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 — **Ing. Aleš Jelínek**
AUTOR PRÁCE

SUPERVISOR — **prof. Ing. Luděk Žalud, Ph.D.**
ŠKOLITEL

BRNO 2017

ABSTRACT

This doctoral thesis deals with processing of point clouds produced by laser scanners and subsequent searching for correspondences between the approximations obtained in this way for the purpose of simultaneous localization and mapping in mobile robotics. The first method performs filtration and segmentation of the data and is able to do both operations at the same time in one algorithm. For vectorization, an optimized total least squares algorithm is introduced. It is probably the fastest algorithm in its category, comparable even to the eliminating methods, which, however, provide significantly worse approximations. For similarity evaluation, optimal registration and correspondence search between two sets of vectorized scans, new analytical methods are presented as well. All of the algorithms introduced were thoroughly tested and their features 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, 145 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.

DECLARATION

I declare that I have written the Doctoral Thesis titled "Vectorized Point Clouds for Mobile Robotics" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Doctoral Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

I would like to say thanks to many people, especially to my supervisor prof. Ing. Luděk Žalud, Ph.D. for his help with my work and guidance through the academical world, to my parents, who made my studies possible and to my wife, for her patience and care.

Brno ....

author's signature

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Simultaneous localization and mapping (SLAM) is a vital part of robots' 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 meaning of the mapped objects. AI researchers proceed in this direction and the following text is meant to contribute to their effort.

This thesis is focused on extraction of geometrical information from raw point clouds and presents sound 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 in scope of this work.

Apart from the theoretical development of novel algorithms, during the work on this thesis there were also some practical engineering problems solved. 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 a part of ATEROS robotic system built and maintained by our research group.

2 STATE OF THE ART OF SIMULTANEOUS LOCALISATION AND MAPPING

Some scientific questions are answered with a single, elegant formula and as far as we know, the solution is definite and complete. But endless combinations of these fundamental principles form a tremendously complicated reality, where generalisation and abstraction is vital for any practical problem solving. Although robotics from the hardware point of view is an exact engineering, the AI tends to overflow to the realm of the "soft sciences", where certainty turns into probability and equality into similarity. SLAM as an integral part of the robot's AI is not an exclusion. 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 work in the rest of this thesis.

2.1 A brief history of SLAM

Robotics in terms of human dreaming about artificially created life is a very old discipline, but in the context of this thesis, we can safely skip all of the voodoo puppets, alchemical homunculi and mechanical automata and move to the beginning of 20th century where the first sings of modern robotics can be spotted. At that time, Nikola Tesla presented his remote controlled boat and in 1920s Karel Čapek published his famous fiction R.U.R. [1], where the term *robot* was introduced and the whole science around it got its name. More serious attempts on creation of autonomous robots were made, Isaac Asimov formulated his famous laws [2] and Norbert Wiener established cybernetics as a science in 1948.

The great expectations put into the robotics discipline turned out to be too optimistic. As the time passed and more researchers tried to build a truly autonomous robots, it became quickly clear, that there are certain aspects of autonomy, which are very natural to humans (and many other living creatures as well), but are quite problematic for artificial implementation. Besides many other challenges, an issue related to orientation in the surrounding environment and memorization of its geometry, texture and meaning have appeared. The first solutions of a robot motion were based on tactile sensors and relatively simple logic rules but as the computers evolved, more advanced algorithms could be proposed and the history of SLAM has finally begun.

2.1.1 Classical solutions

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) [3]. The method partitions a continuous space into a matrix of discrete squares and checks a 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[1] , 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* [4], [5]. In contrast to OG, the map representation is continuous and contains coordinates of the important places in the environment and possible routes between them. Though topological maps are usually considered to be an opposite of metric maps, the presence of coordinates constitutes a linkage to the real-world geometry and makes distinction less strict. Pure graph of nodes (locations) and edges (paths) would have limited usage in practical navigation. Positive aspect of this approach is a direct guidance for a robot, when certain task in known environment is fulfilled. On the other hand, 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. Although it is a marginal approach today, some researchers still work in this direction [6].

