BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

VECTORIZED POINT CLOUDS FOR MOBILE ROBOTICS

VEKTORIZOVANÁ MRAČNA BODŮ PRO MOBILNÍ ROBOTIKU

DOCTORAL THESIS
DIZERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE

Ing. Aleš Jelínek

SUPERVISOR
ŠKOLITEL

prof. Ing. Luděk Žalud, Ph.D.

BRNO 2017

CONTENTS

1 INTRODUCTION

Simultaneous localization and mapping (SLAM) is a vital part of a robot's artificial intelligence (AI). Any mobile robot, which is expected to operate like a rational agent, needs a set of algorithms allowing it to explore, memorize and make assumptions about the surrounding environment. SLAM is a great challenge in robotic science, because despite a large progress in the last thirty years, there are still many opened problems and the theme remains (and surely will remain in the near future) an active subject of research.

As will be shown in the following chapter, mathematical description of the fundamentals of SLAM is already well elaborated. What lacks is a more abstract way of knowledge representation starting with more complex geometrical shapes for object description and leading towards a semantic map containing not only metric information, but also the meaning of the mapped objects. AI research is proceeding in this direction and the following text is meant to support this effort.

This thesis is focused on extraction of geometrical information from the raw point clouds and presents good reasons to use them as a base building block of the future SLAM algorithms in contrast with today's trend to incorporate point-like features only. Mathematical treatment of these objects is inevitably more complicated than in the case of points, but the gains stemming from generalization, noise suppression and data size reduction seem to outweigh these issues. The following chapters describe data processing from acquisition to feature matching and scan registration algorithms. Line segments are used as a higher-dimensional representation of metric data. The back-end algorithms for map maintenance and loop closing are not part of this work.

Except the theoretical development of novel algorithms, there was also some engineering effort involved in the work on this thesis. A decommissioned omnidirectional robot Hermes was rebuilt afresh and a new distributed control system was developed for it. Hermes was later used to acquire data for practical examination of the algorithms mentioned above and became part of the ATEROS robotic system built and maintained by our research group. The acronym ATEROS stands for *Autonomous Telepresence Robotic System*. The key feature is the combination of teleoperation performed by a human operator and autonomous functions performed by the robot itself. This way, the mission of the robot can be carefully supervised and guided from the control center and, either in case of emergency (e.g. lost signal), or automated routine (e.g. convoying), the autonomous functions are ready to help. The possible missions are: autonomous 3D map building, environment contamination measurement, urban search and rescue and other military and civilian applications.

2 STATE OF THE ART OF SIMULTANEOUS LOCALISATION AND MAPPING

This chapter summarizes classical approaches as well as the most recent research, discusses a biological context of the problem and presents the motivation for the approach employed in the rest of this work.

2.1 A brief history of SLAM

1980s are usually considered to be a decade, in which the first serious SLAM algorithms have appeared. The premier published concept was an *occupancy grid* (OG) [17]. The method partitions a continuous space into a matrix of discrete squares and checks the probability of an obstacle being present at each cell. New measurements are incorporated using a Bayesian rule and the OG converges to an authentic representation of reality. Significant drawback is the necessity of an exact information about robot's pose, which means that the OG approach does not deal with the SLAM problem in its full complexity.

A different approach presented at that time was a *topological map* [24]. In contrast to OG, the map representation is continuous and contains coordinates of the important places in the environment and possible routes between them. There is no or very little detail about the surrounding obstacles, because the net of important locations is usually sparse and, by definition, describes only the free space.

Soon after these approaches were introduced, the first probabilistic algorithms have appeared [23]. Metric information became an essential building block of the map and topological description switched from places of interest to landmarks, aka significant features of obstacles. Bayesian rule formed a theoretical background of the probabilistic approach, which allowed to deal with the inevitable uncertainty of every measurement. The technique later evolved into three distinct directions: Kalman filters, Expectation maximization and Particle filters.

Kalman filter (KF) have been developed and used long before the SLAM problem became an issue and its applications felt mainly into the field of accurate pose estimation for navigational purposes [13]. Pose estimation is a low-dimensional task, where only a small amount of sensory data is fused, but a theoretical dimensionality of problems solved using KF is not bounded. On the other hand, computational complexity of canonical KF is not favourable and limits its practical applicability significantly. Under certain circumstances, this can be bypassed by introduction of the *information filter* with a sparse information matrix. Since SLAM is a non-linear problem, many modifications for that purpose are used: *Extended Kalman filter*, *Unscented Kalman filter* and many others. KF is a very popular technique in the SLAM community.

Expectation maximization (EM) technique stems from maximum likelihood estimation [16]. This process estimates parameters of a model with respect to the given observations (or measurements in general), iteratively maximizing the likelihood of the model to provide those data. A big advantage of this method is its ability to effectively work with correspondences in subsequent data sets. Unfortunately, the iterative nature of the algorithm and necessity to repeatedly traverse the observations makes it too slow for real-time applications and its main domain of operation is off-line processing. This is the reason, why the expectation maximization method is usually combined with another algorithm addressing the localization part of the SLAM and EM is used only for local map building [3].

Particle filter (PF) [5] approach has a strong connection to the Kalman filter and *genetic algorithms* (GA). The KF heritage is the Bayesian probability approach to estimation, while GA provide a Monte-Carlo-based framework for evaluation, selection and generation of multiple statistical models. A big advantage of PF is the possibility to model non-linearities and effective maintenance of multiple hypothesis, which is highly beneficial for loop-closing. Rao-Blacwellized version of PF was introduced to overcome the growth of the computational complexity with growing state space, but the promising results were found to be unstable after an extensive period of operation. The main area of PF application is therefore pose estimation, where the dimensionality is limited and the multi-hypothesis approach is highly favourable.

Methods originating from the seminal concepts described above dominate the SLAM scene from the date of appearance up to the time, when this thesis is written and probably will remain important, at least in the near future. Over the last two decades, SLAM have surely became an important and popular research field and the results achieved are unexceptionable. When reviewing the most recent papers, a shift in the direction of research can be spotted. The above mentioned methods clearly provide a strong tools for SLAM solving, but when dealing with complexity of a real world, their applicability seems still quite limited. The gap between theoretical principles and everyday practice brought up a wide variety of new problems waiting to be solved.

In 2001 an important paper [6] with a very promising title *"A solution to the simultaneous localization and map building (SLAM) problem"* was published. For the first time in a history of SLAM, there was a mathematical proof, that the effort of simultaneous localization and map building has a solution. The theory seemed to be mostly finished and a great algorithms were about to be published, but a bare existence of the theoretical solution did not implied a straightforward practical results. Some bottlenecks (nicely surveyed for example in [4]) are as follows:

Computational complexity (CC) is a vital attribute of any algorithm intended to operate on a variably sized set of input data. SLAM faces the inevitable growth of the map, so the CC is crucial for a fluent operation. There is not yet an evidence, that the SLAM problem has some lower bound on a computational complexity, however complexities of the state of the art approaches lead to rather pessimistic expectations.

Feature correspondence, similarly to CC, is also known to be essential from the very beginning of the SLAM research until now [4]. Orientation with respect to the surrounding environment requires extraction of significant landmarks and their recognition from different points of view. Successful correspondence search is usually a demanding operation, so performance optimization is still important, even though the search space is usually limited due to physical constrains on magnitude of robot's movement.

In case of *loop-closing*, the robot visits a place it already was before. Such a situation requires a feature correspondence search across a large area of the map making the previous problem unconstrained and possibly highly computationally demanding.

Convergence was already mentioned many times. In general, a gradual exploration of the environment (growing amount of information), should lead to a more accurate map (landmark position uncertainty should, in limit, close to zero) [6]. This behaviour is critical for long term operation of the robot.

Stability and robustness refers to ability of the algorithm to operate as expected under full range of specified conditions. A small percentage of false correspondences is only a part of this topic. Other influences are e.g. sensor noise, physical accidents and numerical stability of the algorithms.

Dynamic environment brings a whole new set of complications for the SLAM solving algorithms, because the map is not static in time any more. Introduction of other dynamic objects (walking people, moving robots, etc.) brings a temporal dimension to the map and renders the localization part of SLAM to become a subset of mapping.

Structured environment is a double-edged basis for a successful SLAM. On one hand, the environment has to have some structure, so that features can be extracted and a map built. On the other hand, too dissected structure leads to mismatches during correspondence search and highly complicated maps flooded with details.

Map representation and semantics is critical, because practical applications require interaction with objects usually containing many features, complex shape, texture and *meaning*. Though point-like features are usually deemed well explored and shape and texture are intensively studied, the abstract semantics is still at the very beginning.

2.2 Biological context

So far, only the artificial ways of localization and mapping were discussed, but there is also another domain, far from engineering and related subjects, where the SLAM problem is already solved, admittedly in much more functional, efficient and elegant way. Any organism on our planet interacts somehow with its surroundings. Many of them are limited to the nearest neighbourhood, because of a stationary way of life, but many others have developed some kind of locomotion mechanism, which allows them to travel to different locations. Deliberate movement to places of interest is the logical next step and the evolutionary optimization process have infallibly found a method of how to implement it into the growing brains of living creatures.

