l_6 less than $\beta_{(6,-)}$ which is greater than $\beta_{(1,2)}$. Finally, consistency of line segments concludes that $i_{(1,2)}$ in S_c is consistent only with $i_{(2,3)}$ in Figure 4.2(b).

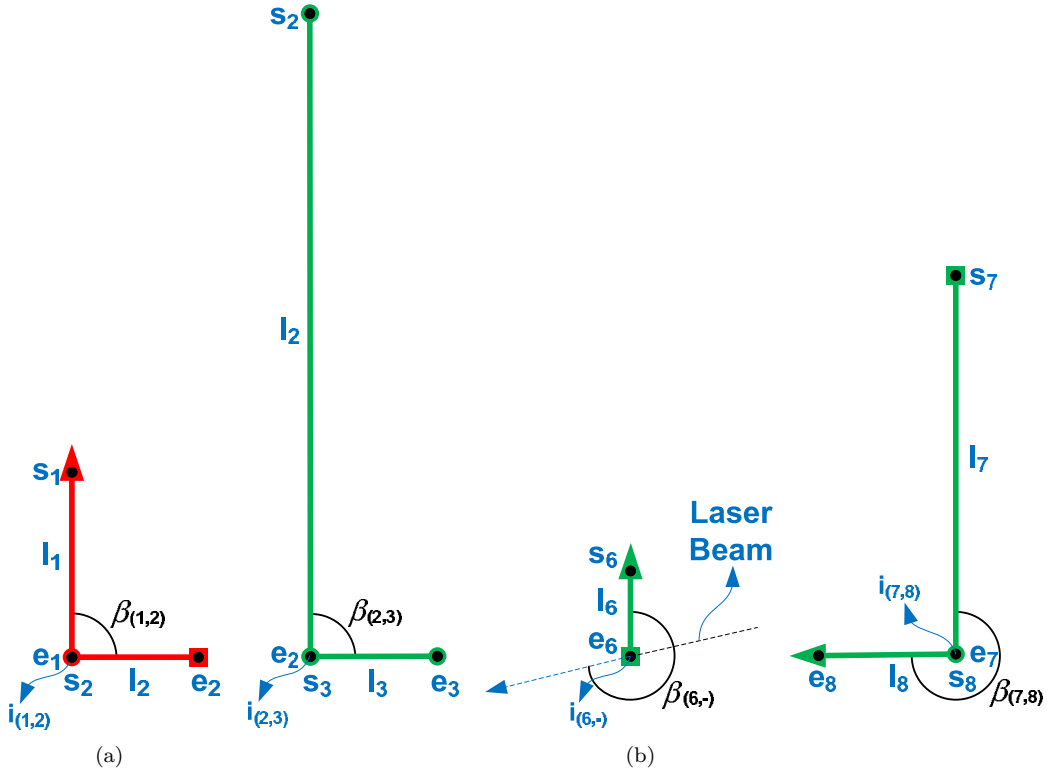


Figure 4.2: (a) Edge $i_{(1,2)}$ formed by (l_1, l_2) in S_c (b) and edges $i_{(2,3)}$, $i_{(6,-)}$, and $i_{(7,8)}$ formed by (l_2, l_3), l_6, and (l_7, l_8) in S_r.

4.1.3 Consistency Tables

Consistency information of line segments and edges are stored in consistency tables which are then used in the line segment matching phase. Using consistency tables helps efficiently identify geometrical relations which do not match due to inconsistencies between geometrical primitives from which they were extracted. A *consistency table* is simply a matrix of binary numbers representing the existence of pairwise consistency between line segments. A cell of a consistency table stores the consistency information between two line segments and it is indexed by the extraction numbers of these line segments. A cell is represented as,

$$T_{x\times c}(k,m) \in \{0,1\} \quad \text{where} \quad \begin{aligned} &1 \le k \le |L_x|\,, \\ &1 \le m \le |L_c|\,, \\ &x \in \{r,c\}. \end{aligned}$$

In this representation, $T_{x\times c}$ is a $|L_x| \times |L_c|$ matrix; L_x and L_c are line segment sets extracted from S_x and S_c, respectively. k and m are the extraction numbers of line segments l_k in L_x and l_m in L_c. A cell $T_{x\times c}(k,m)$ of table the $T_{x\times c}$ is indexed by these numbers and it can be either 0 or 1 showing the existence or lack of consistency between l_k and l_m. As an example, the consistency table $T_{r\times c}$ of line segment set L_r against L_c extracted from S_r and S_c is illustrated in Table 4.1. In the table, headers of the rows and columns are labeled with line segments in L_r and L_c respectively.

		L_c							
		l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
L_r	l_1	0	0	0	1	0	0	0	1
	l_2	1	0	0	0	0	0	0	1
	l_3	0	1	0	0	0	0	0	0
	l_4	1	0	0	0	0	0	1	1
	l_5	0	0	0	0	1	1	0	0
	l_6	0	1	1	1	1	1	1	1
	l_7	0	0	0	1	0	0	1	1
	l_8	1	0	1	0	0	0	1	1

Table 4.1: Consistency table $T_{r\times c}$ of line segment set L_r against L_c. A cell can be either 0 or 1. 1 shows that line segments forming the indices of the cell are consistent. 0 implies inconsistency between line segments.

For the line segment matching phase, the consistency table $T_{c\times c}$ is also created in addition to $T_{r\times c}$, since the numbers of matching geometrical relations within G_c are also required in order to determine the uniqueness of a geometrical relation as explained in Chapter 5. Table T_c^c is given in Table 4.2. Note that $T_{c\times c}$ is necessarily symmetric.

		L_c							
		l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
L_c	l_1	1	0	0	1	0	0	0	1
	l_2	0	1	0	0	0	0	0	0
	l_3	0	0	1	1	0	0	0	1
	l_4	1	0	1	1	0	0	1	1
	l_5	0	0	0	0	1	1	0	0
	l_6	0	0	0	0	1	1	0	0
	l_7	0	0	0	1	0	0	1	1
	l_8	1	0	1	1	0	0	1	1

Table 4.2: Consistency table $T_{c\times c}$ of line segment set L_c against itself.

4.2 Line Segment Length

Line segment length is the most basic geometrical relation used in our method. Each geometrical relation of type line segment length is represented as a labeled quadruple,

$$L(k, \|l_k\|, type(s_k), type(e_k)) \quad \text{where} \quad \begin{aligned} &1 \leq k \leq |L|\,, \\ &\|l_k\| \geq 0, \\ &type : P \rightarrow \{interior\ point, edge\ point\}. \end{aligned}$$

In this representation, k is the extraction number of the line segment l_k. $\|l_k\|$ is the length and $type(s_k)$, $type(e_k)$ are the types of start and end points of l_k. $|L|$ is the number of line segments and P is the list of points in the same scan.

Geometrical relations $L(k, \|l_k\|, type(s_k), type(e_k))$ and $L(m, \|l_m\|, type(s_m), type(e_m))$ of S_c and S_r, respectively, are compared according to the line segment consistency criteria explained in Section 4.1. Following these criteria, Algorithm 2 is used to check whether a geometrical relation of type line segment length can be matched with another one, in O(1).

Algorithm 2 finds the shorter line segment, checks the line segment consistency criteria and returns the result of the comparison. In the algorithm l_s stands for the shorter line segment.

4.3 Angle Between Two Line Segments

Another important geometrical relation used in our method is the relative angle between two line segments within a single range scan. Relative angles are widely used features by many scan matching methods. However, almost all methods narrow the relative angle window to the interval $[0, 180)$ by just computing the angle according to slopes of the line segments.

Algorithm 2 CompareLengths($L(k, \|l_k\|, type(s_k), type(e_k)), L(m, \|l_m\|, type(s_m), type(e_m))$

```
1: isCompatible = N.A.
2: l_s = null
3: if ||l_k|| < ||l_m|| then
4:    l_s = l_k
5: else if ||l_k|| > ||l_m|| then
6:    l_s = l_m
7: else
8:    l_s = null
9: end if
10: if l_s ≠ null then
11:    if type(s_s) and type(e_s) are of type edge point then
12:       isCompatible = false
13:    else
14:       isCompatible = true
15:    end if
16: else
17:    isCompatible = true
18: end if
19: return isCompatible
```

This type of angle computation reduces the uniqueness of the relation since both convex and concave corners are mapped into the same angle range.

In our method we adopt a different way of computing the relative angle between two line segments. Since laser range finder always scans the environment in the counterclockwise direction, the start point of a line segment is always scanned before its end point. This enables us to use one of the line segment as the reference line segment. By aligning the reference line segment l_k on the positive side of the x axis such that the end point e_k of the line segment coincides with the origin of the (x, y) coordinate system, we can increase the range of relative angles to $[0, 360)$. After placing the reference line segment, the current line segment l_m is placed on the coordinate system such that its end point e_m is at the origin as well. Computing the angle starting from the reference line segment to the current line segment in the counterclockwise direction gives us the relative angle between two line segments in the interval $[0, 360)$.

As an example, consider the situation in Figure 4.3. Consider l_2 as the reference line segment and l_1, l_3, l_4, and l_7 as line segments whose relative angles with respect to l_2 are to be computed. In this scenario, computing relative angles with respect to the coordinate axis of the laser range finder with the following formula,

$$\theta = arctan\left(\frac{m_u - m_v}{1 + m_u m_v}\right)$$

where m_u and m_v are the slopes of the line segments l_u and l_v, gives us 90°, 90°, 0°, and 0° for l_1, l_3, l_4, and l_7 respectively. The angle between two line segments can be in

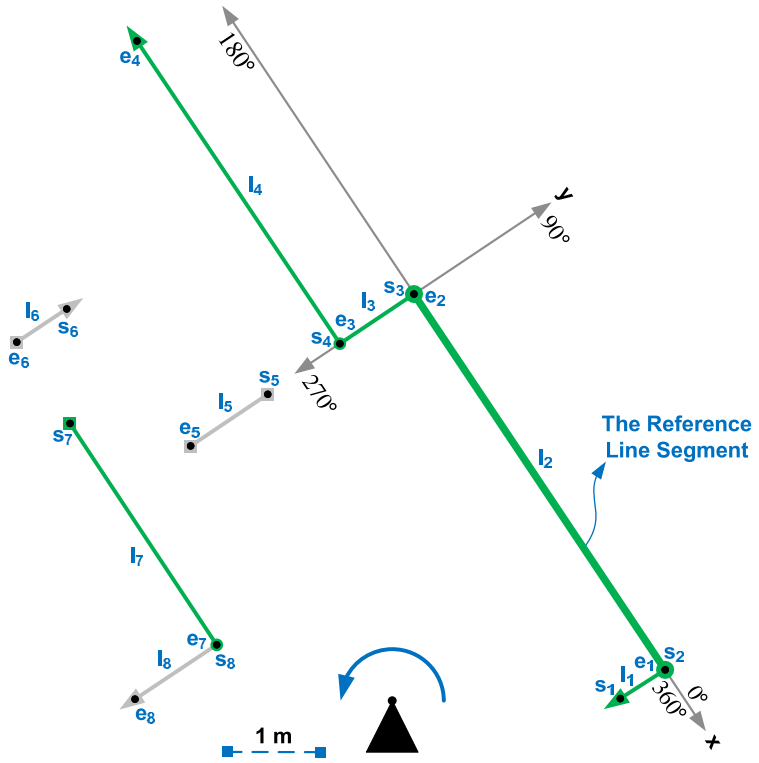


Figure 4.3: Line segments extracted from S_r. l_2 is the reference line segment in order to compute relative angles of l_1, l_3, l_4, and l_7 with respect to itself.

the angular interval $[0, 180)$ with respect to this formula. However, if relative angles are computed as explained above, we get 270°, 90°, 0°, 180° for the same line segments as in Figure 4.4, extending the interval to [0,360).