Soon after these approaches were introduced, the first probabilistic algorithms have appeared [7]. 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 [8]. Pose estimation is a low-dimensional task, where only a small amount of sensory data is fused, but a theoretical dimensionality[2] 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

[1] *Pose* is a term covering both robots position in the given system of coordinates (x, y, z) and the angular orientation (*roll*, *pitch*, *yaw*).

[2] In statistics, the *dimensionality* of a problem corresponds to the number of independent random variables. For a pose estimation problem, the dimensionality is limited, but for a map, where every landmark is described by a position, the dimensionality grown rapidly.

significantly. The computation process involves matrix inversion with $Q(n^3)$ complexity in basic implementation and theoretical limit $O(n^2 \log n)$. Both cases are unacceptable for larger data sets. Under certain circumstances, this can be bypassed by introduction of the *information filter* [9], [10] with a sparse information matrix. Sparsity of the matrix comes at a cost of omission of some details, but allows to achieve better computational times. Second possible drawback is KF linearity and dependence on Gaussian distribution of the noise in measurements. In case of a map, the linearity condition is met, but the simultaneous pose estimation leads to a more complex model involving non-linear functions. Contrary to the computational complexity problem, non-linearity have been effectively dealt with in many derivations of the original KF such as *Extended Kalman filter* [11] [12], *Unscented Kalman filter* [13] and many others. Despite the downsides described, KF is a very popular technique in the SLAM community and many successful application of this approach were published [14], [7], [15], [16], [17].

Expectation maximization (EM) technique stems from maximum likelihood estimation [18], [19]. This process estimates parameter 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. During the *expectation step* of the algorithm, a set of hypothesis about robots pose and correspondences present is generated. In the following *maximization step* those correspondences are incorporated into the map. The method converges to a true map, because the true hypotheses appears regularly in the observations reinforcing its influence on the result, while false hypotheses are distinct and their influence vanishes over time. 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 of measured data after a mission of the robot has finished [20]. If the full SLAM is solved and the robot's pose is unknown, the algorithm has exponential computational complexity and it can potentially fail on the loop-closing problem[3] resulting in an inhomogeneous map. This is the reason, why the expectation maximization method is usually combined

[3]The loop-closing is part of the data association process. When a robot changes its pose between observations only a little bit, the search space for data association is limited and a correct correspondences are usually found. A complicated case appears, when the robot explores for example a long curvy corridor and after some time it visits a place, where it already was before. Due to a long distance travelled and possibly large pose estimation error accumulated between the visits, the data association can easily become wrong and therefore the resulting map becomes inhomogeneous (one place in reality has two separate depictions in the map) or corrupted (the end of the loop is connected to a wrong part of the map, so one place in the map corresponds to two distinct places in reality).

with another algorithm addressing the localization part of the SLAM and EM is used only for local map building [21],[22].

Particle filter (PF) [23] 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. This combination also lead to an alternative labelling of PF: *A multi-hypothesis Kalman filter*. The principle of operation of the PF consists in maintaining a set of possible models (particles) and iterative evaluation of their fitness according to given data. The models with low fitness score are discarded and new particles are generated near their more successful counterparts. As the algorithm proceeds, the particles tend to accumulate in a dense cluster around the best fitting model. 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. The downside of the method is a high growth of the computational time with growing dimensionality and the number of particles, which makes it ineffective for even mid-sized maps. Rao-Blacwellized [24] version of PF was introduced to overcome this issue [25],[26], but the promising results were found to be unstable after an extensive period of operation [27]. The main area of PF application is therefore pose estimation, where the dimensionality is limited and the multi-hypothesis approach is highly favourable. For the full SLAM solution, another method is usually used for map building, for example a combination with EM methods from the previous paragraph is a beneficial combination [22], [28].