Biological SLAM is surprisingly efficient. For example a honeybee, which is quite smart insect species, has only approximately one million of neurons in a one cubic millimetre sized brain. Even with that "computational power" it can memorize position of its home apiary, locations of the blooming flowers and communicate this knowledge to the other members of the hive. All of these tasks prove honeybees ability to successfully solve the SLAM problem.

Human localization and mapping is obviously far more advanced. We attend to much more things in our surroundings and have more experiences giving them wider context. For robotics research, the most inspiring are the recent results, revealing hierarchical structure of the spatial memory and the landmark - layout dualism of the environment representation in a long term memory.

2.3 Motivation for the vectorized approach

Artificial SLAM developed in laboratories is built in a bottom-up manner, starting at the base principles and as more and more obstacles on the way towards the final goal appear, the solutions presented grow in complexity. Biological research of SLAM is similar to reverse-engineering and provides us with great inspiration and an etalon of successful "implementation". As illustrated by the brief literature exploration above, it seems, that both approaches lead to complex features, possibly objects, to be detected in the sensory measurement. These should be further incorporated into a semantic map, which would contain geometrical information as well as other possible relationships. This goal is obviously far away for today's AI and too broad for a single thesis to explore. Further work is therefore focused on geometrical information only.

Object representation requires a dense description of its shape, not a sparse sampling of some selected points with no information (or at least assumption) about the whole surface of the body. Dense representation is mainly achieved using raw point clouds and boundary description. The raw point cloud approach saves us from any kind of advanced processing, but the memory footprint is large, computational costs are high (but easily parallelizable in the most cases) and no separation of objects in the scene is achieved. Boundary description consists of more complicated geometrical objects such as line segments, arcs and splines in two dimensions and planes or bezier patches in 3D. Extraction of these primitives leads to data size reduction of several orders of magnitude, noise suppression and interpolation of the point-like measurements, creating an abstraction of a general shape of the scene. Object separation is possible, but geometry alone is usually not sufficient. Price for these features is higher complexity of the algorithms for segmentation, extraction and matching of the given primitives. For future SLAM, the boundary description is clearly more appealing and will be studied in the following pages of this work. Content of this thesis is focused on point cloud processing, vectorized scan matching and correspondence search, which is sometimes called a SLAM front-end [4]. The back-end mainly consists of the data fusion inference mechanism and will not be covered. Since raw measurement processing and information extraction is the current bottleneck of the SLAM algorithms, focus on the front-end part addresses an opened research problem definitely worth the attention.

3 CONCEPT OF THE WORK

In Figure 3.1 is a scheme of the operations of the presented SLAM front-end. The algorithms between data acquisition and output correspondences and pose were all devised and implemented by the author of this thesis and will be described in detail in the following text. Since these algorithms are mostly studied separately, each chapter starts with a review of the literature and further sections follow a structure of a regular research paper as well.

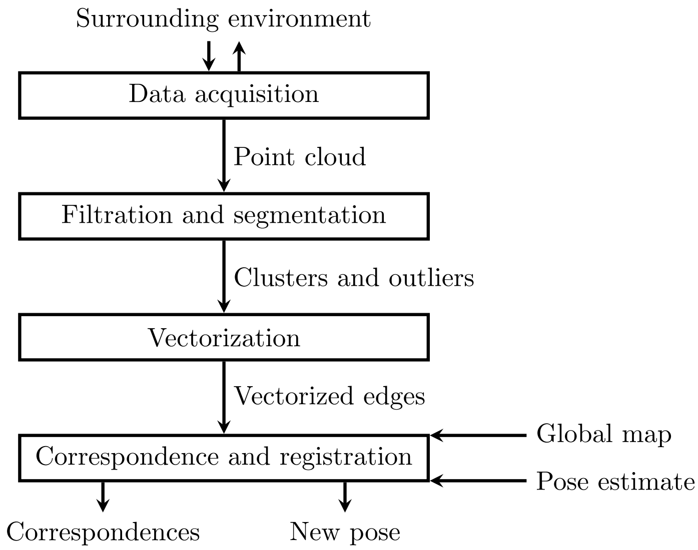


Fig. 3.1: Schematic workflow of the data processing in the vector-based SLAM front-end.

Data acquisition is an essential step in every SLAM algorithm. Quality and quantity of the input data directly determine nature of the subsequent algorithms and achievable outcomes. Regardless of the underlying technology, all data should be converted into the distance measurement and presented as a point cloud transformed in local coordinates of the robot.

Filtration and segmentation is used for removal of the outliers and split of the whole cloud in places, where some discontinuities occur. Outlier filtration is important, because highly flawed measurements can, through the statistical processing, spoil the entire scan. Splitting at discontinuities is intended to separate parts of the point cloud, which could potentially form a continuous edge in the real world.

Vectorization is an operation converting a point cloud (or a continuous cluster in this case) into a set of line segments, which well describe the shape of the point distribution. This process usually involves a large reduction in a memory footprint of the data, because dense point clouds of thousands of points can be often approximated with only a few lines.

Correspondence search and scan registration is the last step of the SLAM front-end. Line segments extracted previously are compared to a region of a global map and correspondences between them are looked for. Once the correspondences are known, the scan is registered to the map and a new pose estimate is calculated. Both correspondences and the new pose are base components for any following SLAM back-end.

4 TECHNICAL MEANS FOR EXPERIMENTATION

Real-world experiments are the final proof of function of every algorithm, theory, or scientific concept in general. Since robotics is a highly practical research area, engineering of a robot for testing is an important part of development.

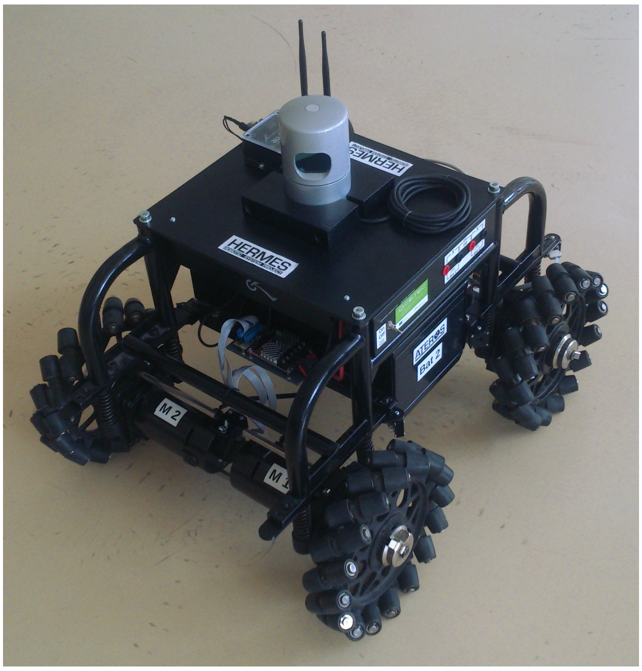


Fig. 4.1: The omnidirectional robot Hermes.

4.1 Hardware design

The requirements put on the robot were the following: medium size (so it can easily pass through a doorway), payload circa 15 kg, good manoeuvrability, on-board computer with reasonable compute power and connectivity and at least one hour of operational time.

An omnidirectional wheel frame was chosen as the best of the alternatives. It can follow any trajectory possible in the 2D plane, which provides excellent manoeuvrability in the tight indoor environment. As the device turned into the robot Hermes was previously an electric wheelchair, the payload condition was exceeded several times and dimensions were suitable for the movement inside of the buildings as well. Instead of a seat for the disabled person, which was originally mounted on the top of the metal frame, there is now rectangular box for the electronic parts. At the very top of the robot, a mount for the laser scanner is present, as can be clearly seen on the overall photograph in Figure 4.1.

Like majority of other mobile robots for laboratory purposes, Hermes is powered by a set of electrical accumulators. The electronics works either with 5 V or 12 V input, which is provided by a self made, dual channel switched-mode power supply. Motors are regulated using the RoboClaw ST 2x45A controllers, each taking care of two motors with quadrature encoders.

As an embedded platform for communication and computation a BananaPi R1 routerboard was selected. It contains a decent dual-core ARM processor and 1 GB of RAM, which should be enough for basic operation. For communication, there are five gigabit Ethernet ports and wireless network adapter, all connected with the processor through a dedicated switching chip, resulting in high throughput of the communication lines.

Sensory equipment is probably the most important hardware for SLAM. This work is focused on point cloud processing, therefore the robot Hermes is equipped with a laser scanner Velodyne HDL-32 by Velodyne LiDAR Inc. It can be clearly seen at the photograph of the robot in Figure 4.1. The scanner is based on the time of flight principle and 700 000 range measurements per second. It is important to note, that the laser scanner provides all necessary information to form and ordered point cloud from the output data. This will be extensively exploited in the Chapter 6.

4.2 Software tools

The first project developed for the purposes of this thesis was a control system for the robot Hermes. The architecture of the control system was made reasonably flexible. The key concept is separation of tasks into a set of standalone programs (either GUI or background running daemon processes), which communicate through predefined communication channels. This allows to run different parts of the system on different machines, communicate using various hardware interfaces and implementations in several programming languages. Every communication channel is composed of two layers. The *interface* part represents a wrapper containing the necessary settings (e.g. IP address and port number) and unifying the input/output behaviour for various hardware interfaces and corresponding low-level protocols. The channel *protocol* is the second layer, which describes format of the packets and encodes/decodes the data. The interface and protocol classes are designed to be freely combinable with each other. The actual commands, responses, etc. are defined separately for each particular standalone process. The distributed control system was used in other experiments performed by our laboratory as well, for example in precise georeferencing [8].