We define the geometrical relation defined above as a labeled triple,

$$A(k, m, \beta_{(k,m)}) \quad \text{where} \quad 1 \leq k < m \leq |L|\,, \\ 0° \leq \beta_{(k,m)} < 360°.$$

In this representation, $\beta_{(k,m)}$ is the relative angle between l_k and l_m, computed with respect to the reference line segment l_k. $|L|$ is the number of line segments in the same scan.

Comparison between relations $A(k, m, \beta_{(k,m)})$ and $A(u, v, \beta_{(u,v)})$ is done by looking at the relative angle between the line segments and their consistency. If the relative angle $\beta_{(k,m)}$ between l_k and l_m, computed with respect to the reference line segment l_k is different than the relative angle $\beta_{(u,v)}$ between l_u and l_v, computed with respect to the reference line segment l_u, then these geometrical relations do not match, which means l_k cannot match with l_u and l_m cannot match with l_v. If the relative angles are equal, then the consistency of line segments pairs (l_k, l_u) and (l_m, l_v) is checked as explained in section 4.1.1. If the line segments are consistent, then we say that $A(k, m, \beta_{(k,m)})$ and $A(u, v, \beta_{(u,v)})$ can match,

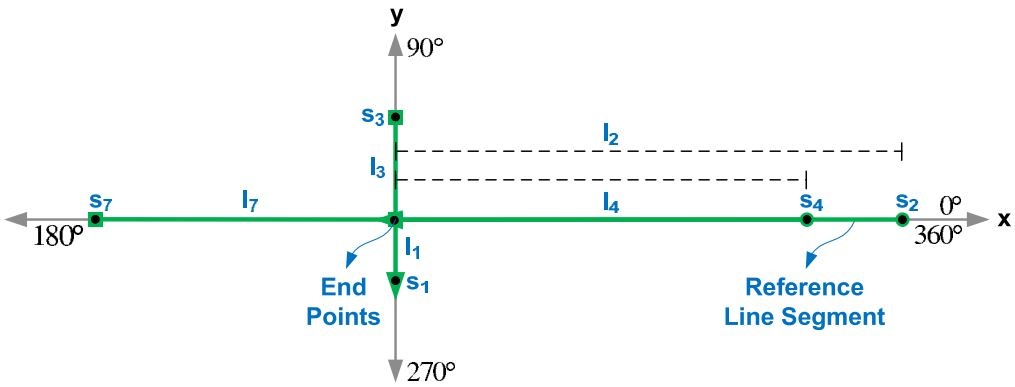


Figure 4.4: Illustration of the computation of the relative angles between l_2 and l_1, l_3, l_4, l_7. Line segments are translated and rotated such that their end points are at the origin and l_2 lies on positive x axis of the coordinate frame. As a result, relative angles between l_2 and other line segments are 0°, 90°, 180°, 270° for l_4, l_3, l_7, and l_1 respectively. These are actually angle differences in the counterclockwise direction between reference and other line segments.

implying line segment pairs (l_k, l_u) and (l_m, l_v) can match. Otherwise, these line segment pairs cannot match.

4.4 Parallel Line Distance

Parallel line distance is another geometrical relation occasionally used by some of the existing scan matching methods. Most of the time, only the *vertical distance* between two parallel line segments is considered, as illustrated in Figure 4.5(b). However, parallel lines bear more pose invariant information other than just vertical distance. For instance, lines perpendicular to parallel line segments and passing through start and end points of these line segments help determine whether these line segments are overlapping or not as illustrated in Figures 4.5(a) and 4.5(c). Overlapping parallel line segments are marked as *Overlap*. By looking at the type of start and end points, parallel line segment pairs can also be marked as *May Overlap*, and *No Overlap* if they are not overlapping. If two line segments cannot overlap, *horizontal distance*, another pose invariant property of two parallel lines is introduced. In case of an overlap, the *overlap length* can also be used as a property.

Consider the current scan S_c illustrated in Figure 4.6. In this scan, l_1 is parallel to l_3, l_5, l_7, and l_8. l_1 does not overlap with l_3 and l_5 with the same horizontal and different vertical distances. It overlaps with l_7 at least $\|l_1\|$, and it may overlap with l_8 since both l_1 and l_8 are incomplete line segments as a result of their start points to be interior points.

In addition, relative angle between parallel line segments helps to distinguish similar parallel line segment pairs in terms of parameters such as vertical distance and overlap type. The relative angle between parallel line segments is computed as explained in Section 4.3 and

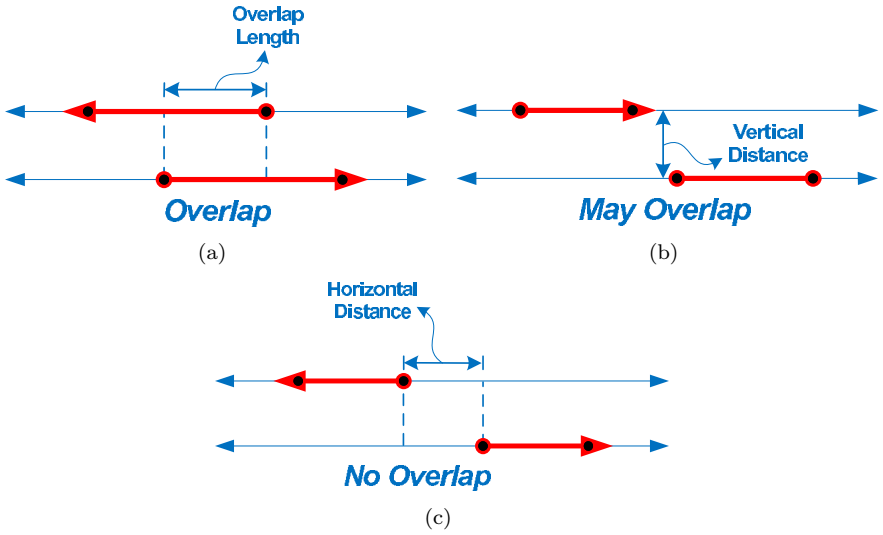


Figure 4.5: By looking at the type of start and end points, parallelism between two line segments can be marked as (a) *Overlap*, (b) *May Overlap*, or (c) *No Overlap*. If two parallel line segments cannot overlap, the horizontal distance between these line segments can be used as another pose invariant property.

can be either 0° or 180°. For instance, as illustrated in Figure 4.7(c), line segment pairs (l_1, l_4) and (l_2, l_3) are similar in terms of both vertical distance and overlapping type. However, the relative angle between line segments l_1 and l_4 is 0° as illustrated in Figure 4.7(a), and the relative angle between l_2 and l_3 is 180° as shown in Figure 4.7(b) where l_1 and l_2 are reference line segments. The reason for the relative angles to be different is that, the line segment pair (l_2, l_3) represents a corridor while (l_1, l_4) does not. As a result, incorporating relative angle information between line segments into the definition of a geometrical relation of type parallel line segments, contributes to an increase in the distinguishability of geometrical relations.

A geometrical relation of type parallel line distance is denoted by a labeled sextuple,

$$P(k, m, \beta_{(k,m)}, o_{(k,m)}, d^h_{(k,m)}, d^v_{(k,m)}) \quad \text{where} \quad \begin{array}{l} 1 \leq k < m \leq |L|\,, \\ 0° \leq \beta_{(k,m)} < 360°, \\ o_{(k,m)} \in \{\textit{overlap, may overlap, no overlap}\}, \\ d^h_{(k,m)}, d^v_{(k,m)} \geq 0. \end{array}$$

In this representation, $\beta_{(k,m)}$ is the relative angle, $o_{(k,m)}$ is the overlapping property, $d^h_{(k,m)}$ and $d^v_{(k,m)}$ are the horizontal and vertical distances between line segments l_k and l_m. If l_k and l_m are overlapping, then $d^h_{(k,m)}$ stands for the overlap length. In case of the overlapping property of the line segments to be *May Overlap* $d^h_{(k,m)}$ is undefined for the relation. Two geometrical relations of this type $P(k, m, \beta_{(k,m)}, o_{(k,m)}, d^h_{(k,m)}, d^v_{(k,m)})$ and $P(u, v, \beta_{(u,v)}, o_{(u,v)}, d^h_{(u,v)}, d^v_{(u,v)})$ match if all parameters are equal (or within the bounding

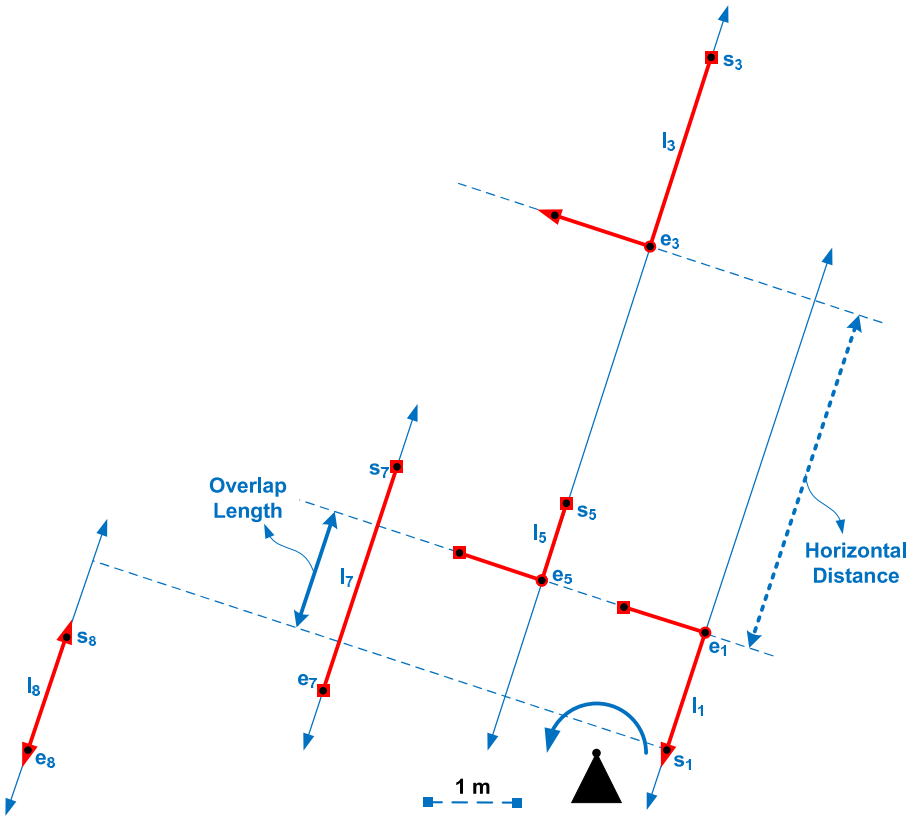


Figure 4.6: The current scan S_c illustrating the relationship between l_1 and l_3, l_5, l_7, l_8 in terms of parallelism.

error window). If geometrical relations match, then we can say that line segment pairs (l_k, l_u) and (l_m, l_v) can match.

4.5 Edge Distance

Distance between two edges is also an effective geometrical relation for determining whether line segments forming these edges can match. In order to increase the matching performance of this geometrical relation, we also consider additional angular relations between line segments forming the edges and the virtual distance-line as shown in Figure 4.8.