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. KF approach derivatives are (among many others) monocularSLAM [29] and [30] , SLSJF [31], and RT-SLAM [32]. The well known applications of PF method are for example FastSLAM1.0 [25] and FastSLAM2.0 [26]. EM have evolved into GraphSLAM [20] and other mixed approaches such as [22]. Time to time some experiments employ evolutionary algorithms [33], neural networks [34] or some other novel approach, but these are rare occasions. Many open-source implementations with varying quality and complexity are also available on the internet. Over the last two decades, SLAM have surely became an important and popular research field and the results achieved are unexceptionable. The base concepts are presented in specialized books [14] and thorough surveys regularly map the actual advancements and trends in the field [35], [36], [37], [38], [39], [40] and [41]. Reviewing the most recent of these 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.

2.1.2 Theoretical considerations for practical applications

In 2001 an important paper [42] with a very promising title *"A solution to the simultaneous localization and map building (SLAM) problem"* was published. From a theoretical point of view, these were a ground-breaking news. 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 work adopts the KF as a core probabilistic mechanics and demonstrates, that in the limit, the landmarks become fully correlated and the error of the distance between them converges to zero, if the method is correctly performed. The theory seemed to be mostly finished and a great algorithms were about to be published.

Looking at the previous chapter, an utter majority of approaches rely on KF or Bayesian probability in general. On the other hand, in many cases some limitations were discussed, namely computational complexity and feature correspondence were frequently mentioned. Once again, a bare existence of theoretical solution did not implied a straightforward practical results.

Computational complexity (CC) is a vital attribute of any algorithm intended to operate on a variably sized set of input data. If a robot should be able to map a large scale environment, CC of the underlying method has to be the lowest possible. Obviously, in conjunction with a limited computational power of any hardware, any CC but constant-time ($O(1)$) posses some upper limit on the maximal size of the operational area. As long as this upper bound is high enough, the practical applicability is not endangered. So far, the only algorithm claiming $O(1)$ operation was FastSLAM [25], [26], but it has other drawbacks, disqualifying it as a candidate on a full-featured SLAM. On the other hand, there is not yet an evidence, that the SLAM problem has some lower bound on a computational complexity. Some hope for $O(1)$ algorithm still exists, however complexities of the state of the art approaches lead to more pessimistic expectations.

Feature correspondence, similarly to CC, is also known to be essential from the very beginning of the SLAM research [35] until now [41]. Orientation with respect to the surrounding environment requires extraction of significant landmarks and their recognition from different points of view. Clearly not all landmarks are suitable for this purpose and mismatches inevitably happen. If a correspondence is present but not spotted, it usually does not mean a significant problem, maybe a little slowdown of convergence. Much more sever consequences may have a false positive[4] correspon-

[4]Statistics of classification of a binary quantity allows four cases to happen: *True positive* - the

dence making a link between two non-corresponding landmarks [35]. Outcome of such a situation heavily depend on the data fusion method used. If a method can handle certain amount of erroneous correspondences (like for example most of the approaches EM does), the SLAM algorithm will be stable during a long period of operation. High susceptibility to mismatches (the case of many simpler KF algorithms) means, that only a very well distinguishable features[5] can be used, otherwise the solution gets inevitably corrupted after certain time. CC of feature correspondence search is not as limiting as in the case of the mapping algorithm, because the search space is usually bounded. Every moving robot can change its pose in a given time interval only by a certain degree, because of its mass, motor performance and so on. If the environment is homogeneously populated with landmarks and measurements are acquired in periodical intervals, every feature matching algorithm assuming this upper bound, deals with more or less constant amount of data every iteration. Despite this practical positive effect, a successful correspondence search is usually a demanding operation, so performance optimization is still important.