A vector processing library is the second important bundle of software made during the work on this thesis, because it contains implementations of the theoretical methods described further in the text. Similar to the distributed control system, it is written in C++, but with an emphasis on speed of operation. The library is built on templates and does not take advantage of polymorphism, runtime type information and other programming tools, which would impose unwanted time penalty to evaluation of the algorithms. Especially the simplest classes would be significantly afflicted by this effect. To keep the examination honest, there are not any platform specific optimizations, such as *single instruction - multiple data* paralleling and so on. In a production-grade code, these optimisations should be definitely present.

5 POINT CLOUD SEGMENTATION AND FILTRATION

Since the data are obtained in an arbitrary environment and the noise is present, filtering and segmentation have to be performed prior to vectorization. Proper preparation identifies all outlying points in a scan and determines dense clusters of points, which could potentially form an edge of a real object. Outliers form only very small clusters, which are usually discarded, since no meaningful edge can be extracted from them. Large clusters are forwarded to the subsequent vectorization algorithm.

5.1 Literature overview

Segmentation and clustering of point clouds is an active research area today, but the utter majority of effort is directed towards 3D point clouds, not the 2D case, which is in question. Survey [19] provides a useful taxonomy of segmentation methods available.

Region growing approaches are based on selection of one or more points (seeds), which are considered members of future clusters. Using a bottom-up approach, the seed starts to expand to the points in the neighbourhood and, as long as a condition is met, the cluster grows. Iterative nature makes these algorithms rather unpredictable. If working on ordered data sets, the processing speed of every iteration is relatively fast, but the speed of convergence is heavily dependent on the structure of the point cloud.

Machine learning provides many good ways for data segmentation. Probably the most important are the modified K-means methods. Another possibility is hierarchical clustering, which is similar to region growing, but the initial seed distribution stems from different principles.

Many other interesting approaches exist, but these are focused on 3D clouds as well and propose elaborated algorithms, which could not be justified in a much simpler 2D case.

5.2 Segmentation of the ordered point clouds

A brief literature exploration in the previous section brought two important insights. First, if the points in a cloud have certain order, the algorithm can exploit it and proceed faster. Second, outliers can easily spoil the object fitting process. On this basis, an algorithm, independent on line fitting and working sequentially on an ordered point cloud, was devised.

The diagram of the algorithm function is depicted in Fig. 5.1. At input, there is a raw cloud of N points, the distance threshold t is a maximal spacing between points belonging to a single cluster and the constant K, which sets the number of neighbouring points being examined during the proximity test, which allows rejection of solitary outlying points without breaking a continuity of the otherwise compact cluster.

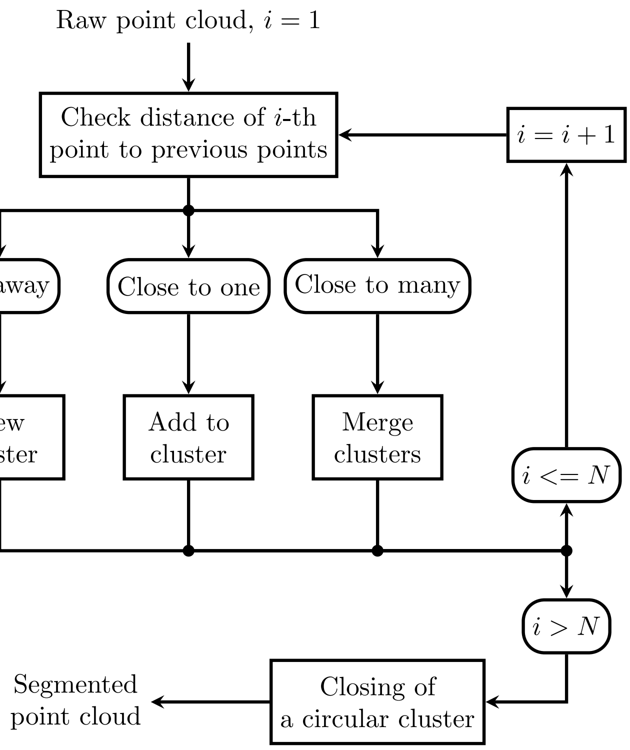


Fig. 5.1: Schematic diagram of the segmentation algorithm.

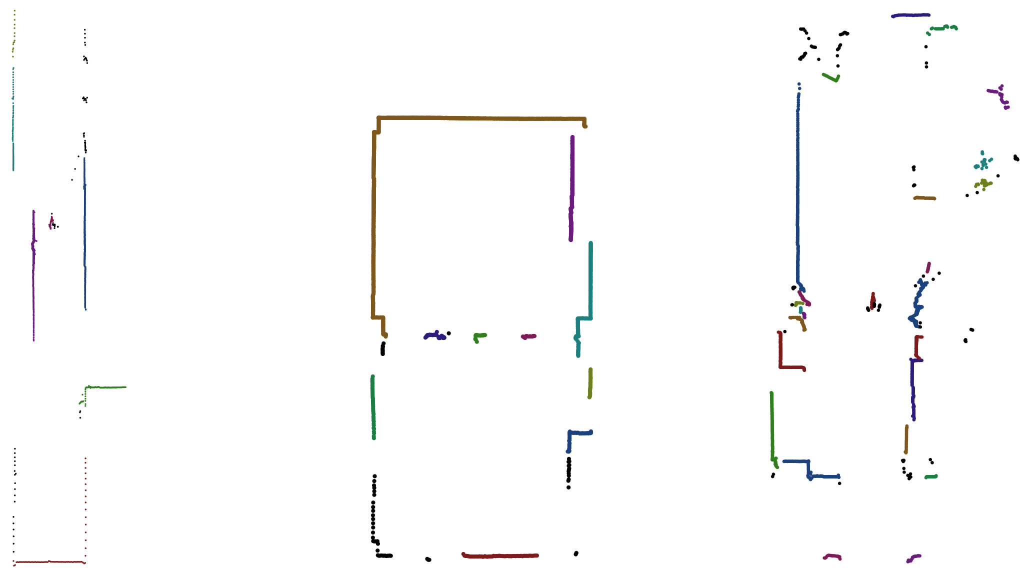
The algorithm proceeds from one point in an ordered point cloud to another and checks if K of the previous points is closer or farther than the distance t. If K is larger than i, it stops at $i = 1$. After the comparison finishes, three alternatives may occur as demonstrated in Fig. 5.1. First, an i-th point might be too distant from all of the K previous points, which leads to establishment of a new cluster. Second, if the working point is close enough to some of the previous points and all these points belong to the same cluster, the working point is added to their parent cluster. The third case arises, when the working point can belong to more than one cluster. In such a situation, all of the given clusters and the working point are joined together and form a new bigger cluster unifying all of them. Then the algorithm proceeds to the next point and the process continues in the next iteration. Once the last point is processed, there is a final check, whether the cloud is circular and the points from the beginning and the end belong to the same cluster.

The output of the algorithm is a set of clusters with varying number of points. Clusters having a size below a certain value are denoted as outliers and are removed from the scan. The rest contains continuous chains of points with guaranteed maximal distance between them.

5.3 Experimental verification

A lot of scans from real environments as well as simulations were examined during the development. Here is only a representative part illustrating its main features. All experiments were processed in Cartesian coordinates using the same parameter settings: $K = 10$, t was adaptively scaled ($t_l = 5$cm and $t_h = 20$cm) and the clusters were considered outlying if they contained less than fifteen points.

The set of experiments depicted in Fig. 5.2 was held in an indoor environment, using the equipment described in Chapter 4. The narrow corridor in Fig 5.2a is ideally suited for testing of the algorithm's ability to deal with the spreading points in a row. Near the scanner, where the measurement density is reasonable, the clusters are continuous and father away, they fall apart into solitary outliers. Figures 5.2b and 5.2c come from an empty room and a highly structured laboratory. The apparent continuous clusters are well distinguished and the cluttered debris forms either small clusters or merely the outliers.



(a) Corridor. (b) Empty room. (c) Structured laboratory.

Fig. 5.2: Three scans from the real indoor environment. Clusters are depicted in colors, while outliers are jointly rendered in black.

5.4 Résumé of the segmentation algorithm

The algorithm is well applicable in regular situations and an overall robustness in scenarios similar to real-life scans is high enough, so the algorithm can be safely used in practise. Experimental verification in the previous section showed good results in situations with proper algorithm settings. Outlier rejection was sufficient. Since it was developed as a minimal working preprocessor for the subsequent operations in the vector-based pipeline in Fig. 3.1, there are some additional issues that would have to be handled in a production-grade implementation, namely the sampling density should be addressed in a more elaborated way. Otherwise, to the best knowledge of its author, the algorithm is reliable and stable solution for the 2D point cloud segmentation.

6 VECTORIZATION OF THE POINT-LIKE DATA SETS

This chapter covers results already published in several papers [9], [12] and [10] written by the author of this thesis on the subject. The vectorization algorithm is treated as a whole, with all optimizations and augmentations, which have been gradually devised to address weaknesses of the prior attempts. The following text directly draws from these publications and any changes made apply to merge and refinement of the documents rather than adding new facts.