Edge distance is a geometrical relation which provides the highest level of data in terms of the environmental structure. A geometrical relation of this type is represented as a labeled septuple,

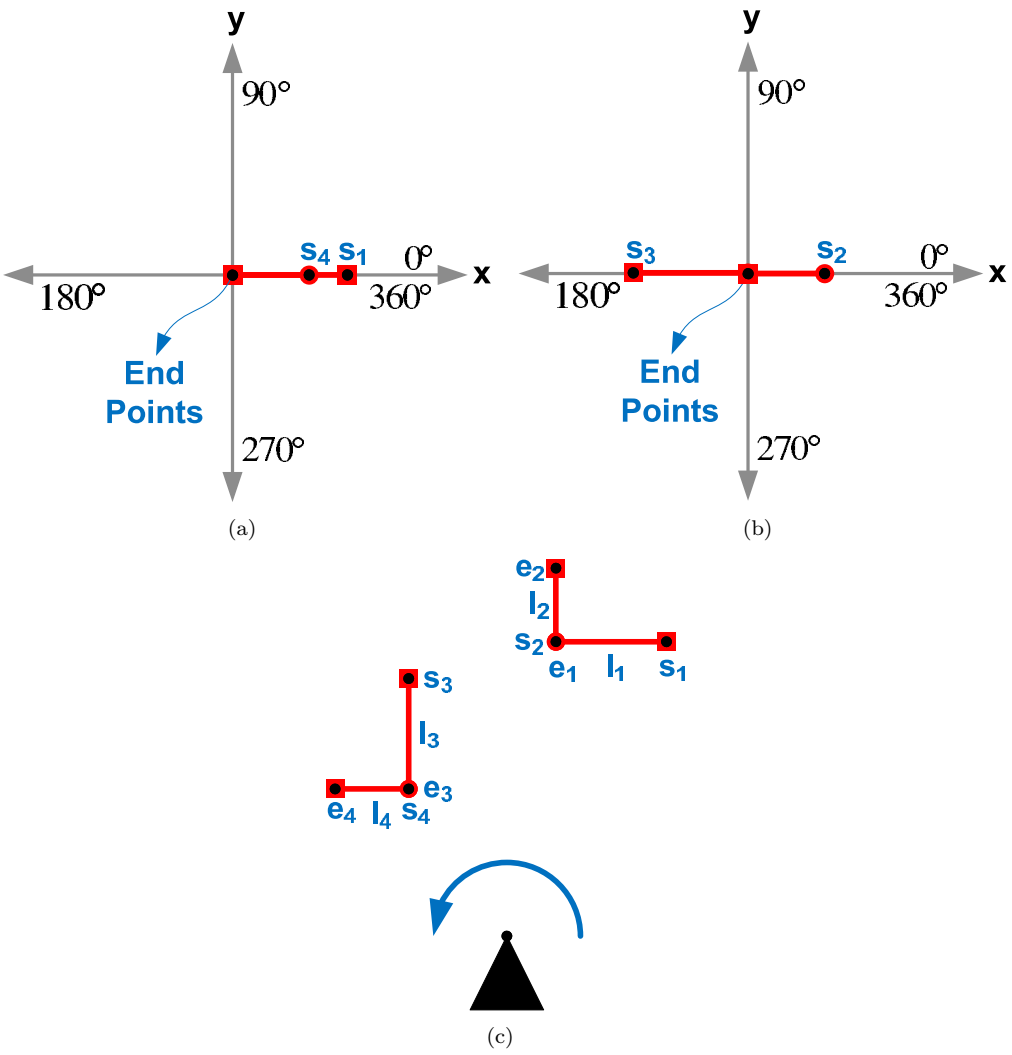


Figure 4.7: Parallel line segment pairs (l_1, l_4) and (l_2, l_3) are similar in terms of vertical distance and overlapping type. However, (a) the relative angle between l_1 and l_4 is 0° and (b) the relative angle between l_2 and l_3 is 180°.

$$E(k, m, d^e_{(k,m)}, \theta_1, \theta_2, \theta_3, \theta_4) \quad \text{where} \quad \begin{aligned} & 1 \leq k < m \leq |E|\,, \\ & 0° \leq \theta_1, \theta_2, \theta_3, \theta_4 < 360°, \\ & d^e_{(k,m)} \geq 0. \end{aligned}$$

In this representation, k and m are the extraction numbers of the edges. $d^e_{(k,m)}$ is the distance between the edges. θ_1, θ_1, θ_3, and θ_4 are the relative angles between line segments forming the edges and the distance line. These relative angles are extracted in the counterclockwise direction. $|E|$ stands for the number of edges in the same scan.

A geometrical relation of this type $E(k, m, d^e_{(k,m)}, \theta_1, \theta_2, \theta_3, \theta_4)$ match with another relation $E(u, v, d^e_{(u,v)}, \theta_1, \theta_2, \theta_3, \theta_4)$ if $d^e_{(k,m)}$ is equal to $d^e_{(u,v)}$ and angles are consistent with each

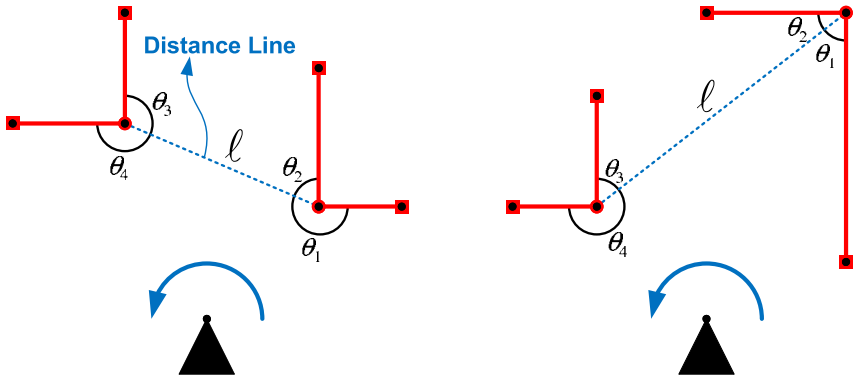


Figure 4.8: Edge distances and relative angles.

other. If an edge is a jump edge then one of the angles cannot be computed. In this case an upper bound for this angle is estimated as is done for l_6 in Figure 4.2(b).

Chapter 5

Line Segment Matching

After extracting geometrical relations, the next step in the scan matching process is to find line segments common to both scans. In this chapter, we introduce a novel method to solve this problem by defining a *distinguishability* measure. Once common line segments are determined, it is easy to compute the pose difference of the current scan S_c over the reference scan S_r by considering the pose differences between common line segments.

5.1 Distinguishability

Intuitively, we define distinguishability as the repetition frequency of a specific geometrical relation g_c both in G_c and G_r. If the total number of appearances of g_c in both geometrical relation sets is low, that is, if the distinguishability of g_c is high, then the possibility of correct matches between corresponding line segments of these geometrical relations is high. Formally, we define the distinguishability of a geometrical relation g_c in geometrical relation sets G_c and G_r as a score,

$$score = \frac{1}{Count(g_c, G_c, T_{c\times c}).Count(g_c, G_r, T_{r\times c})} \quad \text{where} \quad Count : (g_c, G_x, T_{x\times c}) \rightarrow \mathbb{N}, \; x \in \{r, c\}.$$

In this formula, the function *Count* yields the number of geometrical relations in a geometrical relation set G_x, matching with a specific geometric relation g_c. As the number of repetition of a specific geometrical relation g_c decreases in G_c or G_r, the score computed by the formula increases. This implies higher distinguishability for g_c and higher possibility for correct line segment matches. The details of the function *Count* are given in Algorithm 3.

Algorithm 3 Count$(g_c, G_x, T_{x\times c})$

```
1: count = 0
2: for all g_x ∈ G_x do
3:    isConsistent = true
4:    for all l_i ∈ g_x corresponding to l_j ∈ g_c do
5:       if T_{x×c}(i, j) ≠ 1 then
6:          isConsistent = false
7:       end if
8:    end for
9:    if isConsistent and geometrical parameters of g_x and g_c are consistent then
10:      count = count + 1
11:   end if
12: end for
13: return count
```

For every geometrical relation $g_x \in G_x$, Algorithm 3 first checks the consistency of line segments in g_x against the ones in g_c by using the consistency table $T_{x\times c}$ introduced in Section 4.1.3. If all line segments are consistent, then the consistency of all geometrical parameters in g_x and g_c are checked. If these are also found to be consistent, then g_x is determined to match to g_c, and the number of matching geometrical relations is incremented by one.

5.2 Matching Table

At the core of our line segment matching method is the idea of accumulating distinguishability scores in a matching table. This helps identify line segments common to two line segment sets L_r and L_c. A *matching table* is simply a $|L_r| \times |L_c|$ matrix of lists, storing identification numbers of some scores computed with respect to the distinguishability of geometrical relations in G_c against G_c and G_r. Every entry in a list is an identification number of a score computed for a specific geometrical relation g_c. Scores are stored in an array indexed by their identification numbers. The reason for this is to maintain a dynamic matching table with respect to the validity of scores.

Initially, all lists of a matching table point to valid scores by means of score identification numbers as illustrated in Figure 5.1(a). As the decisions about the correctness of line segment matches are given, some scores become invalid. For instance, assume that line segment pair (l_i, l_j) where $l_i \in L_r$ and $l_j \in L_c$ are determined to be an incorrect match. Then, it is evident that all score identification numbers stored in the list of (l_i, l_j) point to incorrect scores in the score array. Since most of our geometrical relations consist of two or more line segments, and since the same score is assigned to the line segments of a geometrical relation, the same score identification numbers corresponding to the invalid scores are in the lists of some other line segment pairs as well, as illustrated in Figure 5.1(b). As a result, this property helps maintain a dynamic matching table with respect to the validity of scores.

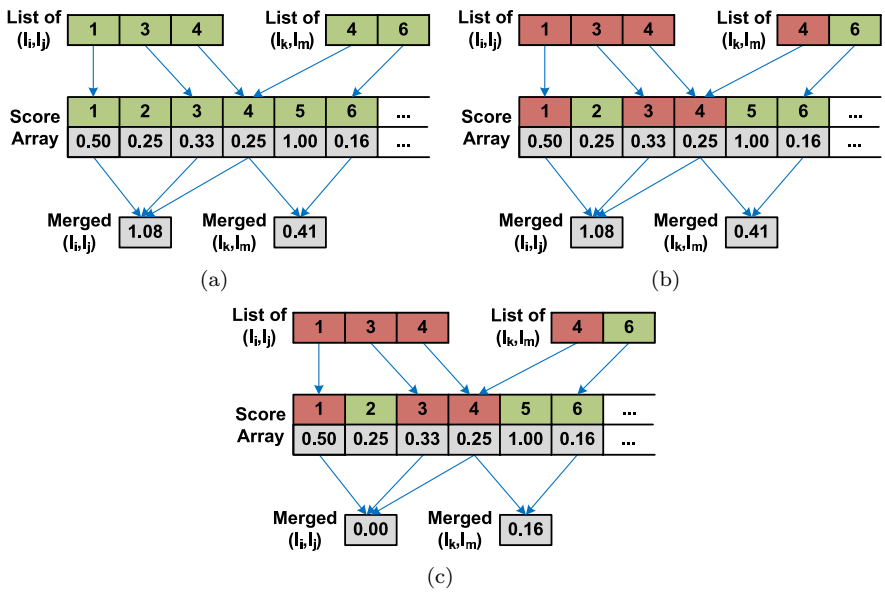


Figure 5.1: (a) All scores corresponding to score identification numbers in the given lists of (l_i, l_j) and (l_k, l_m) where $\{l_i, l_k\} \in L_r$ and $\{l_j, l_m\} \in L_c$ are initially valid. (b) In case that l_i is determined not to match with l_j, all scores corresponding to identification numbers in the list of (l_i, l_j) are marked as *invalid*. Identification number 4 is in both lists and it automatically becomes invalid in the list of (l_k, l_m). (c) Merged score for the pair (l_i, l_j) becomes 0.00 because all scores corresponding to identification numbers in their list are invalidated. Merged score for (l_k, l_k) goes down to 0.16 from 0.41 as a result of discarding invalid scores.

The structure of a matching table is similar to a consistency table introduced in Chapter 4.1.3, except that it holds list of scores instead of flags indicating the consistency of every line segment pair. For a given pair of geometrical relation sets G_c and G_r, Algorithm 4 is used to build the matching table for line segment sets L_c and L_r.