More complicated situation arises, if the search space is not limited. This is the case of *loop-closing*, when the robot visits a place it was already before, but comes from unexplored area. Such a situation is not rare, basically every crossroad means potential loop-closing issues. Lack of upper bound disqualifies many state of the art algorithms for consecutive measurements and leads to the methods using hiearachical structure [21], general descriptors [43] and various heuristics [44] to reduce the dimensionality of the problem at first. This process may lead to a single successful solution or multiple hypothesis. In later case the solution branches out and only further exploration can eliminate false expectations. An extreme example of unbound feature matching is the *kidnapped robot problem*. Kidnapping stands for any unexpected and significant change in robots pose and usually results in searching through the whole map. This situation means critical failure for most of the SLAM algorithms, only some PF based approaches are designed to deal with it.

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) [42]. This behaviour is critical for long term operation of the robot, because non-converging method usually diverges and provides more and more erroneous results

correspondence really exists and was found, *true negative* - the features do not correspond and were denoted as such, *false positive* - there is no correspondence, but the algorithm marked it valid and *false negative* - a correspondence is present, but was not discovered by the algorithm. Testing of the decision making algorithms often follows this methodology.

[5]Really high level of feature differentiality is in practice achieved using some kind of artificial landmarks. On the other hand, full-featured SLAM should not rely on modifications of the environment, because it breaks autonomy of the robot.

as the time proceeds [41]. Theoretical proof of the convergence is therefore an important attribute of any SLAM algorithm, but due to unexpected influences affecting the robot a general proof is hard or even impossible to obtain. Some set of initial assumptions usually precedes the proofs and behind these borders only a practical verification is available.

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. External effects influence the sensors (illumination, electromagnetic noise) or directly the robot (slippery ground, colliding objects, partial breakdown). Also the environment may contain objects which are familiar to us, but a robot encountering a glass pane or steel netting can easily critically fail. Another big set of possible hazards stems from a software implementation of the mathematically exact procedures. Variety of issues spans from trivialities such as division by zero and continues over a finite precision of variables up to numerical stability of the whole algorithms. Though in case of correspondence mismatch we usually demand fault tolerance, the rest of possible failures is mostly dealt with in a fail safety manner[6].

Dynamic environment brings a whole new set of complications for the SLAM solving algorithms. Correspondence evaluating procedures usually rely on a fact, that the features observed are still, no modifications to the environment are happening and only the robot is considered to be moving. The map is therefore static and the only modifications permitted origin from the precision improvement caused by incorporation of the new data. Movement of the robot itself is a different, although very related, task. 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. Large variations in possible dynamics are also a serious issue. Changes can happen gradually over a longer period of time (melting snow), quickly, but infrequently (cars in a parking place), unpredictably but smoothly (animals, people) or even faster than the robot can reliably sense (thrown or falling objects). A general treatment of this task was not yet presented, but several attempts have already emerged [45], [46], [47], since the need for such solution is quite urgent. An autonomous robot cannot operate in the real world if it cannot cope with other moving entities.

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

[6] *Fault tolerant* system is one, that can successfully operate even if the operating conditions are not fully satisfied. *Fail safe* system is designed to turn down with minimal collateral damage possible if a failure of some of its part appears. Importance of these policies in mobile robotics is evident.

and a map built. On the other hand, too dissected structure leads to mismatches during correspondence search and highly complicated maps flooded with details. Typical examples are natural locations with bushes, tall grass, treetops and so on. Presence of even a mild dynamics (wind) in these environments makes them pretty much impossible to map with the most of the state of the art algorithms, because the vast majority of features rapidly changes in a fraction of second. Thorough filtration is used to separate well distinguishable static features from a noise of untrustworthy ones or a high level abstraction, which works with the whole objects, whose dynamics is limited. The filtration approach produces maps with sparse features and a lot of data specific to a single measurement, which makes them inconvenient for further utilisation. The abstraction approach produces much more natural maps with complex objects, each covering many features. In later case, the map is more appropriate, but the algorithms working this way are far from maturity.