6.1 Introduction to vectorization

The vectorization of point sets is a technique frequently used in many areas. A typical application is point cloud processing in robotics, polyline simplification in cartography and edge extraction in image processing. General point clouds are bare sets of points without any further structure, but the laser scans, polylines in maps and skeletonized boundaries in image processing share one attribute of great importance: the data in these applications can often be considered a linked list of points with a known order, jointly describing a continual edge. This assumption is not fulfilled in all practical cases, but a wide range of practical applications either directly satisfy this condition or can be converted into a form which does so.

General vectorization algorithms are capable of processing any input data, regardless these data are ordered or not. A widely used algorithm of this type is the *Hough transform* (HT), which finds application in image processing and, occasionally, robotics. The *RANSAC* method is another example of the general approach and is used in all the disciplines mentioned above. It is a nondeterministic method and does not guarantee any exactly predictable results. Many more algorithms can be found in this category, but they are generally computationally intensive and, as such, rather unsuitable for real-time vectorization [20].

Point eliminating (PE) algorithms need ordered input data to work correctly and are mainly used in cartography to simplify maps with a high scale. The basic idea behind these algorithms lies in an approximation of a set of consecutive points by one line with some error metrics, describing the quality of that approximation. If a predefined error threshold is exceeded, the last acceptable approximation is used and a new iteration starts. The downside of these algorithms is that the approximating line is defined by only a small number of approximated points (most often only two), which leads to a loss of information from all the other points, which were discarded. Examples among many others are the *Douglas-Peucker algorithm* (DP) [7] and *Reumann-Witkam algorithm* (RW) [21], which are very fast in comparison to the general algorithms above.

Total least squares vectorization (TLS) algorithms overcome the problem of losing information from discarded points. This method also needs ordered data on input. The most common case is an orthogonal least squares regression, as is the case of the *incremental algorithm* (INC) described in [20]. The algorithm iterates through the ordered data and computes a linear approximation of the tested range of points. If the error metric is in the limit, the algorithm continues. If the error metric is exceeded, the algorithm saves the last valid line and starts a new one. Computational complexity is $O(n)$, but each iteration needs a larger amount of computation compared to the PE algorithms. On the other hand, linear regression approximates the given points much better.

PE and TLS algorithms exploit the continuity of the given point cloud, which is prone to give suboptimal results due to the inbuilt threshold. The algorithms iterate through the points, compare the error function to the threshold and once it is exceeded, a new approximation is started. This means, that the room for an error given by the value of the threshold is always fully utilized and the approximation of the cluster is globally suboptimal.

According to the facts discussed above, there are basically two categories of algorithms suitable for real-time operation: point eliminating methods, which are fast and not so accurate, and linear regression methods, which provide the best approximation, but are generally slower. Even worse, both of these approaches employ threshold for termination of a single approximation line, which leads to suboptimal results. Methods for unordered sets of points may provide globally optimal results, but their performance is orders of magnitude worse than in case of PE and TLS methods.

The rest of this chapter is focused on a novel method for fast total least squares vectorization, exhibiting the speed of the PE methods, while producing TLS results. Further augmentation for globally optimal results is presented as well. The augmented version of the algorithm provides globally optimal results at slightly increased computational costs.

6.2 Theoretical background of the total least squares fitting techniques

For the algorithm described in the next section, we need to derive an alternative form of the orthogonal regression, which will contain only terms with a simple sum. The following equation was used for a regression line description $ax + by + c = 0$. Since each line has an infinite number of possible normal vectors, an additional condition ensuring its unit length is used: $a^2 + b^2 = 1$.

The squared distance $l_i^2(a,c)$ between an arbitrary point $P_i(x_i, y_i)$ and the regression line is:

$$l_i^2(a,c) = \left(ax_i + \sqrt{1-a^2}y_i + c\right)^2. \tag{6.1}$$

Particular coefficients for the best fitting line are the solution of the following set of equations:

$$\sum_{i=1}^{N} \frac{\partial l_i^2(a,c)}{\partial a} = 0, \qquad \sum_{i=1}^{N} \frac{\partial l_i^2(a,c)}{\partial c} = 0. \tag{6.2}$$

which leads to the equation:

$$n = \frac{p}{q} = \frac{a\sqrt{1-a^2}}{1-2a^2} = \frac{\sum\limits_{i=1}^{N} x_i \sum\limits_{i=1}^{N} y_i - N \sum\limits_{i=1}^{N} x_i y_i}{N \sum\limits_{i=1}^{N} x_i^2 - N \sum\limits_{i=1}^{N} y_i^2 - \left(\sum\limits_{i=1}^{N} x_i\right)^2 + \left(\sum\limits_{i=1}^{N} y_i\right)^2}, \tag{6.3}$$

where n is the whole fraction, p is the numerator and q the denominator. Parameters of the regression line a, b and c are then as follows:

$$a = \pm\sqrt{\frac{1}{2} \pm \frac{1}{2\sqrt{4n^2+1}}}, \qquad b = \pm\sqrt{1-a^2}, \qquad c = -\frac{1}{N}\left(a\sum_{i=1}^{N} x_i + b\sum_{i=1}^{N} y_i\right). \tag{6.4}$$

During the vectorization process, some metric of the quality of approximation is necessary. When using the least squares fitting method based on statistical properties of the approximated set of points, the variance of distances between points and the line is a natural choice of such metrics. Based on the previous results, the variance is given by the following formula:

$$\sigma^2 = \frac{1}{N}\left(a^2\sum_{i=1}^{N} x_i^2 + 2ab\sum_{i=1}^{N} x_i y_i + b^2\sum_{i=1}^{N} y_i^2\right) - c^2. \tag{6.5}$$

6.3 The FTLS vectorization algorithm

This section introduces the optimization for vectorization, using the total least squares method. The process has three consecutive stages, which will be discussed separately.

Preprocessing phase consists of augmentation of the initial data pairs in the format $(x_i, y_i), i = 1 \dots N$, according to the formula:

$$\boldsymbol{a}_i = \left[x_i, y_i, \sum_{k=1}^{i} x_k, \sum_{k=1}^{i} y_k, \sum_{k=1}^{i} x_k^2, \sum_{k=1}^{i} y_k^2, \sum_{k=1}^{i} x_k y_k\right], i = 1 \dots N. \tag{6.6}$$

The computational complexity of the preprocessing stage of the algorithm is inevitably linearly dependent on the number of points N, because if all points in a cluster should contribute to the approximation, each of them needs to be used at least once.

The line fitting process is driven by the user defined threshold σ_{th}, which determines the maximal permissible standard deviation of points along the line. The algorithm is described by the diagram in Fig. 6.1.

There are three control variables in the algorithm. j and k demarcate the interval in the list of vectors $\boldsymbol{a}_i$, which is currently being examined, and s serves to adjust this range by changing the value of k in the following parts of the algorithm.

The main loop repeats until the end of the list (cluster) is reached. The repeat condition is not used to control execution of the program, instead the cycle is exited once the control variable k equals N, which means that all the points have been processed.