		L_c							
		l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
L_r	l_1	0.00	0.00	0.00	0.46	0.00	0.00	0.00	0.03
	l_2	***5.61***	0.00	0.00	0.00	0.00	0.00	0.00	0.14
	l_3	0.00	***9.75***	0.00	0.00	0.00	0.00	0.00	0.00
	l_4	1.28	0.00	0.00	0.00	0.00	0.00	0.25	0.14
	l_5	0.00	0.00	0.00	0.00	1.92	***7.50***	0.00	0.00
	l_6	0.00	1.58	1.17	1.68	0.58	1.50	2.25	0.20
	l_7	0.00	0.00	0.00	0.07	0.00	0.00	***6.25***	1.31
	l_8	0.11	0.00	0.17	0.00	0.00	0.00	0.92	1.87

Table 5.1: Merged matching table of line segment sets L_c and L_r.

For every geometrical relation g_c in G_c, Algorithm 4 finds geometrical relations in G_r matching with g_c. In order for two geometrical relations to match, all line segments and geometrical parameters of these geometrical relations must be consistent. If two geometrical relations g_c and g_r match, then the distinguishability score for these geometrical relations

Algorithm 4 ComputeScores(G_c, G_r)

```
1: matchingTable = |L_r| × |L_c| matrix of lists of scoreIDs
2: scores = list of |G_c| entries of type (score, isScoreValid)
3: scoreID = 0
4: for all g_c ∈ G_c do
5:    score = 1/Count(g_c, G_c).Count(g_c, G_r)
6:    for all g_r ∈ G_r do
7:       isConsistent = true
8:       for all l_i ∈ g_r corresponding to l_j ∈ g_c do
9:          if T_{x×c}(i, j) ≠ 1 then
10:            isConsistent = false
11:         end if
12:      end for
13:      if isConsistent and geometrical parameters of g_x and g_c are consistent then
14:         scoreID = scoreID + 1
15:         isScoreValid = true
16:         scores(scoreID) = (score, isScoreValid)
17:         for all l_i ∈ g_r corresponding to l_j ∈ g_c do
18:            add scoreID to the list at matchingTable (i, j)
19:         end for
20:      end if
21:   end for
22: end for
23: return (matchingTable, scores)
```

is appended to the lists of all corresponding line segments pairs in g_c and g_r, accumulating in the matching table. Note that the same score computed for a g_c is added to the lists of all corresponding line segments of g_c and every g_r matching with g_c. Even if we use merged scores in order to determine the correct matches, scores are appended to lists separately instead of a single merged score, since some of the scores in a list may become invalid. These misleading scores are eliminated as some line segment pairs are determined to not match in the line segment matching phase. As a result, the dynamic structure of a matching table necessitates the distinguishability scores to be stored separately in lists. In order to determine a correct match, a merged matching table is created corresponding to the current state of the matching table. A *merged matching table* is a $|L_r| \times |L_c|$ matrix where a cell indexed by (i, j) includes the sum of all valid scores stored in the list of the matching table indexed by the same indices (i, j) as illustrated in Figure 5.1(c). For example, the merged matching table transformed from the matching table created by Algorithm 4 for L_c in Figure 5.2(a) and L_r in Figure 5.2(b) is given in Table 5.1. Note that the merged scores for line segment pairs which really match are higher than the ones which should not be matched. In order to eliminate the scores of incorrect matches, that is, the invalid scores, Algorithm 5 is run on the matching table.

5.3 Line Segment Matching Algorithm

Higher scores in the merged matching table imply higher possibility for two line segments to correspond to each other. Using this information, the problem of matching line segments reduces to the problem of determining scores high enough for a correct match. In order to determine the correct matches, the most convenient way is to iterate over the highest merged scores. If the merged score in a cell indexed by (i, j) is high enough, this means that l_i in L_r and l_j in L_c are matching line segments. In order to find matching line segment pairs, Algorithm 5 is run on the matching table along with the list of scores computed for each g_c in G_c. Note that, Algorithm 5 should determine that, l_1, l_2, l_6, and l_7 in L_c, illustrated in Figure 5.2(a), match with l_2, l_3, l_5, and l_7 in L_r, illustrated in Figure 5.2(b) respectively.

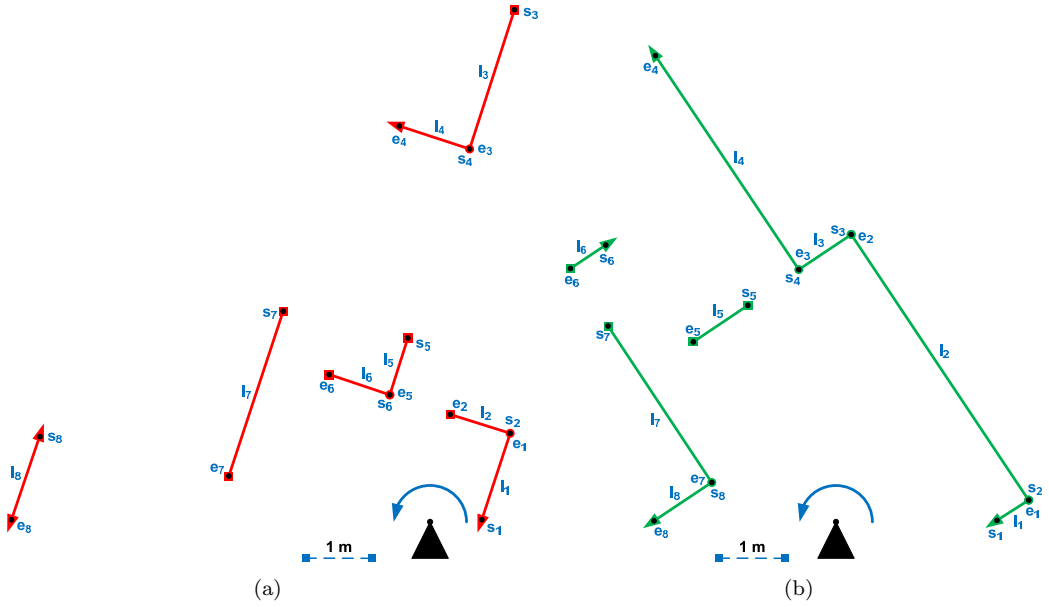


Figure 5.2: (a) l_1, l_2, l_6, and l_7 in L_c correspond to (b) l_2, l_3, l_5, and l_7 in L_r.

Algorithm 5 first finds the maximum of accumulated scores, *maximumScore*, in the table along with the indices (k, m) of this score by means of Algorithm 6. The maximum of accumulated scores indicates two line segments, l_i in L_r and l_j in L_c, which involve in the most distinguishable geometrical relations. Considering this line segment pair as a correct match, we know that the rotational difference between all other matching line segment pairs should be approximately the same with the rotational difference between l_i and l_j. Our algorithm uses this fact as a precondition for other correct matches. Once a pair of line segments is determined to be a correct match, it becomes evident that all the scores in the same row or column of the correct match are invalid. Assuming that the pose difference between two scans is not too big as in the case of S_c in Figure 5.2(a) and S_r in Figure 5.2(b), pairs of line segments indexed with the indices between row indices in $[0, k]$ and column indices in $[m, |L_c|]$ also cannot be correct matches, because the matching line segments are always detected in the same order. Remember that the index of a line segment is equal

Algorithm 5 FindMatchingLineSegments($matchingTable, scores, threshold$)

1: $matchingPairs$ = list of entries of type (l_k, l_m) where $l_k \in L_r$, $l_m \in L_c$
2: $(k, m, maximumScore) = FindMaximumScore(matchingTable, scores, matchingPairs)$
3: $bestMatchAngle = \beta_{(k,m)}$ where $l_k \in L_r, l_m \in L_c$
4: $currentAngle = bestMatchAngle$
5: **while** $maximumScore > threshold$ **do**
6: **if** $bestMatchAngle \approx currentAngle$ **then**
7: add new entry (l_k, l_m) to $matchingPairs$
8: $MarkInvalidScores(scores, k, m, matchingTable)$
9: **else**
10: **for all** $scoreID \in matchingTable(k, m)$ **do**
11: $(score, isScoreValid) = scores(scoreID)$
12: $scores(scoreID) = (score,$ **false**$)$
13: **end for**
14: **end if**
15: $(k, m, maximumScore) = FindMaximumScore(matchingTable, scores, matchingPairs)$
16: $currentAngle = \beta_{(k,m)}$ where $l_k \in L_r, l_m \in L_c$
17: **end while**
18: **return** $matchingPairs$

to its extraction number. The same is true for the indices between the row indices in $[k, |L_r|]$ and column indices in $[0, m]$. The algorithm continues to find other correct matches until the maximum score stays over a predetermined value, *threshold*. For every match which is determined to be correct, the algorithm eliminates invalid scores with the function $MarkInvalidScores$ given in Algorithm 7. At the end of Algorithm 5, the merged matching table becomes as illustrated in Table 5.2.

		L_c							
		l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
L_r	l_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	l_2	***4.82***	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	l_3	0.00	***8.08***	0.00	0.00	0.00	0.00	0.00	0.00
	l_4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	l_5	0.00	0.00	0.00	0.00	0.00	***7.12***	0.00	0.00
	l_6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	l_7	0.00	0.00	0.00	0.00	0.00	0.00	***5.33***	0.00
	l_8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 5.2: Merged matching table after running Algorithm 5 on the table. Table explicitly shows that l_1, l_2, l_6, and l_7 in L_c match with l_2, l_3, l_5, and l_7 in L_r respectively.

5.4 Finding The Next Best Match

The next best match is a pair of line segments (l_k, l_m) where $l_k \in L_r$ and $l_m \in L_c$ are given by the cell with the next highest merged score indexed by (k, m) in the corresponding matching table. In order to find the next best match, Algorithm 6 is run on the matching

table.

Algorithm 6 FindMaximumScore($matchingTable, scores, matchingPairs$)

```
1: maximumScore = 0
2: rowIndex = 0
3: columnIndex = 0
4: for i : 1 → |L_r| do
5:    for j : 1 → |L_c|  do
6:       currentScore = 0
7:       for all scoreID ∈ matchingTable(i, j) do
8:          (score, isScoreValid) = scores(scoreID)
9:          if isScoreValid then
10:            currentScore = currentScore + score
11:         end if
12:      end for
13:      if currentScore > maximumScore then
14:         isNewMaximum = true
15:         for all (l_k, l_m) ∈ matchingPairs do
16:            if i == k and j == m then
17:               isNewMaximum = false
18:            end if
19:         end for
20:         if isNewMaximum then
21:            maximumScore = currentScore
22:            rowIndex = i
23:            columnIndex = j
24:         end if
25:      end if
26:   end for
27: end for
28: return (rowIndex, columnIndex, maximumMergedScore)
```

Algorithm 6 finds the maximum merged score in the matching table. If the current merged score is higher than the previous merged score the algorithm checks whether this merged score is used before to determine a correct match, by searching the matching line segment pairs found in the previous iterations. If the current merged score imply a new match, it is considered as the current maximum score. At the end of the algorithm the maximum merged score which imply a new match is returned along with its indices in the matching table.

5.5 Eliminating Incorrect Matches

Once l_i in L_r and l_j in L_c are determined to be matching line segments, some incorrect matches can be automatically determined assuming that a line segment in L_c can only match with a single line segment in L_r. Following this assumption, given that l_i in L_r and l_j in L_c match, matches that can be determined to be incorrect are,

$$
\begin{array}{lll}
(l_k, l_m) \quad \text{where} & k = i \Leftrightarrow m \neq j, & (1) \\
& k \neq i \Leftrightarrow m = j, & (2) \\
& 1 \leq k \leq |L_r|\,, & \\
& 1 \leq m \leq |L_c|\,, & \\
& l_k \in L_r,\ l_m \in L_c. &
\end{array}
$$

Here inequalities at lines (1) and (2) represent incorrect matches with the same row or column indices (i, j) respectively, relying on the assumption that a line segment in L_c can only match with a single line segment in L_r. If l_i in L_r and l_j in L_c is determined to match, then Algorithm 7 is used to eliminate the incorrect matches.