Map representation and semantics are the issue here. So far, we have spoken about occupancy grids, topological maps and metric maps. Although essential for the past SLAM (and majority of recent attempts as well), these approaches seem to lack some significant information, which limits, how far we can advance with them. Today, the most of the algorithms for SLAM rely on point-like features, creating a map of their relative positions. Contrary to this approach, 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. Once the robots will grasp at least the basic meaning of the world around them, it will be a qualitative revolution comparable to the introduction of advanced probabilistic techniques to the world of SLAM research.

More interesting topics in SLAM research exits, for example multi-robot cooperation during map building [48] or exploration planning [49], but these are not very relevant to the focus of this thesis.

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. Nature and the living creatures in particular are being in question in this case. 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. Questions "Where am I?" and "Where have I been?" were spoken much later, but the answers quietly appeared, when the first animal found its hideout, water source, or whatever important thing it needed to return to. At that time, the first SLAM task on the planet Earth was successfully solved.

Biological SLAM is surprisingly efficient. For example a honeybee, which is quite smart insect species, has only approximately one million of neurons [50] in a brain[7]. Even with that "computational power" it can memorize position of its home apiary, locations of the blooming flowers, communicate this knowledge to the other members of the hive [54] and, if it happens to fall into hands of curious researchers, go through a maze labelled with colourful landmarks [55]. All of these tasks prove honeybees ability to successfully solve the SLAM problem. Bees map of the world probably would not satisfy a human observer, but with one cubic millimetre of neural cells and negligible energy consumption it is an impressive result.

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. Cognitive psychology examines our spatial memory from the perspective of information processing (both conscious and unconscious) and neuroscience maps the brain in search for physiological basis of its abilities. Bare comparison of a human brain and a modern computer shows enormous difference in strengths and weaknesses of both[8] and so is different the approach (and results) of SLAM.

Early literature on human orientation and spatial memory mostly addresses consequences of injuries and reveals, which parts of the brain are related to these functions [56]. Episodic, short term memory for space recognition [57] is dedicated to reasoning, learning and remembering of the already learned information. For robotics research are more inspiring the recent results, revealing hierarchical structure of the spatial memory [58] and the landmark - layout dualism [59] of the environment representation in a long term memory [60]. Layout generally refers to the determining shape of the area, while the landmarks are well distinguishable objects [61], [62]. Recognition and matching of both can be explained by the hierarchical system and

[7]For comparison - one million is also an approximate number of ganglion cells (or neurons for simplicity) in retina of a single human eye [51]. Besides those cells, there are approximately 100 millions of photoreceptors [52] and a large amount of other specialised cells [53]. In this context, abilities of honeybees brain look truly remarkable.

[8]Modern, consumer grade graphics cards have a computational power of circa $5 - 10 \cdot 10^{12}$ floating point operations per second. Under a very optimistic assumption, that every human being on the Earth can perform a single floating point operation every second, an inexpensive GPU is still *a thousand times* faster than the whole mankind. Contrary, neither supercomputer was able to pass the Turing test yet, so fast arithmetic is clearly not everything.

the massively parallel structure of the brain, allowing high success rate and reaction times in fraction of a second [63]. The most important is the fact, that layouts of small areas and objects are probably treated and compared as a whole. That way, every comparison is supported by much larger amount of information, than in the case of point-like features, leading to better results of more complex approach. Incomplete information, which sometimes limits the artificial algorithms, is in the natural SLAM supplemented by the long term experience [64]. However the exact function of the spatial memory and the related mechanisms is not yet fully understood, the general findings described above provide a good inspiration for artificial attempts at SLAM.

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". It was possibly the daily experience with the natural sense for localization and space memory, which kept the scientists optimistic and interested, even though the mathematical evidence of SLAM solubility was not yet proven. 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.

Maps with point-like features achieved great success, but their potential seems to be nearly depleted. 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.