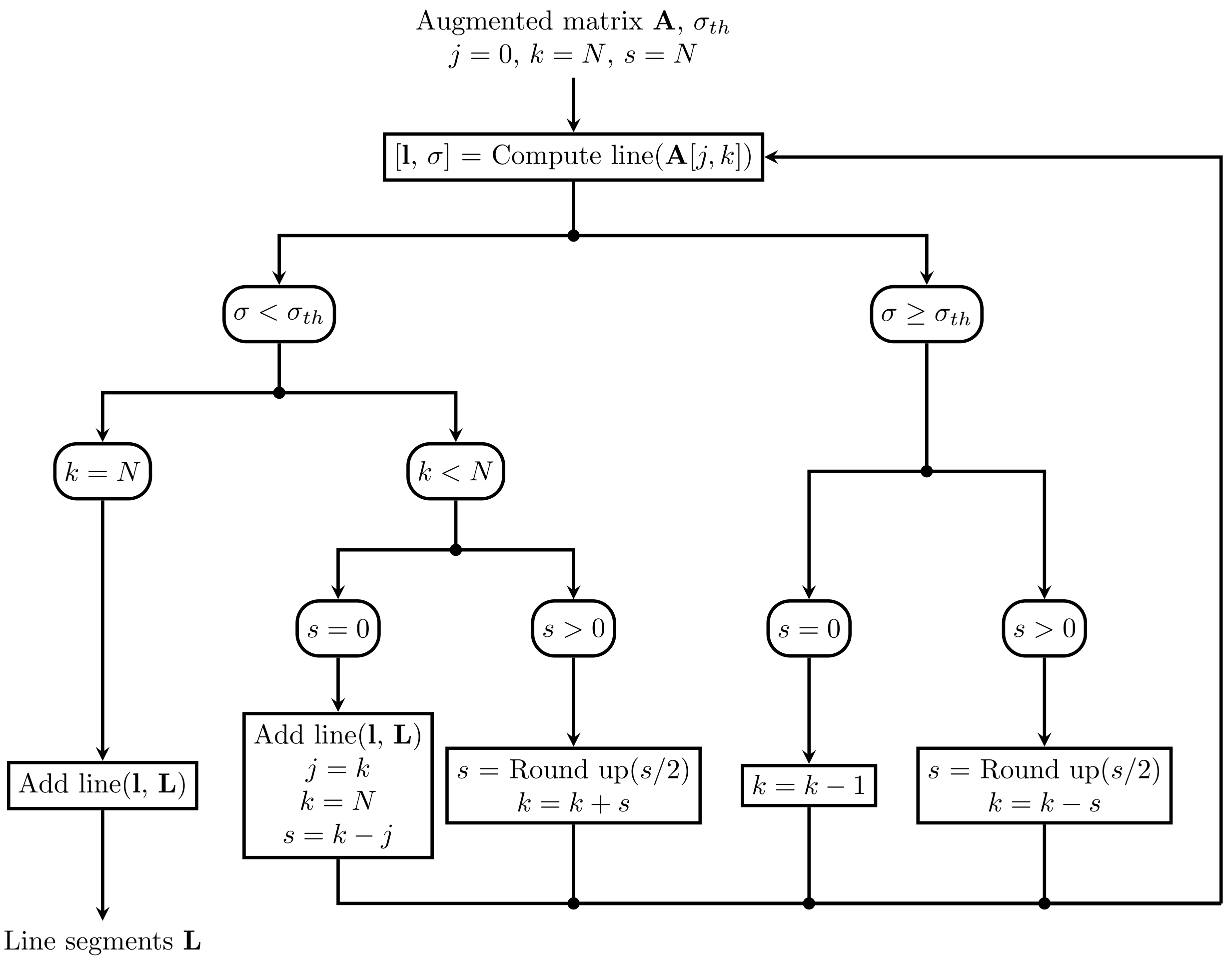


Fig. 6.1: Diagram of the line fitting stage of the FTLS vectorization algorithm. $\mathbf{A}$ - set of vectors $\boldsymbol{a}_i$, $\mathbf{L}$ - set of extracted lines.

Every iteration starts with computation of an approximating line for the interval of points $(j, k]$. Thanks to precomputation, it is a one-step operation, regardless of the number of points within the tested range. Sums are computed as a simple difference between Σx, Σy, Σx^2, Σy^2, Σxy, belonging to the k-th point, and appropriate sums belonging to the j-th point. Using the results, linear regression is computed and a vector of the coefficients $\mathbf{l} = [a\ b\ c]^T$ describing the line and a standard deviation σ of the points around it are stored.

The following decision structure implements the binary search method using the variables j, k and s and is able to locate the points, where one approximation should be switched to another. The computational complexity of this part of the algorithm is $O(m \log N)$, where m is the number of lines found. Logarithmic dependence on number of points N is achieved thanks to the precomputation and binary search approach. For a large N and small m, this approach is much faster than the linear algorithms.

Postprocessing stage truncates the lines extracted in the previous step and returns them as an output and the algorithm. If needed, the lines might be converted into a single polyline, approximating the given cluster. The beginning and the end of the polyline are simply found as perpendicular projections of the first and the last point in the cluster to the appropriate lines. Intersections of consecutive lines are used to identify the rest of the control points of the polyline.

When approximating a curved point cloud, where no significant corners are present, the intersection method of control points identification may sometimes fail. This happens if the transition from one line to another lies near an inflection point of the curvature. Good strategy is to add a third line between the two and bridge the inflection point by it, which solves the problem.

The average computational complexity of this stage of the vectorization algorithm is close to $O(m)$. In most cases, no corrections are necessary and each line is processed only once, but the number of corrections depends on the input data and cannot be easily predicted.

6.4 Augmentation for the globally optimal approximations

The FTLS algorithm outputs an ordered set of extracted lines, which approximates the whole cluster being processed, but the approximation suffers the threshold related suboptimality as described in Section 6.1. For this reason, an augmentation procedure effectively solving this issue is presented. In the following text, the augmented version of the algorithm is going to be referenced by the AFTLS abbreviation.

Unfortunately, the analytic solution to the globally optimal result search is extremely complicated or even impossible, because the problem is highly non-linear. It also does not grow monotonically, which possibly leads to an unpredictable number of local minima. To overcome this issue, the *Nelder-Mead method* (NM) [18] was employed to manipulate the position of the break points and find their globally optimal location. NM method is a non-gradient optimization technique, which relies only on enumeration of the optimized function for various input arguments. This is a key feature, because thanks to the precomputed sums of the FTLS algorithm, the evaluation of the error function is very fast.

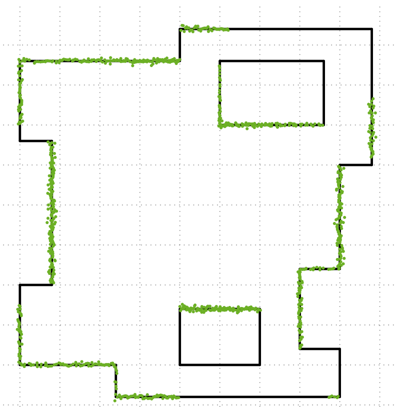
There are two major differences to the original approach [18]. The first is the introduction of constraints to otherwise unconstrained method and the discreteness of the arguments of the optimized function, but both were appropriately treated in the implementation. The concept ensures, that no worse result than the initial guess can appear and in the vast majority of cases a significant improvement can be observed.

6.5 Testing and experiments

Both FTLS and AFTLS algorithms were deeply tested in [12] and [10], there are only a single representative example of comparison between DP, RW, INC, FTLS and AFTLS algorithms.

The simulation process generates point clouds with all major properties of the real measurements, therefore filtering and clustering have to be performed prior to vectorization. Proper preparation removes all unnecessary points from a scan and determines dense clusters of points, which could potentially form an edge of a real object. The input data were processed using the algorithm described in Chapter 5. The error area in precision benchmarks is the *discrepancy* metric described in Section 7.4.

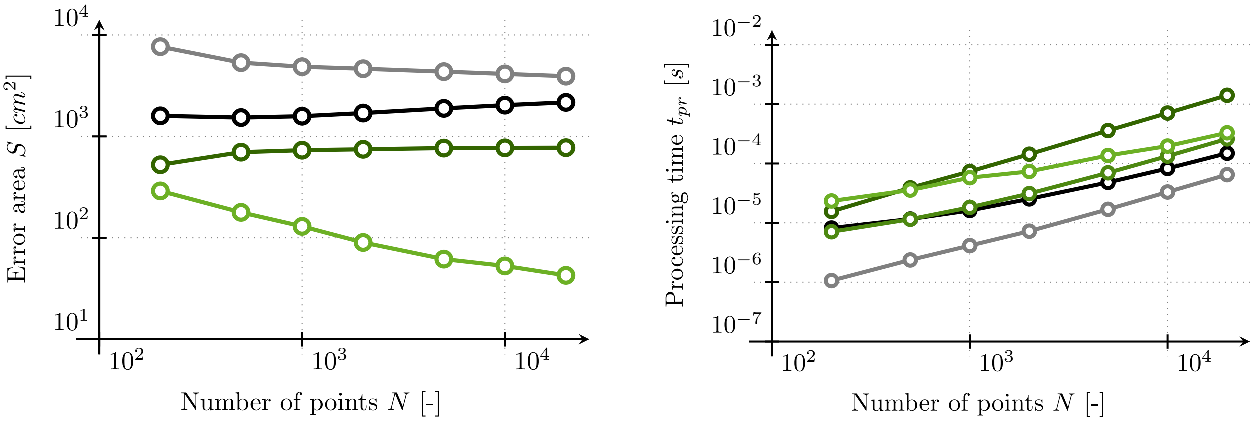
The synthetic test in Figure 6.2 mimics a real indoor environment without small objects. The line extraction success rate is good for the TLS and the DP methods, only in rare occasions an error occurs. The RW algorithm produces an excessive amount of additional lines. The precision benchmark illustrates TLS superiority over the PE methods and AFTLS refinement abilities for large clusters. The FTLS and INC characteristics does not decrease, because of



(a) The point cloud. (Grid 50 x 50 cm)

No. of points	No. of lines	Extracted lines [%] TLS	DP	RW
200	15	+3,8	-0,3	+39,9
500	15	+1,7	+0,1	+76,2
1000	15	+0,7	+0,2	+93,8
2000	15	+0,1	+0,1	+101,9
5000	15	0	+0,1	+108,9
10000	15	0	+0,1	+111,7
20000	15	0	0	+114,1

(b) The relative line extraction success-rate.



(c) The precision benchmark. FTLS and INC share the same characteristics.

(d) The speed benchmark.

Fig. 6.2: The third experiment corresponds to a clean, real environment. There are not any small objects, but the general shape with several corners and partially visible edges illustrates usual indoor object layout. (AFTLS - light green, FTLS - green, INC - dark green, DP - black, RW - gray)

the threshold imposed error, AFTLS is not influenced. The speed benchmark illustrates linear complexities of the INC and RW algorithms and good performance of the FTLS algorithm. The AFTLS costs are more significant and settle down on linear growth at higher cluster densities, but its performance for larger point clouds is still much better than of the original INC algorithm.

6.6 Recapitulation of the FTLS and AFTLS vectorization methods

A novel approach to compute a total least squares approximation of an ordered point cloud - the FTLS (Fast Total Least Squares) algorithm was presented and an augmentation (AFTLS) to deal with the threshold problem in vectorization was shown as well. The algorithms were tested in many experiments with the following results:

The synthetic tests revealed three facts: Firstly, the point eliminating methods are proven to yield less accurate results than the TLS methods. The error area is several times larger for the DP algorithm and even higher for the RW algorithm. Secondly, the INC algorithm is always slower than the FTLS algorithm and most often than the AFTLS algorithm as well, which means that the FTLS algorithm is probably the fastest way to vectorize an ordered point cloud using the TLS approximation, with speed comparable the the state of the art point eliminating algorithms. Finally, the AFTLS augmentation is proved to successfully solve the threshold problem described in Section 6.1 and provides gradually improving results as the number of available points grows.