Algorithm 7 MarkInvalidScores($scores, k, m, matchingTable$)

```
1: for i : 1 → row size of matchingTable do
2:   if i ≠ k then
3:     for all scoreID ∈ matchingTable(i, m) do
4:       (score, isScoreValid) = scores(scoreID)
5:       scores(scoreID) = (score, false)
6:     end for
7:   end if
8: end for
9: for j : 1 → column size of matchingTable do
10:  if j ≠ m then
11:    for all scoreID ∈ matchingTable(k, j) do
12:      (score, isScoreValid) = scores(scoreID)
13:      scores(scoreID) = (score, false)
14:    end for
15:  end if
16: end for
```

Algorithm 7 marks all the scores, associated to $scoreID$'s in the lists corresponding to incorrect matches, as invalid. This automatically eliminates the same invalid scores in other lists, since Algorithm 6 merge only valid scores in a list.

5.6 Determining The Pose Difference

Line segments common to two line segment sets L_c and L_r explicitly shows the rotational difference between the scans S_c and S_r. After rotating S_c onto S_r, the translational difference between S_c and S_r can be computed with respect to common edges of the common line segments.

5.6.1 Computing The Rotational Difference

It is evident that every matching line segment pair (l_k, l_m) where $l_k \in L_c$ and $l_m \in L_r$ is expected to have approximately the same rotational difference. In order to compute the rotational difference between S_c and S_r as precisely as possible, an effective way is to assign weights to the rotational differences between matching line segment pairs. The weights are determined according to the precision of the differences. The precision of the rotational difference between any (l_k, l_m) depends on their lengths because of two reasons. First, as the length of a line segments l_k and l_m fitted to range values increases, they converge to ideal l_k and l_m because of the fact that range values have a fixed amount of measurement error and the effect of error decreases as the length of line segments increases. Second, assuming that l_k has a fixed length, as the length of l_m increases, the precision of the rotational difference between itself and l_k increases because of the same fact. As a result, the precision of the rotational difference increases as $\|l_k\|$, $\|l_k\|$ and the ratio given below increases.

$$\text{Length Ratio}(l_i, l_j) = \frac{\|l_i\|}{\|l_j\|} \quad \text{where} \quad \begin{aligned} &\|l_i\| < \|l_j\|, \\ &l_i \in L_c \Leftrightarrow l_j \in L_r, \\ &l_j \in L_c \Leftrightarrow l_i \in L_r. \end{aligned}$$

Combining all properties which increase the precision of the rotational difference yields the formula,

$$\begin{aligned} \text{Precision Value}(l_i, l_j) &= \|l_i\|.\|l_j\|.\text{ Length Ratio}(l_i, l_j) \\ &= \|l_i\|.\|l_j\|.\frac{\|l_i\|}{\|l_j\|} \\ &= \|l_i\|^2. \end{aligned}$$

Using the precision values as weights, Algorithm 8 computes the rotational difference between S_c and S_r.

Algorithm 8 ComputeRotationalDifference($matchingPairs$)

1: $rotationalDifference = 0$
2: $total = 0$
3: **for** $i = 1 : |matchingPairs|$ **do**
4: $\quad (l_k, l_m) = matchingPairs(i)$
5: $\quad total = total +$ Precision Value$(l_i, l_j).\beta(l_k, l_m)$
6: **end for**
7: $rotationalDifference = total / |matchingPairs|$
8: **return** $rotationalDifference$

Algorithm 8 accumulates the rotational difference of every matching line segment pair multiplied by their weights and at the end it divides the accumulated value by the number of matching line segment pairs in order to find the rotational difference between S_c and S_r.

5.6.2 Computing The Translational Difference

After rotating S_c over S_r, the translational difference between these scans can be computed by a weighted average of the translational difference between matching edges. Since the matching line segments are known, matching edges can be found as given in Algorithm 9. Precision values introduced in Section 5.6.1 are used to assign weights to the translational differences between common edges.

Algorithm 9 ComputeTranslationalDifference($matchingPairs$)

```
1: translationalDifference = 0
2: total = 0
3: count = 0
4: for i = 1 : |matchingPairs| do
5:    (l_k, l_m) = matchingPairs(i)
6:    for j = 1 : |matchingPairs| do
7:       (l_u, l_v) = matchingPairs(j)
8:       if i ≠ j and i_(k,u) ≠ null and i_(m,v) ≠ null then
9:          precisionValue_(k,u) = Precision Value(l_k, l_u)
10:         precisionValue_(m,v) = Precision Value(l_m, l_v)
11:         if precisionValue_(k,u) < precisionValue_(m,v) then
12:            total = total + precisionValue_(k,u).(i_(k,u) − i_(m,v))
13:         else
14:            total = total + precisionValue_(m,v).(i_(k,u) − i_(m,v))
15:         end if
16:         count = count + 1
17:      end if
18:   end for
19: end for
20: translationalDifference = total/count
21: return translationalDifference
```

Algorithm 8 accumulates the translational differences between matching edges multiplied by the minimum of the precision values of the corresponding line segments forming the edges as their weights. Then it computes the translational difference between S_c and S_r by dividing the accumulated value with the number of matching edges. Translating S_c over S_r with the computed translational difference results in the merged local map given in Figure 5.3.

5.7 Algorithm Extensions

If the pose difference between S_c and S_r is small it can be assumed that the matching line segments are always detected in the same extraction order. This means that if l_i in L_c is matching with l_j in L_r, l_{i+1} in L_c can only match with a line segment l_k in L_r where $k > j$. This assumption also helps detect some additional incorrect matches. Following this assumption, given that l_i in L_r and l_j in L_c match, matches that can be determined to be incorrect are,

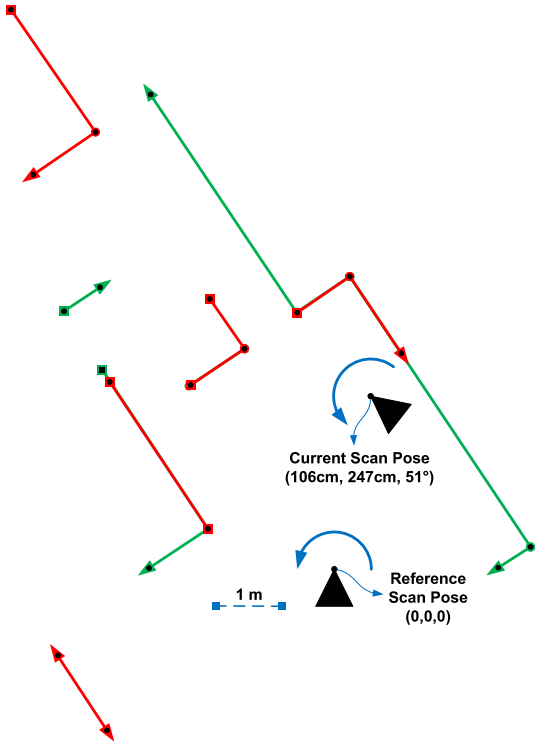


Figure 5.3: The current range scan is aligned over the reference scan resulting in a merged local map. The pose difference between the scans is $(106cm, 247cm, 51°)$ as (x, y, θ).

$$
\begin{aligned}
(l_k, l_m) \quad \text{where} \quad & k < i \Leftrightarrow m > j, && (1)\\
& k > i \Leftrightarrow m < j, && (2)\\
& 1 \leq k \leq |L_r|\,,\\
& 1 \leq m \leq |L_c|\,,\\
& l_k \in L_r,\ l_m \in L_c.
\end{aligned}
$$

Here inequalities at lines (1) and (2) represent the assumption that the matching line segments are always detected in the same extraction order. If l_i in L_r and l_j in L_c is determined to match, then Algorithm 10 is used to eliminate the incorrect matches. Note that the algorithm is called twice as,

$$
\begin{aligned}
&MarkInvalidScores(scores, i, j, matchingTable, 0, i, j, |L_c|),\\
&MarkInvalidScores(scores, i, j, matchingTable, i, |L_r|\,, 0, j)
\end{aligned}
$$

instead of a single call,

$$MarkInvalidScores(scores, i, j, matchingTable).$$

Invoking Algorithm 10 with the given parameters eliminates all incorrect matches detected following the decision that l_i in L_r and l_j in L_c match, including the ones detected by the assumption explained in Section 5.5.

Algorithm 10 MarkInvalidScores($scores, k, m, matchingTable, rStart, rEnd, cStart, cEnd$)

```
1: for i : rStart → rEnd do
2:   for j : cStart → cEnd do
3:     if i ≠ k or j ≠ m then
4:       for all scoreID ∈ matchingTable(i, j) do
5:         (score, isScoreValid) = scores(scoreID)
6:         scores(scoreID) = (score, false)
7:       end for
8:     end if
9:   end for
10: end for
```

5.8 Scan Merging and Map Construction

Aligning geometrical primitives extracted from the current scan S_c over the ones extracted from the reference scan S_r results in a new geometrical primitive set with duplicate line segments and edges. Merging common geometrical primitives after each scan alignment step produces a local map of the environment. Algorithm 11 shows how a local map of the environment can be created by merging geometrical primitives extracted from S_c to the ones in the current local map.

Algorithm 11 takes the current local map, matching line segment pairs of L_c and L_r, and L_c as arguments in order to integrate L_c into the current local map, *map*. If *map* is empty all line segments in L_c is added to *map* by assigning each of them a unique $mapID$. Otherwise, common line segments in L_c and L_r listed in $matchingPairs$ are merged into *map* by means of $mapID$s assigned to the line segments in L_r in the previous iteration of the map construction process.

It is possible that there exists line segments in L_c which have no matches in $matchingPairs$, but have matches in *map*. For this case, such matches are detected by considering the overlapping property, $o(i, k)$ and the vertical distance property, $d^v_{i,k}$ of parallel line segments introduced in Section 4.4. It is evident that if a line segment in L_c and one or more line segments in *map* match, they have to be parallel, overlapping, and has a vertical distance close to zero, since L_c is aligned over L_r which was integrated into *map* in the previous invocation of Algorithm 11. Once these line segments are detected, they are merged into a single line segment in *map*. If a line segment in L_c has no match in $matchingPairs$ or *map*, then it is added to *map* with a unique $mapID$.

Algorithm 11 UpdateMap($map, matchingPairs, L_c$)

```
currentMapID = number of line segments in map
if currentMapID == 0 then
  for all l_i ∈ L_c do
    currentMapID = currentMapID + 1
    set mapID of l_i to currentMapID
    add l_i to map
  end for
else
  for all (l_i, l_j) ∈ matchingPairs where l_i ∈ L_r and l_j ∈ L_c do
    for all l_k ∈ map do
      if mapID of l_i is equal to mapID of l_k then
        mergedLineSegment = Merge(l_j, l_k)
        remove l_k from map
        add mergedLineSegment to map
      end if
    end for
  end for
  for all l_i ∈ L_c where l_i has no match in matchingPairs do
    mapLineSegments = null
    for all l_k ∈ map do
      if o(i, k) yields overlap and d^v_{i,k} ≈ 0 then
        add l_k to mapLineSegments
      end if
    end for
    if |mapLineSegments| > 0 then
      mergedLineSegment = MergeAll(l_i, mapLineSegments)
      remove l_k from map
      add mergedLineSegment to map
    end if
  end for
  for all l_i ∈ L_c where l_i has no match in map do
    currentMapID = currentMapID + 1
    set mapID of l_i to currentMapID
  end for
end if
return map
```

Chapter 6

Experimental Results

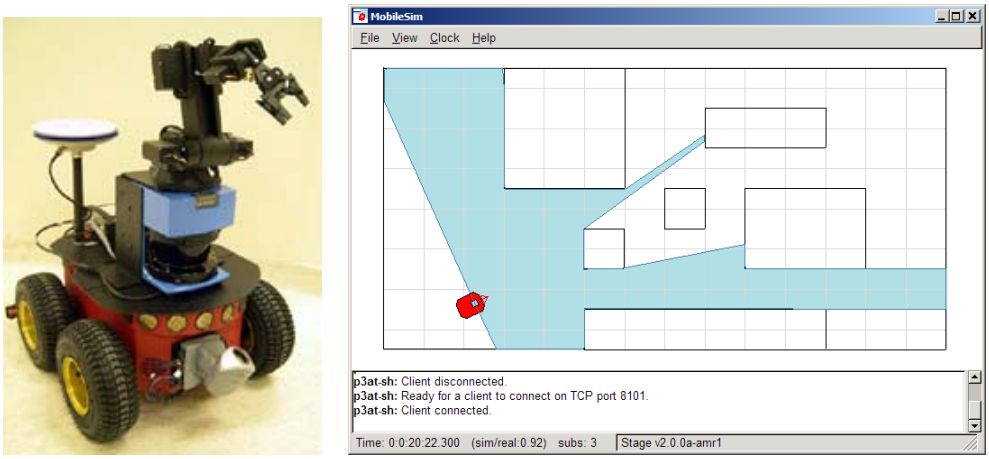
In the alignment phase of the scan matching process, S_c is aligned over S_r with respect to the pose difference computed by the scan matching algorithm. Pose error between the aligned current scan S'_c and S_r is the most important indicator of the accuracy of scan matching. The accuracy of pose information of scans recorded from pose sensors like odometry, GPS, or DGPS affects the accuracy of the estimated pose error. Even if it is easy to get perfect pose information in simulations, it is not possible to get good pose information for real scan records. In this thesis, we propose two pose independent estimation methods for pose error, called *Error Area* and *Error Area Percentage* in order to compute the pose error between S'_c and S_r.