Related literature presents many concepts, principles, methods and algorithms producing and processing the boundary description of the scene. Most of the papers are very recent and no established methods or unified theoretical background exists. When this work have started four years ago, the situation was even less clear and beginning from a 2D point cloud and vectorization using line segments seemed to be an easy introduction into more complex matters. As the work proceeded, some decades old, established algorithms turned out to be improvable, some theory turned out to be too limiting and a room for discoveries unexpectedly appeared. Content of this thesis is focused on point cloud processing, vectorized scan matching and correspondence search, which is sometimes called a SLAM front-end [41]. 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 VECTOR BASED FRONT-END FOR SLAM

Every larger process containing many cooperating parts needs some organization structure and a SLAM front-end is not an exclusion. The first section of this chapter provides an overview of the whole process and identifies compact, indivisible operations and information flows connecting them together. The next section briefly explains structure of the rest of the thesis.

3.1 Data processing workflow

In picture 3.1 is a highly simplified scheme of the operations involved in the presented SLAM front-end. The very beginning of the process is information acquisition from the environment, while at the end, data are received and provided to a subsequent data fusion back-end. The algorithms in between were all devised and implemented by the author of this thesis and will be described in detail in the following text.

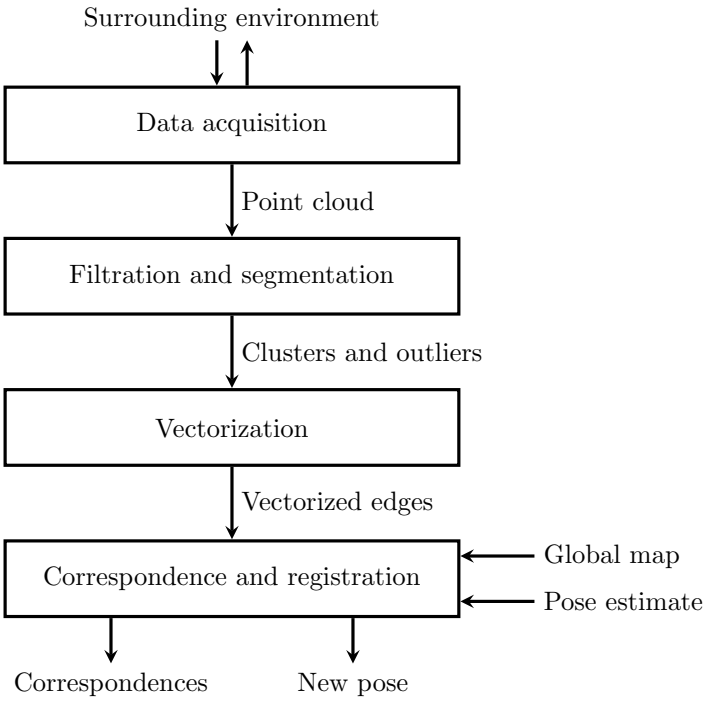


Fig. 3.1: Schematic workflow of the data processing in the vector-based SLAM front-end.

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 outputs of the whole process. Primary sensors can be either passive (senses only, e.g. RGB camera) or active (affects the environment and senses the response, e.g. laser scanner). This step involves large amount of measurements in a quick succession (laser scanner) or a truly parallel measurement with a matrix of the sensing elements working at the same time (camera). In both cases the sensing period should be short enough, so the environment could be considered static and the data set is called a *scan*. If this condition is not met, significant problems with shape distortion appear, making further processing much more complicated. 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. Unification of the data structure and coordinate system allows to freely switch between different data acquisition systems or following processing algorithms. For the purposes of this work, only a 2D slice (mostly horizontal) is produced.

Filtration and segmentation of the raw point cloud at this moment might seem inappropriate with respect do the discussion in Section 2.3, but it is not. The goal of this step is not an identification of the objects in a scene, but merely a 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, sometimes also called *clustering* or segmentation, is intended to separate parts of the point cloud, which could potentially form a continuous edge in the real world. Point cloud split into a set of outliers and a couple of clusters then passed to the next step.

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. In case the lines are properly fitted into the cloud and every point contributes its bit of information to the result, vectorization also reduces the natural noise present in the data. Third important feature of this process is generalization. Conversion of discrete data back into continuous entities has some caveats, but if appropriate care is taken, the benefits largely outweigh potential risk. After the vectorization is finished, all clusters are replaced by the line segments (or edges in general).