The point clouds obtained in a man-made environment are generally quite similar, and would not provide as comprehensive overview of the scalability of the algorithms as the synthetic tests do. On the other hand, the real scans are often more cluttered and exhibit some effects, which are hard to simulate. Despite those secondary issues, the results prove theoretical expectations and synthetic tests to be correct.

The point clouds from an indoor environment are usually composed of a set of polylines; thus, the use of line extracting algorithms is beneficial for the given purpose. The described results enable the (A)FTLS algorithm to be used in online mapping systems, where the speed and quality of approximation are critical factors. The best performance (in comparison with the other algorithms) was observed on the largest point clouds, which is a promising perspective for the future, because laser proximity scanners represent a rapidly developing technology and more output points are generally expectable.

7 VECTOR MAP SIMILARITY AND REGISTRATION

General registration of various sets of measurements is a frequent task in many fields involving data processing, because in many situations only a partial observation is possible and a fusion of the incomplete information into a compact aggregate is therefore necessary. Partial observability problem is ubiquitous in mobile robotics, thus thinking over the methods not satisfying this requirement would be pointless. The same applies to the distinction between rigid and non-rigid transformation between the registered shapes. Robotic mapping generally assumes existence of a partially static environment at least, so only a rigid transformation between observations will be considered.

7.1 Feature matching and shape registration techniques

Iterative closest point (ICP) [2] is the most important method dominating the field. It is well elaborated, applicable in both 2D and 3D spaces and according to [2], it is able to work with point sets, line segment sets, implicit curves, parametric curves, triangle sets, implicit surfaces and parametric surfaces, which covers most of the practical situations. The authors also provide a proof of the convergence towards a local minimum. Today, the original ICP has dozens of variants, suggesting, that there are still issues to be addressed and the operation is not always perfect.

Many alternatives to the ICP method exists, but their popularity is significantly lower and usually exhibit some kind of drawback, which limits the applicability and therefore do not impose real a competition to the ICP. Examples are: *Correlation*, RANSAC, *Principal component analysis* and some methods adopted from the field of computer vision (*SIFT*, *SURF*). Another possiblity is to model the point cloud as a *mixture of Gaussians*, but unfortunately, the methodology is applicable only to raw point clouds.

Methods directly operating on more complex primitives, such as line segments, non-uniform rational B-splines, triangle meshes etc. are very rare. [1], [15] and [14] are sparse remarks on the theme of line segment based shape matching. Second area in the shape registration field, where advanced primitive sometimes appear, are the modifications to the original ICP. For example the *generalized ICP* [22] uses local planar approximations.

Some measure of similarity or distance betweed the shapes is necessary for each registration algorithm. The most relevant criteria to the demands of SLAM are based on the Hausdorff distance, but many others exist. Unfortunately, most of them are hard to describe mathematically because of the inbuilt conditional statements and require repeated recomputation in the iterative fitting algorithms, which limits their performance significantly. From the point of view of robotic mapping, there is clearly a need to leave points and model the environment in a more thorough way. Reinvestigation of this area is therefore in place.

7.2 Vector maps similarity criterion

The method described further in this section satisfies the requirements of the SLAM problem and was already published in the paper [11] written by the author of this thesis. Mobile robotics requires variance to all transformations, with zero output for any two line segments lying on the same line. This behaviour is of great importance, because the robot, due to an obstructed view, rarely observes an object as a whole. Partial information about the edges of the surrounding objects results into uncertainty about their real dimensions, because there is no prior information, which part of the real edge did the robot actually sensed. This uncertainty is expressed as a perfect match anywhere along a line defined by an infinite extension of the static line segment. Only two or more skew line segments can solve this ambiguity and define a single transformation between the new and the static data.

In the following computations, the two arbitrary line segments defined by the end-points $\boldsymbol{AB}$ and $\boldsymbol{CD}$ are being used. To keep the notation lucid, a magnitude of a cross product is used even for a 2D problem. For every crossproduct in this section a condition of zero z coordinate for any vector involved is applied. Further equations frequently contain a substitution: $\mathbf{x} = \boldsymbol{B} - \boldsymbol{A}$ and $\boldsymbol{y} = \boldsymbol{D} - \boldsymbol{C}$. The criterion is defined as a sum of two squared areas:

$$S_{crit}(l_1, l_2) = \| (\boldsymbol{B} - \boldsymbol{A}) \times (\boldsymbol{D} - \boldsymbol{C}) \|^2 + 16 \| (\boldsymbol{B} - \boldsymbol{A}) \times (\boldsymbol{S} - \boldsymbol{A}) \|^2, \tag{7.1}$$

where $\boldsymbol{S} = (\boldsymbol{A} + \boldsymbol{B} + \boldsymbol{C} + \boldsymbol{D})/4$. The coefficient 16 balances the influence of both parts of the criterion and its value is justified in the full version of the thesis. Originating from this definition, the requirements stated above were proved to be fulfilled.

In practice, there are a lot of situations, where two sets of line segments are being tested for similarity and one of them is often iteratively adjusted by a rigid transformation according to a formula $\boldsymbol{P'} = \mathbf{R} + \boldsymbol{t}$, where $\mathbf{R}$ is a rotation matrix, $\boldsymbol{t}$ is a translation vector and $\boldsymbol{P}$ some point to be transformed. The overall similarity for a set of N line segment pairs is then given by the sum:

$$\begin{aligned}
\sum_{i=1}^{N} S_{crit,i} = \; & s^2 \sum (\boldsymbol{x}_i \cdot \boldsymbol{y}_i)^2 + c^2 \sum \| \boldsymbol{x}_i \times \boldsymbol{y}_i \|^2 + 2cs \sum (\boldsymbol{x}_i \cdot \boldsymbol{y}_i) \| \boldsymbol{x}_i \times \boldsymbol{y}_i \| \\
& + \sum \| \boldsymbol{x}_i \times \boldsymbol{v}_i \|^2 + 2s \sum (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) \| \boldsymbol{x}_i \times \boldsymbol{v}_i \| + 2c \sum \| \boldsymbol{x}_i \times \boldsymbol{z}_i \| \| \boldsymbol{x}_i \times \boldsymbol{v}_i \| \\
& + 4t_y \sum x_{x,i} \| \boldsymbol{x}_i \times \boldsymbol{v}_i \| - 4t_x \sum x_{y,i} \| \boldsymbol{x}_i \times \boldsymbol{v}_i \| \\
& + s^2 \sum (\boldsymbol{x}_i \cdot \boldsymbol{z}_i)^2 + 2cs \sum (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) \| \boldsymbol{x}_i \times \boldsymbol{z}_i \| + c^2 \sum \| \boldsymbol{x}_i \times \boldsymbol{z}_i \|^2 \\
& + 4st_y \sum x_{x,i} (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) - 4st_x \sum x_{y,i} (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) \\
& + 4ct_y \sum x_{x,i} \| \boldsymbol{x}_i \times \boldsymbol{z}_i \| - 4ct_x \sum x_{y,i} \| \boldsymbol{x}_i \times \boldsymbol{z}_i \| \\
& + 4t_y^2 \sum x_{x,i}^2 - 8t_x t_y \sum x_{x,i} x_{y,i} + 4t_x^2 \sum x_{y,i}^2,
\end{aligned} \tag{7.2}$$

where $\boldsymbol{x} = \boldsymbol{B} - \boldsymbol{A}$, $\boldsymbol{y} = \boldsymbol{D} - \boldsymbol{C}$, $\boldsymbol{z} = \boldsymbol{D} + \boldsymbol{C}$, $\boldsymbol{v} = \boldsymbol{B} - 3\boldsymbol{A}$, $c = \cos(\theta)$, $s = \sin(\theta)$, both from the rotation matrix $\mathbf{R}$ and t_x, t_y come from the translation vector. Separation of the variables related to the rigid transformation and the sums originating from the initial description of every examined line segment is a crucial result. It allows to precompute the sums only once for the whole set and then evaluate the cumulative similarity for any transformation in constant time. Assuming N is the number of line segment pairs being examined and T is the number of transformations performed, the computational complexity of the whole task is reduced from $O(NT)$ for established criteria to $O(N + T)$ for the presented criterion. This could lead to significant performance improvement of iterative matching algorithms, which rely on completely static, or partially updated data set.

All features and performance in the iterative algorithms were tested practically in benchmarks and the experimental results exactly confirmed the claims stated in this section.

7.3 Registration of the vectorized laser scans

This section deals with a procedure of analytic computation of an optimal transformation, which fits the corresponding line segments in a single step. Contrary to the previous approach, this method is not suitable for similarity evaluation, because it always finds the optimal transformation and the ambiguity metric, which reflects the criterion value, corresponds to the state *after* that transformation was applied. The methods are therefore applicable in different situations.