6.1 Experimental Setup

We use the simulation environment of a Pioneer-3AT robot [24] in Figurea, called *MobileSim* which is a software for simulating mobile robots and their environments, for debugging and experimentation. *MobileSim* is based on the Stage library, created by the Player/Stage/Gazebo project [21]. As illustrated in Figureb, the robot in our simulation environment has a laser range scanner, SICK LMS200 with $\pm 2.5cm$ range and 0.1° angular error which are the same with the error parameters of a real SICK LMS200.

6.2 Pose Error

In order to investigate the pose computation accuracy of our algorithm, we created a simulated data set consisting of 3069 scans with perfect pose information recorded on the green (gray in b/w) path illustrated in Figure 6.2(a). The red (black in b/w) path in the figure is



(a) (b)

Figure 6.1: (a) Pioneer-3AT research robot and (b) its simulation environment in Stage.

the estimated path and blue line segments represent the $14m \times 7m$ map created by our scan matching algorithm without any pose information. Figure 6.2(b) illustrates a zoomed region of the same map in order to show the displacement between the real and the estimated path. Figures show that our scan matching algorithm is successful enough even for creating maps in the absence of pose information.

The experiments also show that the algorithm is fast enough for real-time pose estimation and map building, since each pair of scans is matched at 3.48ms on average which is far less that the recording time of a scan with a SICK Laser Range Scanner which is 24ms.

Figure 6.3(a) illustrates the rotational error between S_c and S_r with respect to the real rotational difference. The error is computed by the formula,

$$\begin{aligned} \textit{Rotational Error}\,(S_c, S_r) &= |(\theta'_c - \theta'_r) - (\theta_c - \theta_r)| \\ &\text{where} \quad 0^\circ \leq \theta'_c, \theta'_r, \theta_c, \theta_r < 360^\circ. \end{aligned}$$

In the formula, θ'_c and θ'_r are the estimated, and θ_c and θ_r are the real rotational values of S_c and S_r respectively.

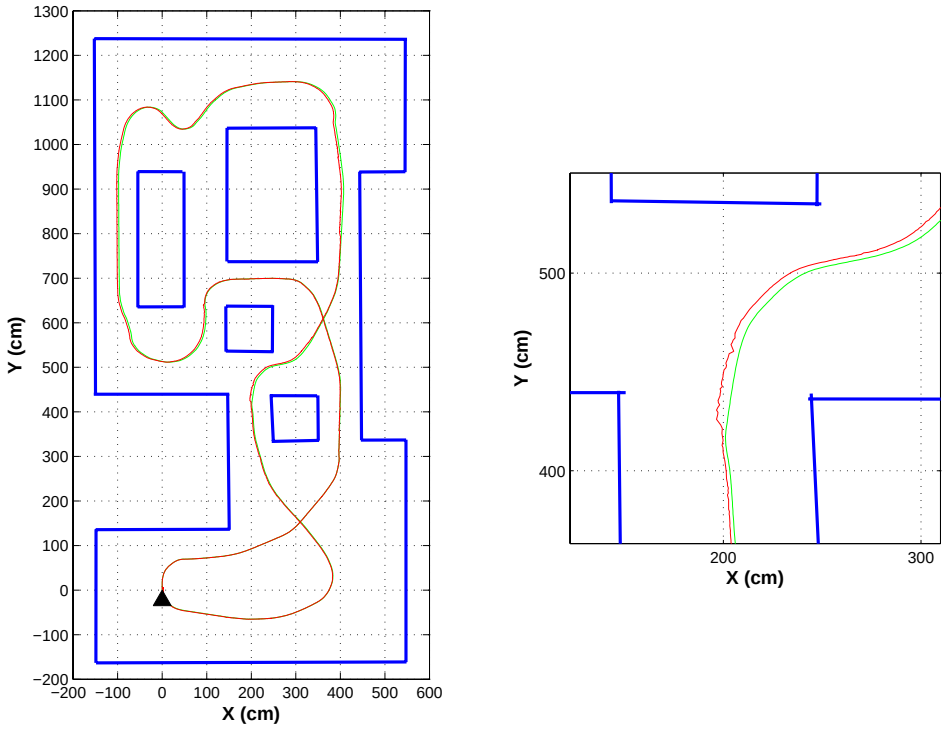
Figure 6.3(c) illustrates the global rotational error of S_c which is the rotational difference between the real and the estimated pose of S_c. The error is computed by the formula,

$$\begin{aligned} \textit{Global Rotational Error}\,(S_c) &= |\theta'_c - \theta_c| \\ &\text{where} \quad 0^\circ \leq \theta'_c, \theta_c < 360^\circ. \end{aligned}$$

In the formula, θ'_c is the estimated, and θ_c is the real rotational value of S_c.

Figure 6.3(b) illustrates the translational error between S_c and S_r with respect to the real translational difference. The error is computed by the formula,

$$\textit{Translational Error}\,(S_c, S_r) = \sqrt{((x'_c - x'_r) - (x_c - x_r))^2 + ((y'_c - y'_r) - (y_c - y_r))^2}.$$



(a) (b)

Figure 6.2: (a) Map created from 3069 scans. Distance traveled: 43.19m. Average processing time: 3.48 ms for matching two scans. (b) Green path stands for the real path traversed by the robot. Red path is determined by scan matching.

In the formula, x'_c, x'_r, y'_c, and y'_r are the estimated, and x_c, x_r, y_c, and y_r are real coordinates of S_c and S_r respectively.

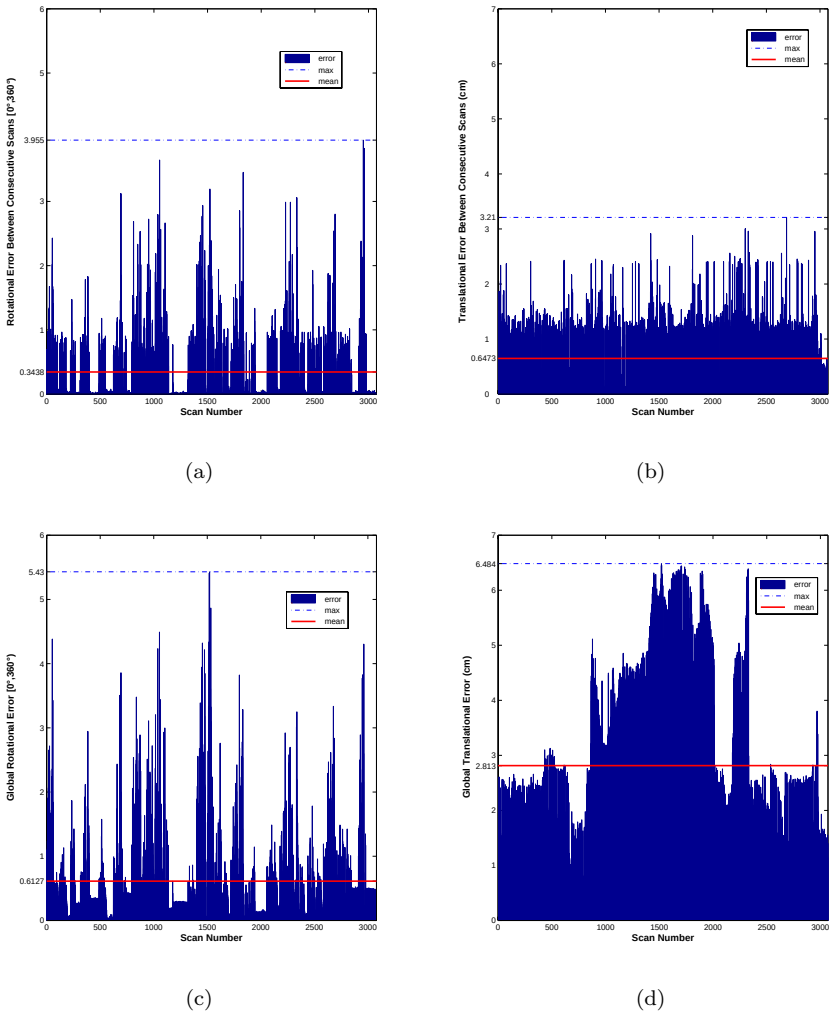
Figure 6.3(c) illustrates the global translational error of S_c which is the translational difference between the real and the estimated pose of S_c. The error is computed by the formula,

$$\textit{Global Translational Error}\,(S_c) = \sqrt{(x'_c - x_c)^2 + (y'_c - y_c)^2}.$$

In the formula, (x'_c, y'_c) is the estimated, and (x_c, y_c) is the real coordinate of S_c.

Since our scan matching algorithm does not require pose information, it is possible to match scans with very large pose differences. Repeating the experiment above just with 23 scans results in an estimated path and map illustrated in Figure 6.4(a). Note that some line segments are missing in the constructed map since the range sensor did not get a chance to observe them as a result of the large displacement between consecutive scans.

The results of the experiment are illustrated in Figures 6.5(a), 6.5(b), 6.5(c), 6.5(d).



(a) (b)

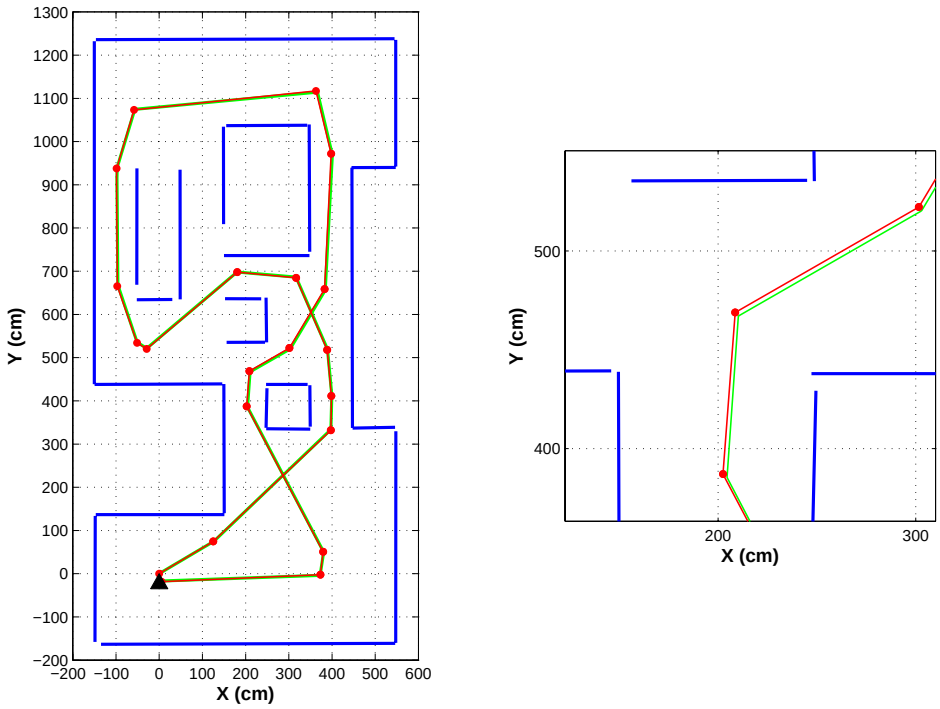
(c) (d)

Figure 6.3: (a) Rotational and (b) translational error between consecutive scan pairs (S_i, S_{i+1}) where $0 \leq i < 3069$ and (c) global rotational and (d) translational error of each scan S_i where $0 < i \leq 3069$.