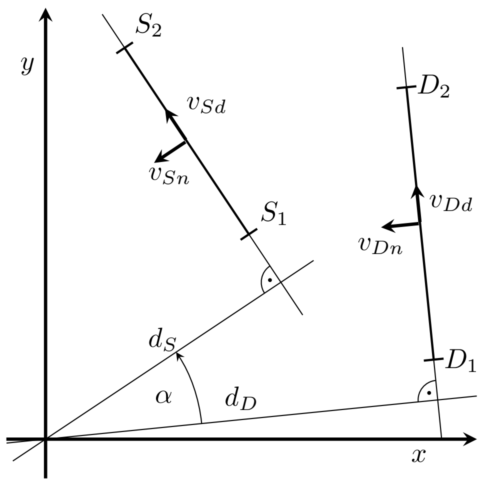


Fig. 7.1: Static (S) and dynamic (D) line segments during the fitting process with directional and normal vectors marked out.

Equations in this section stem from the situation depicted in Figure 7.1. For every two corresponding line segments, an infinite amount of rigid transformations exists (for the purposes of robotics, only rotation and translation are used, reflection was omitted), which bring the dynamic line segment on the same line with the static one. In the three dimensional space of all possible transformations (one rotation and two translations in 2D) the infinite set of possibilities forms a line, which can be described by the following equation:

$$p_i : \begin{bmatrix} (d_{Di} - d_{Si})\boldsymbol{v}_{Sni} \\ \alpha_i \end{bmatrix} + \delta_i \begin{bmatrix} \boldsymbol{v}_{Sdi} \\ 0 \end{bmatrix} = \boldsymbol{P}_i + \delta_i \boldsymbol{v}_i, \tag{7.3}$$

where all the variables in the first part correspond to Fig. 7.1 and $\boldsymbol{P}_i$ and $\boldsymbol{v}_i$ are denominations for further computation. There is only one acceptable rotation for all transformations, which will greatly simplify further computations. If more pairs of line segments are being examined, the subspace of all possible transformations is determined for all of them.

Optimal transformation is then computed as the closest one to all precise transformations. Point to line distance in 3D is given by $l_i = \| (\boldsymbol{x} - \boldsymbol{P}_i) \times \boldsymbol{v}_i \| \, / \, \| \boldsymbol{v}_i \|$, where $\boldsymbol{x}$ is the wanted transformation and $\boldsymbol{P}_i$ and $\boldsymbol{v}_i$ come from equation (7.3). Searching for $\boldsymbol{x}$ with arbitrary weight w for each correspondence leads to a least squares problem:

$$\sum_{i=1}^{N} w_i \frac{\partial l_i^2(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial (\| \boldsymbol{x} - \boldsymbol{P}_i \|^2 - (\boldsymbol{x} - \boldsymbol{P}_i) \cdot \boldsymbol{v}_i)^2}{\partial \boldsymbol{x}} = 0, \tag{7.4}$$

leading to a set of linear equations:

$$\begin{bmatrix} \sum w_i v_{yi}^2 & -\sum w_i v_{xi} v_{yi} & 0 \\ -\sum w_i v_{xi} v_{yi} & \sum w_i v_{xi}^2 & 0 \\ 0 & 0 & \sum w_i \end{bmatrix} \begin{bmatrix} x_x \\ x_y \\ x_\alpha \end{bmatrix} = \begin{bmatrix} \sum w_i P_{xi} v_{yi}^2 - \sum w_i P_{yi} v_{xi} v_{yi} \\ \sum w_i P_{yi} v_{xi}^2 - \sum w_i P_{xi} v_{xi} v_{yi} \\ \sum w_i P_{\alpha i} \end{bmatrix}. \tag{7.5}$$

Solving for x_x, x_y directly provides optimal transformation. x_α, being an angle, is a circular quantity, so the Mitsuta's averaging method should be used instead of a simple mean. At this point, an optimal transformation for a given set of corresponding line segments is successfully computed. The only possibility of failure appears, if all static line segments are *exactly* collinear, but even if the line segments are *nearly* collinear, the result is unreliable.

Reliability can be enumerated using a dispersion of the unit direction vectors of the lines in the transformation space from the equation (7.3). To keep the computation centred at zero, the opposites of the direction vectors are added as well and a covariance matrix is formed:

$$\mathbf{E} = \begin{bmatrix} \mathrm{Var}(x) & \mathrm{Cov}(x,y) \\ \mathrm{Cov}(x,y) & \mathrm{Var}(y) \end{bmatrix} = \frac{1}{\sum w_i} \begin{bmatrix} \sum w_i v_{xi}^2 & \sum w_i v_{xi} v_{yi} \\ \sum w_i v_{xi} v_{yi} & \sum w_i v_{yi}^2 \end{bmatrix}. \tag{7.6}$$

The reliability R is then defined as: $R = \sqrt{\mathrm{Det}(\mathbf{E})}$. It can be proved, that R can change in the interval $[0; 1]$, where zero demarcates an unsolvable situation and one means the highest reliability possible. There is a smooth transition between these two states, so the unreliable results caused by nearly collinear line segments can be easily detected.

Ambiguity exposes the internal criterion function with some additional options and reflects how much compromises were needed to find the optimal transformation. The distances between the transformation and the optimal lines are treated as tensed springs and the ambiguity corresponds to the energy accumulated:

$$E_\alpha = k_\alpha \left(\sum w_i P_{\alpha i}^2 - x_\alpha \sum w_i P_{\alpha i} \right), \tag{7.7}$$

$$E_{xy} = k_{xy} \left(\sum w_i (P_{xi} v_{yi} - P_{yi} v_{xi})^2 - \sum w_i (x_x v_{yi} - x_y v_{xi})^2 \right), \tag{7.8}$$

where k_{xy} and k_α are the appropriate stiffness coefficients and the overall ambiguity is defined as $A = E_\alpha + E_{xy}$. If the match is perfect, the lines of optimal transformations intersect at one point and the ambiguity is zero. Noised measurements from practical experiments exhibit some ambiguity, but it stays limited. If the limit is exceeded, a strong suspicion of badly set correspondences is in place.

7.4 Correspondences extracting algorithm

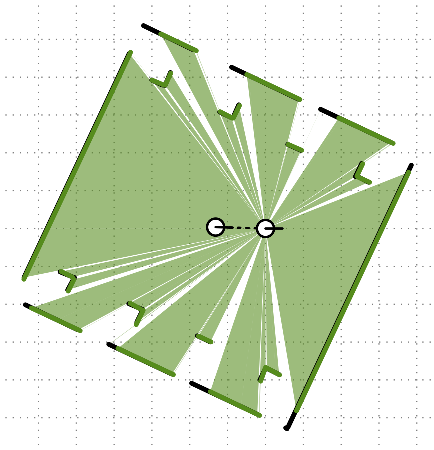
An important assumption employed in both previous sections was a prior knowledge of the correspondences. This is obviously not a frequent case in practice, so the need of correspondence finding algorithm is high. The idea of the algorithm is based on testing of the various sets of possible correspondences with the method above. Gradual process of searching through the entire space of all possible correspondences should lead to minimization of the internal criterion and result into final set of highly probable correspondences.

All potential correspondences are found at first. The search space is given by the estimated transformation and its accuracy. Maximal error of translation and rotation demarcates a subset of all permitted transformations. For every pair composed of one line segment from the static and one from the dynamic set, a set of perfect match transformations according to the formula (7.3) is computed. If it crosses the permitted range of transformations, the pair is deemed to be potentially corresponding.

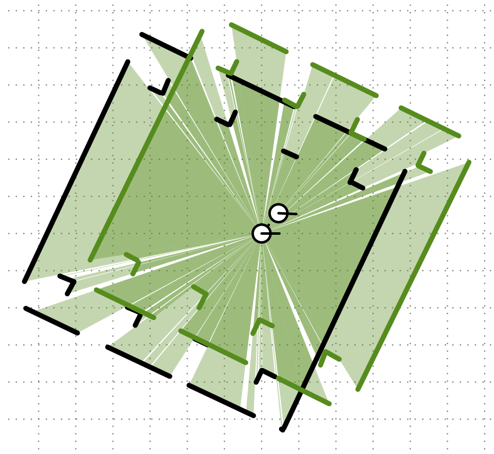
Unambiguous collinear bundles are established in the second stage of the registration process. The line segments in a single bundle are nearly collinear (low *reliability*), but the correspondences do not collide (low *ambiguity*). Operation of this stage is driven by two user provided thresholds. The reliability threshold affects the maximal permitted dispersion of the lines, while the ambiguity threshold reflects noise in the measurements.

Unambiguous bundle combinations result in the solutions, containing selection of the possible correspondences. Two different relationships between the bundles from previous section can appear. Some bundles, if combined, produce a new bundle which has low reliability and high ambiguity. This combination of properties leads to correspondence collision and the bundles must not appear in the same solution. The second possibility is high reliability and low ambiguity and signs a valid combination, which is a final result for arbitrary sets of line segments. More distinct solutions can be available.

Expected view test is an addition specifically designed for SLAM applications. Physical environment imposes some additional constraints, which can be exploited to filter the solutions even more. Based on the transformation given by the previous step, the algorithm generates an expected observation in the static data set and compares it to the dynamic one as shown in Figure 7.2. The light green area is the third error metric called *discrepancy* and, for a valid solution, it closes to zero.



(a) Correct correspondences, low discrepacy.



(b) Wrong correspondences, high discrepacy.

Fig. 7.2: Discrepancy visualization for correct and wrong sets of correspondences. The light green area demarcates the difference between expected observation and the dynamic set and corresponds to the discrepancy error metric.

7.5 Summary of the work on vector map registration

This chapter is dedicated to similarity evaluation and registration of the vectorized laser scans and this section summarizes the most important results.

Section 7.2 presents a novel line segment similarity criterion. It is fully differentiable and the derivatives are continuous in the whole domain of definition. The criterion is designed directly for SLAM applications in robotics. The correspondences between line segments must be established in advance, but the criterion also supports precomputation, which reduces computational complexity of some algorithms from $O(NT)$ to $O(N+T)$ (where N is a number of line segment pairs being examined and T a number of transformations performed). All of these features were theoretically derived and practically tested.

Section 7.3 describes an analytical approach to the vectorized scan registration with the known correspondences. The optimal transformation is computed in a single step. The method also provides a metric for evaluation of a *reliability* of the computation and *ambiguity*, which corresponds to the internal criterion function and reflects, how many compromises were necessary to align the corresponding line segment pairs. Precomputation is also supported, which means constant time evaluation, if a single corresponding pair is added or removed.

Section 7.4 is dedicated to the corresponding pairs searching in the two sets of line segments. The algorithm utilizes the previous method to evaluate different combinations of correspondences and aims to find the largest set of pairs resulting into reliable and unambiguous registration. For the purpose of robotic mapping, an additional criterion (*discrepancy*) was added, which stems from the physical constraints of the real-world measurement. Though it provides good results, the algorithm is in development and further testing and refinements will be definitely carried out.

8 CONCLUSION AND FUTURE WORK

This thesis is focused on processing of the raw point clouds for further utilization in SLAM in robotics. The preparation of the data starting at filtration and segmentation up to the association with the data already known, is usually referred to as the SALM front-end and seems to be a larger research challenge, than the probabilistic inference mechanism for the actual fusion of all measurements into a single, coherent map. The brief summary of the state of the art in Chapter 2 revealed a strong tendency towards semantic maps, which contain various information about the objects, including their physical dimensions [4]. Because these cannot be easily described by a set of points, more complex geometrical primitives are necessary to be used. Similar tendency can be seen in the point cloud registration algorithms, which, after the years of supremacy of the ICP method [2], seem to utilize approximations as well [22]. The research of vectorization and registration of the more complex geometrical entities is therefore worth the effort, because it directly addresses problems solved in robotics community and in some other fields as well.

The main contribution of this work is concentrated in Chapters 5, 6 and 7. Segmentation and filtration described in Chapter 5 is quite straightforward preparation for further algorithms, but as the approach is quite distinct from methods found during the literature review, it can be considered novel. The method is robust, the execution time low and the parameters of the algorithm directly reflect the physical essence of the measurement.

Chapter 6 is dedicated to vectorization of the continuous clusters using the total least squares method. First, an optimized algorithm (FTLS) is introduced, which significantly reduced computational time of the traditional incremental TLS algorithm and now it is probably the fastest way of computation of this kind of approximation available, especially for large point clouds. Second contribution is an augmentation of this algorithm (AFTLS) addressing the threshold related error, which is common for pretty much all of the fast vectorization methods. Further refinement of the results is able to suppress the problem and deliver better approximations. Results described in this chapter were published in [9], [12] and [10].

Registration and data association is covered in Chapter 7. The novel criterion function provides some useful features such as differentiability and continuity in the whole domain of definition and meets the requirements of SLAM applications. The method supports precomputation, which can reduce computatial complexity of certain set of algorithms. The criterion was published in [11]. Chapter 7 also presents a method for registration of two sets of line segments with known correspondences in one step, providing the optimal transformation, the reliability and ambiguity metrics. Based on this tool, an algorithm for correspondence searching is introduced. Though it is still in development, the theory can be considered proved and the results of the decision process of correspondence search are very promising.

The results briefly summarized above provide a good base for further research. Once the correspondence searching algorithm will be finished, there is plenty of directions to move further. Future work may lead to three dimensional problems, more complicated geometrical primitives such as b-splines, or the SLAM itself.

BIBLIOGRAPHY

[1] M. Alshawa. ICL, Iterative Closest Line: A novel point cloud registration algorithm based on linear features. *ISPRS 2nd summer school*, (10):53–59, 2007.

[2] P. P. Besl, N. McKay, H. McKay, and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[3] M. Bosse. Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework. *The International Journal of Robotics Research*, 23(12):1113–1139, 2004.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, dec 2016.

[5] P. Del Moral. Nonlinear Filtering: Interacting Particle Resolution. *Markov Processes and Related Fields*, 2(4):555–580, 1996.

[6] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, jun 2001.

[7] D. H. Douglas and T. K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, oct 1973.

[8] P. Gabrlik, A. Jelinek, and P. Janata. Precise Multi-Sensor Georeferencing System for Micro UAVs. *IFAC-PapersOnLine*, 49(25):170–175, 2016.

[9] A. Jelinek. Practical Aspects of Total Least Squares Vectorization of Point Clouds in Mobile Robotics. *IFAC-PapersOnLine*, 48(4):193–198, 2015.

[10] A. Jelinek and L. Zalud. Augmented Postprocessing of the FTLS Vectorization Algorithm - Approaching to the Globally Optimal Vectorization of the Sorted Point Clouds. In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, pages 216–223. SCITEPRESS, 2016.

[11] A. Jelinek and L. Zalud. Line segment similarity criterion for vector images. In *25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision WSCG 2017*, pages 1–7, 2017.

[12] A. Jelinek, L. Zalud, and T. Jilek. Fast total least squares vectorization. *Journal of Real-Time Image Processing*, jan 2016.

[13] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35, 1960.

[14] Y. P. Kwon. Line Segment-based Aerial Image Registration. Technical report, University of California at Berkeley, Berkeley, 2014.

[15] F. Lu. *Shape Registration for Using Optimization for Mobile Robot Navigation.* PhD thesis, University of Toronto, 1995.

[16] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[17] H. Moravec and A. E. Elfes. High Resolution Maps from Wide Angle Sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116–125, 1985.

[18] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, jan 1965.

[19] A. Nguyen and B. Le. 3D point cloud segmentation: A survey. In *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 225–230. IEEE, nov 2013.

[20] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. *Autonomous Robots*, 23(2):97–111, jun 2007.

[21] K. Reumann and A. P. M. Witkam. Optimizing Curve Segmentation in Computer Graphics. In *Proceedings of International Computing Symposium*, pages 467–472, Amsterdam, 1974. North-Holland Publishing Company.

[22] A. V. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. *Proc. of Robotics: Science and Systems*, 2:4, 2009.

[23] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, 4:167–193, 1987.

[24] R. Wallace, K. Matsuzaki, Y. Goto, J. Crisman, J. Webb, and T. Kanade. Progress in Robot Road-Following. *Proceedings 1986 IEEE International Conference on Robotics and Automation*, 3:1615–1621, 1986.

Papers [9], [10] [11], and [12] are author's original works publishing results summarized above. Paper [8] uses some of the technical means of experimentation developed for this thesis, but is not primarily focused on the discussed matters.

ABSTRACT

The doctoral thesis deals with processing of the point clouds produced by the laser scanners and subsequent searching for correspondences between the approximations obtained this way for the purpose of simultaneous localization and mapping in mobile robotics. The first method is dedicated to filtration and segmentation of the data and is able to perform both operations at the same time in one algorithm. For vectorization, an optimized total least squares algorithm is introduced. It is probably the fastest in its category and closes to the eliminating methods, which provide significantly worse approximations. For similarity evaluation, optimal registration and correspondence search between two sets of vectorized scans, the analytical innovative methods are presented as well. All of the algorithms introduced are thoroughly tested and their features are investigated in many experiments.

KEYWORDS

Point cloud, segmentation, filtration, vectorization, least squares, similarity criterion, registration, correspondence search, SLAM.

ABSTRAKT

Disertační práce se zabývá zpracováním mračenen bodů z laserových skenerů pomocí vektorizace a následnému vyhledávání korespondencí mezi takto získanými aproximacemi pro potřeby současné sebelokalizace a mapování v mobilní robotice. První nová metoda je určena pro segmentaci a filtraci surových dat a realizuje obě operace najednou v jednom algoritmu. Pro vektorizaci je představen optimalizovaný algoritmus založený na úplné metodě nejmenších čtverců, který je v současnosti patrně nejrychlejší ve své třídě a blíží se tak eliminačním metodám, které ovšem produkují výrazně horší aproximace. Inovativní analytické metody jsou představeny i pro účely vyjádření podobnosti mezi dvěma vektorizovanými skeny, pro jejich optimální sesazení a pro vyhledávání korespondencí mezi nimi. Všechny představené algoritmy jsou intezivně testovány a jejich vlastnosti ověřeny množstvím experimentů.

KLÍČOVÁ SLOVA

Mračno bodů, segmentace, filtrace, vektorizace, metoda nejmenších čtverců, kritérium podobnosti, sesazování, vyhledávání korespondencí, SLAM.

JELÍNEK, Aleš. *Vectorized Point Clouds for Mobile Robotics*. Brno, 2017, 30 p. Doctoral thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Advised by prof. Ing. Luděk Žalud, Ph.D.

Vysázeno pomocí balíčku thesis verze 2.63; http://latex.feec.vutbr.cz