6.3 Error Area and Error Area Percentage

Literature on scan matching does not provide any techniques to understand the accuracy of a match if pose information associated with scans does not exist. We suggest the computation of error area and error area percentage as effective techniques to estimate the correctness of matching two scans. *Error Area* is the sum of areas between every matching pair of line segments l_i and l_k extracted from S_c and S_r respectively, with respect to the current robot pose, as illustrated in Figure 6.6. Computing error area indicates the accuracy of matching S_c to S_r in terms of both translation and rotation.

When we consider the local map created by matching S_c and S_r as a *traversability map* consisting of two states {*Traversable, Not traversable*}, the error area indicates the



(a) (b)

Figure 6.4: (a) Map created from 23 scans. Some line segments are missing in the map because they could not be sensed due to high pose difference between scans. (b) Green path stands for the real path traversed by the robot. Red path is determined by scan matching.

misclassified area. For example, in Figure 6.6, error area is the sum of areas A_1, A_2, A_3, and A_4 as a result of misclassifying A_1 and A_3 as *Not traversable* which were classified as *Traversable*, and misclassifying A_2 and A_4 as *Traversable* which were classified as *Not traversable* by S_r. The area which is classified the same for both S_c and S_r is called the *Common Area*.

Another indicator of the accuracy of scan matching is the percentage of error area. *Error Area Percentage* is the percentage of the error area to the total area of matching line segments which is the sum of the error area and the common area. For the case in Figure 6.6,

$$Error\ Area\ Percentage = \frac{Error\ Area}{Error\ Area + Common\ Area} \times 100$$

$$\text{where} \quad Error\ Area = A_1 + A_2 + A_3 + A_4.$$

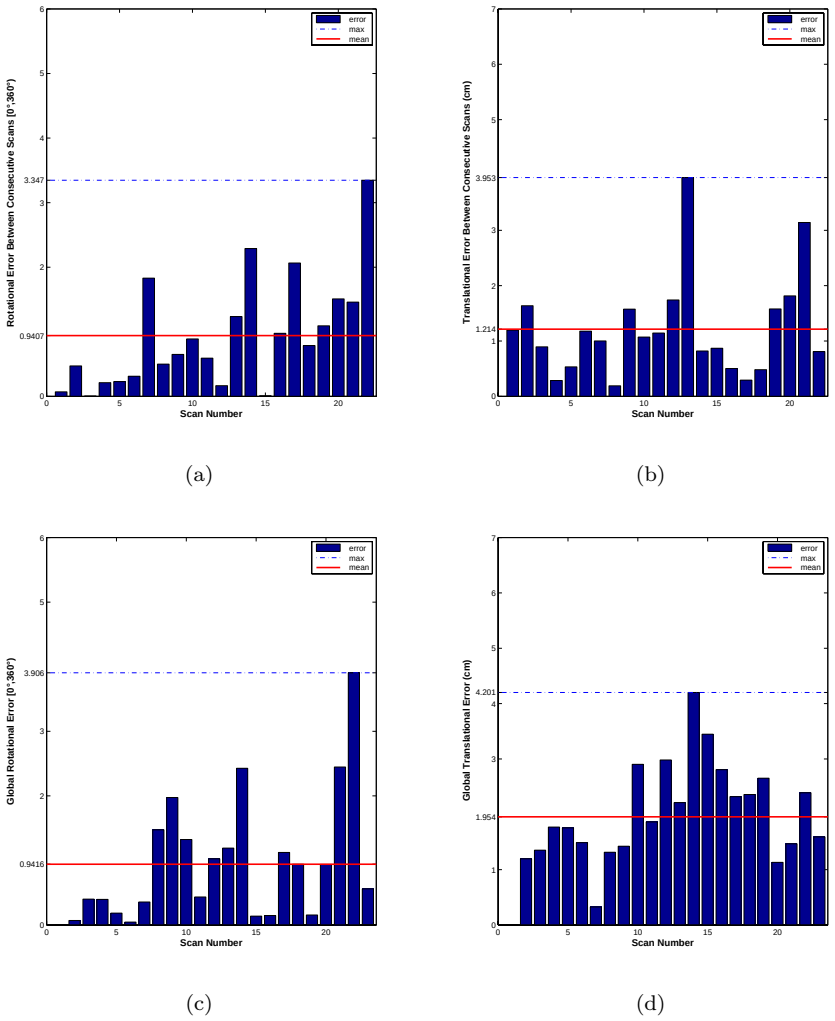


Figure 6.5: (a) Rotational and (b) translational error between consecutive scan pairs (S_i, S_{i+1}) where $0 \leq i < 23$ and (c) global rotational and (d) translational error of each scan S_i where $0 < i \leq 23$.

6.3.1 The Relationship between Pose Error and Error Area

In order to compute the real pose error between S'_c and S_r, the true pose difference between these scans should be known. Even if it is possible to get the real pose difference in simulations, for real scan records, real pose difference can only be approximated. In case that S_c and S_r are recorded without any pose information, it is not possible to get an idea about the pose error between S'_c and S_r.

In the absence of absolute pose information, error area is an effective indicator of the pose error between S'_c and S_r, since error area is proportional to pose error for acceptable rotational and transitional errors between S'_c and S_r. In order to illustrate the relationship between error area and pose error, we match the scan in Figure 6.7 with its exact copy, since

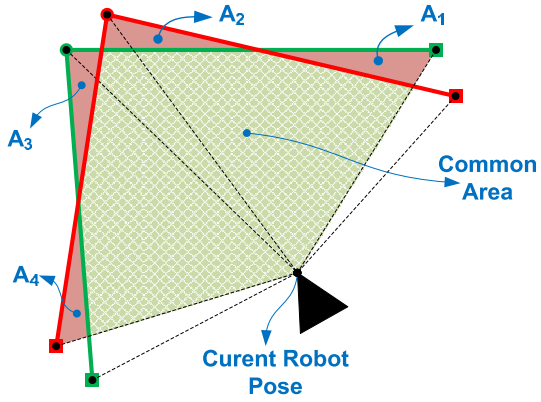


Figure 6.6: The sum of A_1, A_2, A_3, and A_4 is the error area between two scans. S'_c misclassifies A_1 and A_3 as *Not traversable* which were classified as *Traversable*, and misclassifies A_2 and A_4 as *Traversable* which were classified as *Not traversable* by S_r.

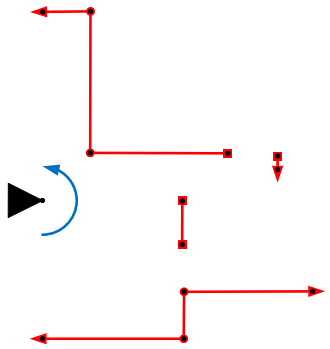
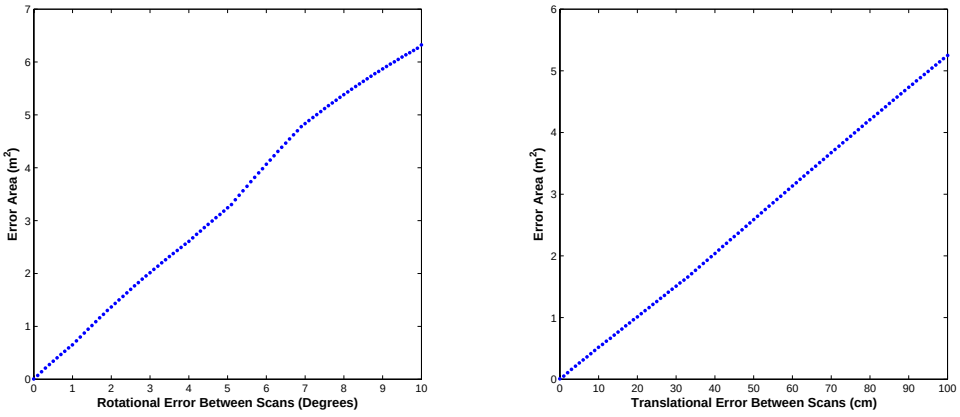


Figure 6.7: Scan used for investigating rotational and translational error on error area.

such a match does not cause a pose error. Injecting incremental rotational and translational error into the alignment phase of S_c over S_r results in an increase in error area as illustrated in Figures 6.8(a) and 6.8(b). It is evident that, after the alignment phase all matching line segments pairs of S'_c and S_r should be parallel. Our algorithm checks parallelism within an error window of 5° for rotation and $10cm$ for translation which are small portions of error ranges given in the figures. As a result, error area can be used in order to estimate the magnitude of the pose error between S'_c and S_r if the pose information of S_c and S_r does not exist.

6.3.2 The Relationship between Translational Difference and Error Area

Scan matching without pose information has the capacity to match scans with large pose differences. In order to see whether translational difference between S_c and S_r affects error area between S'_c and S_r, an experiment was conducted in the simulation environment illustrated in Figure 6.9. In this experiment, the robot moves directly to the intersection of



(a) (b)

Figure 6.8: (a) The effect of rotational (b) translational error on error area.

walls at a constant velocity, traveling 18.65m and recording 1480 scans. There is only translational differences between scans because rotational movements at a constant translational difference does not affect the results as long as the same linear structures in the environment are sensed by the sensor. In order to see the effect of real translational difference on error area, scans are recorded along with perfect pose information. However, this information was not used to align S_c over S_r, that is, pose error between matched scans was not eliminated.

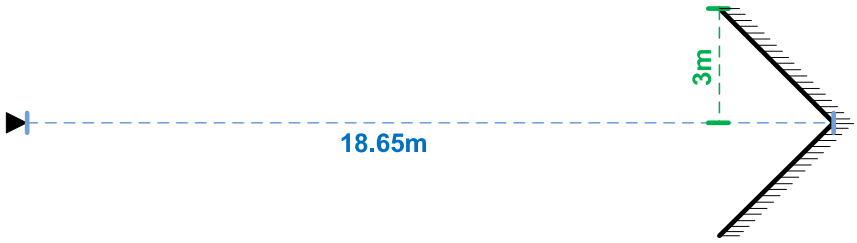
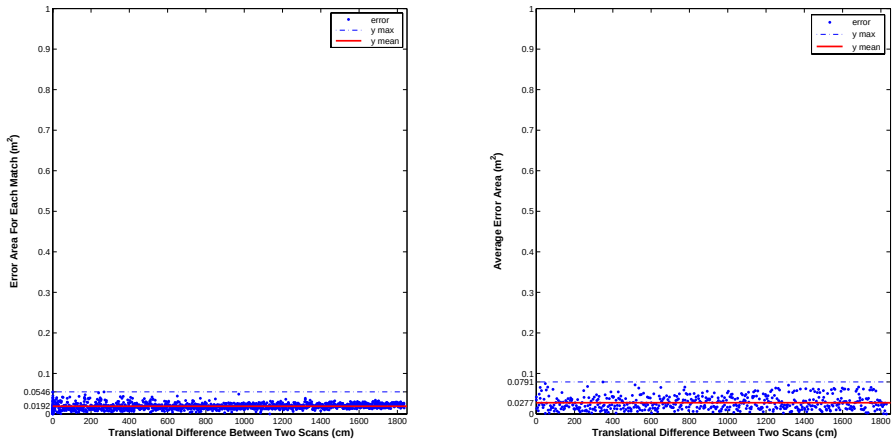


Figure 6.9: Experimental simulation environment for investigating the relationship between translational difference and error area.

We conducted two different experiments on the recorded data. In the first experiment, we matched S_0 to every other scan S_i where $0 < i \leq 1480$. Figure 6.10(a) illustrates the relationship between error area and real translational difference between S_c and S_r. As can be noticed in the figure, translational difference between scans has no effect on error area. In the second experiment, we matched all scan pairs (S_i, S_j) where $0 \leq i < j \leq 1480$ and compute an average error area for each match within the same translational difference range which is $1cm$ for this experiment. As illustrated in Figure 6.10(b), translational difference between scans has no effect on error area again. However, the upper bound and the mean is larger than the ones in Figure 6.10(a) implying that range values in S_0 is more accurate than the average.

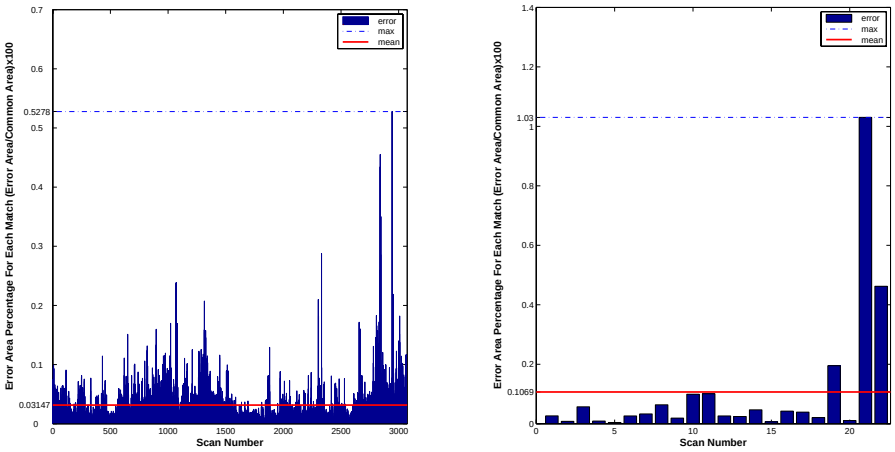
Error area percentages for 3069 and 23 scans are illustrated in Figure 6.11(a) and 6.11(b).



(a) (b)

Figure 6.10: (a) Error area and (b) average error area with respect to the translational difference between two scans.

Note that errors given in the figures are similar.



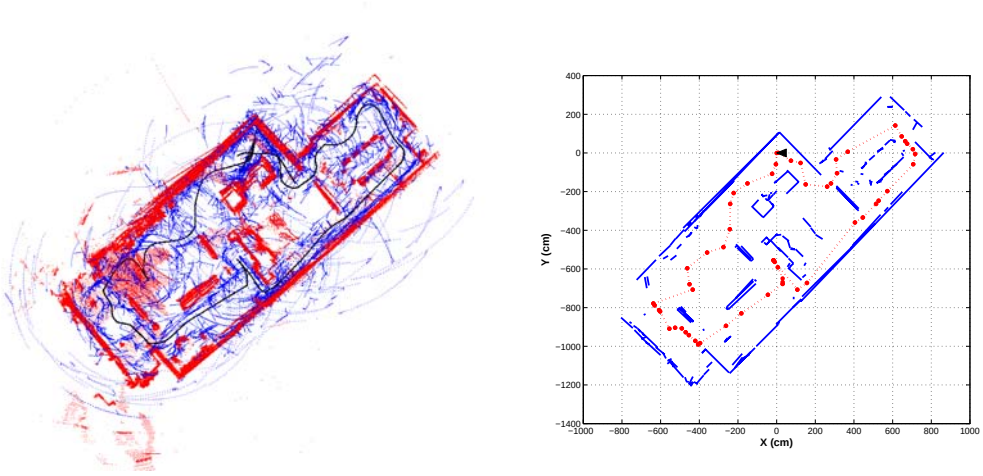
(a) (b)

Figure 6.11: (a) Error area percentage for 3069 and (b) 23 scans.

6.4 Matching Real Scans

Figure 6.12(b) illustrates the map of a home taken from Radish repository [13]. Raw sensor readings at are given in Figure 6.12(a).

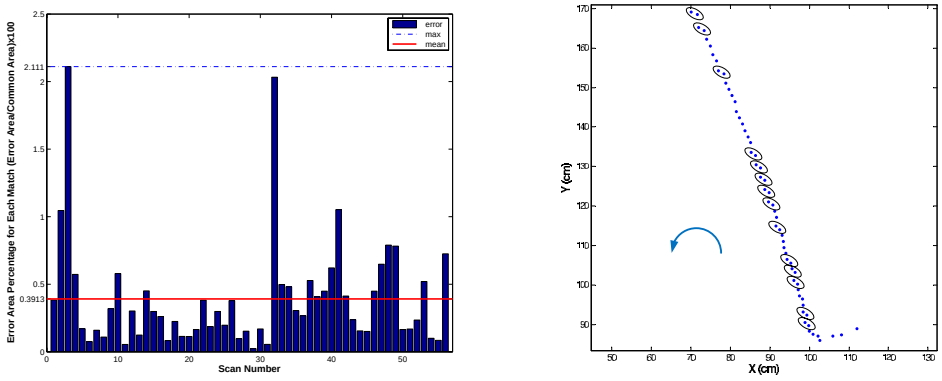
Error area percentage is illustrated in Figure 6.13(a). When the robot makes a rotational



(a) (b)

Figure 6.12: (a) Real LADAR data taken from Radish repository. (b) Map created by our algorithm. Translational error is $(7.84cm, -0.66cm)$ at (x, y) axis and rotational error is $2.12°$.

movement while the LADAR is in the range-scanning phase, linear structures are sensed distorted as illustrated in Figure 6.13(b) resulting in higher error area percentage values which means higher pose errors.



(a) (b)

Figure 6.13: (a) Error area percentage for real LADAR data. (b) Linear structures in the environment are sensed distorted because of the rotational movement of the robot in the counterclockwise direction.

Chapter 7

Conclusion and Future Work

In this thesis, we presented a new scan matching method based on matching line segments of two scans recorded at two different locations without any pose information. Two line segment sets corresponding to current and reference scan are matched by comparing geometric relationships derived from line segments within the same line segment sets. By applying a scoring algorithm, these geometrical relationships are scored in a matching table and line segments are matched according to these scores. Our method is able to match scans with very large displacements.

We can extend our scan matching algorithm to perform global scan matching, map building, place recognition, loop closing and multirobot mapping very easily because it does not rely on an initial pose estimate and because it employs simple geometrical relationships between line segments in order to find pose difference between scans. We do not require an *a priori* map for pose difference computation. We can match a scan directly to another scan. Therefore our method can be used for exploration and map building in unknown indoor environments.

All scan matching methods which use line segment based representation of scans do not have any extendibility in terms of using geometrical primitives other than line segments. Contrary to these methods, our method has the potential of using different types of geometrical primitives. For instance, geometrical primitives such as circles and arcs can easily be integrated into our method as long as simple geometrical relations can be defined such as circle center to line segment distance.

Bibliography

[1] Francesco Amigoni, Simone Gasparini, and Maria Gini. Map building without odometry information. In *Proceedings of the IEEE International Conference on Robotics and Automation ICRA 2004*, pages 3753–3758. IEEE Press, 2004.

[2] Francesco Amigoni, Simone Gasparini, and Maria Gini. Building segment-based maps without pose information. *Proceedings of the IEEE*, 94(7):1340–1359, 2006.

[3] Francesco Amigoni, Simone Gasparini, and Maria L. Gini. Scan matching without odometry information. In Helder Arajo, Alves Vieira, Jos Braz, Bruno Encarnao, and Marina Carvalho, editors, *ICINCO (2)*, pages 349–352. INSTICC Press, 2004.

[4] III Andrew J. Martignoni and William D. Smart. Localizing while mapping: a segment approach. In *Eighteenth national conference on Artificial intelligence*, pages 959–960, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[5] Tim Bailey, Eduardo Mario Nebot, Julio Rosenblatt, and Hugh F. Durrant-Whyte. Data association for mobile robot navigation: A graph theoretic approach. In *ICRA*, pages 2512–2517, 2000.

[6] O. Bengtsson and A. J. Baerveldt. Matching of laser range-finder scans in changing environments. In *Proc. of the European Workshop onAdvanced Mobile Robots*, pages 103–109, 1999.

[7] J. Borenstein, H. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. In *Journal of Robotic Systems*, volume 14(4), pages 231–249, 1997.

[8] I. J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *Robotics and Automation, IEEE Transactions on*, 7(2):193–204, 1991.

[9] J. Gomez E. Zalama, G. Candela and S. Thrun. Concurrent mapping and localization for mobile robots with segmented local maps. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2001.

[10] T. Einsele. Real-time self-localization in unknown indoor environments using a panorama laser range finder, 1997.

[11] S. Engelson and D. McDermott. Error correction in mobile robot map learning. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 3, page 25552560, 1992.

[12] J.-S. Gutmann and C. Schlegel. Amos: comparison of scan matching approaches for self-localization in indoor environments. In *Proc. Euromicro Workshop Advanced Mobile Robots*, pages 61–67, 1996.

[13] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), http://radish.sourceforge.net, 2003.

[14] Ewald von Puttkamer Joachim Weber, Klaus-Werner Jrg. Apr - global scan matching using anchor point relationships. In *The 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, 2000.

[15] L. J. Latecki, R. Lakaemper, X. Sun, and D. Wolter. Building polygonal maps from laser range data. In *Proc. Int. Cognitive Robotics Workshop*, pages 56–62, 2004.

[16] Longin Jan Latecki and Rolf Lakamper. Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190, 2000.

[17] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *CVPR94*, pages 935–938, 1994.

[18] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *J. Intell. Robotics Syst.*, 18(3):249–275, 1997.

[19] Viet Nguyen, Stefan Gchter, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. *Auton. Robots*, 23(2):97–111, 2007.

[20] T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE Trans. Comput.*, 23(8):860–870, 1974.

[21] Player/Stage/Gazebo project. Stage library, http://playerstage.sourceforge.net, 2008.

[22] T. Rfer. Building consistent laser scan maps. In *Proceedings of the 4th European Workshop on Advanced Mobile Robots (Eurobot 2001)*, volume 86, page 8390, 2001.

[23] T. Rfer. Using histogram correlation to create consistent laser scan maps. In *Proceedings of the IEEE International Conference on Robotics Systems (IROS-2002)*, pages 625–630. EPFL; Lausanne, Switzerland, 2002.

[24] Mobile Robots. Pioneer 3at research robot, http://www.mobilerobots.com, 2008.

[25] Rainer Trieb Thomas Edlinger, Ewald von Puttkamer. Accurate position estimation for an autonomous mobile robot fusing shaft encoder values and laser range data. In *IARP - International Advanced Robotics Programme ; 2nd Workshop on Sensor Fusion and Environmental Modelling*, Session 5B, Oxford, England, September 2-5 1991.

[26] Masahiro Tomono. A scan matching method using euclidean invariant signature for global localization and map building. In *ICRA*, pages 866–871. IEEE, 2004.

[27] B. Tovar, R. Murrieta-Cid, and C. Esteves. Robot motion planning for map building. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[28] G. Weiss, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation, IROS*, pages 12–16, 1994.

[29] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Computational Science & Engineering*, 4(4):10–21, 1997.

[30] Xiang Zhiyu Xu Zezhong, Liu Jilin. Scan matching based on cls relationships. In *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, volume 1, pages 99– 104, 2003.

[31] Li Zhang and Bijoy K. Ghosh. Line segment based map building and localization using 2d laser rangefinder. In *ICRA*, pages 2538–2543, 2000.