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. To determine the portion of the global map taken into account a crude pose estimate and maximal permissible error are used. In case, that no pose estimate is available, the last known position is used and the constraints are given only by physical limits of the robot. This

operation limits the search space of the algorithm and allows its fast operation. 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.

3.2 On further structure of this document

Literature review in Chapter 2 is intentionally focused on SLAM techniques rather than vectorization, matching and other topics closer related to the theme of the work. The initial review is provided for illustration of the localization and mapping matters as a whole and identification of the recent research interests.

Chapter 4 covers technical content, mainly focused on construction of the robot Hermes, which was used in practical experiments. The interesting software and hardware solutions are mentioned and data acquisition through a laser scanner is described.

Chapters 5, 6 and 7 correspond to the second, third and fourth block in Fig. 3.1 and represent the core of this work (especially 6 and 7). Since these algorithms are mostly studied separately by the researchers, each chapter starts with a review of the literature available and further sections approximately follow a structure of a regular research paper as well. This strategy should allow a reader to read those chapters separately with all the relevant information at one place.

As usual, Conclusion 8 finalises the work and discusses the results achieved and possible future improvements.

4 TECHNICAL MEANS FOR PRACTICAL EXPERIMENTS

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. This thesis is primarily focused on point cloud processing, which has only a very few direct connections to the hardware, so the technical implementations will be covered briefly with respect to their crucial features and deeper descriptions will be omitted. On the other hand, a lot of time-consuming work was put into the robot Hermes (see Figure 4.1) and the thesis would not be complete without this context.

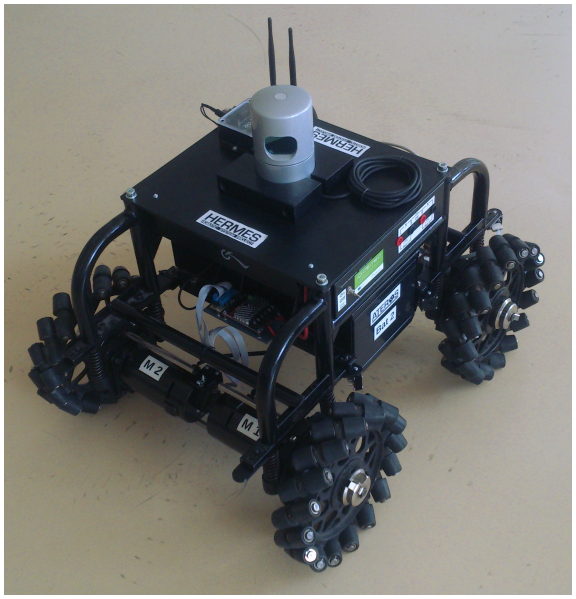


Fig. 4.1: The omnidirectional robot Hermes.

4.1 Hardware design

This section covers all physical equipment of the robot, whether it is machinery or electronic devices. The robot was supposed to operate in an indoor environment and bear advanced telemetric sensors and some computational platform for control and data processing. There is a large amount of possible constructions and even larger selection of different parts available from commercial subjects. Comprehensive overview of the possibilities is out of the scope of this work and can be found for example in [65] among many other sources. Here will be discussed only the particular solutions with justifications of the critical decisions, but without exceedingly wide context.

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. The following subsections describe, how these parameters were achieved.

4.1.1 Mechanical construction

Taking into account the requirements above, 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. The construction is very sturdy, mainly made out of welded iron pipes. The wheels are the Mecanum style [66], with passive rollers mounted in a 45 degree angle with respect to the axis of rotation. Such wheels are obviously not sufficient for even a mildly hard terrain, but in a man made environment, where the flat and hard surfaces are mainly present, this disadvantage is negligible.

Omnidirectional robots are very popular among the research community [66], [67], [68], [69] and the control of the wheel frame is therefore well explored. The

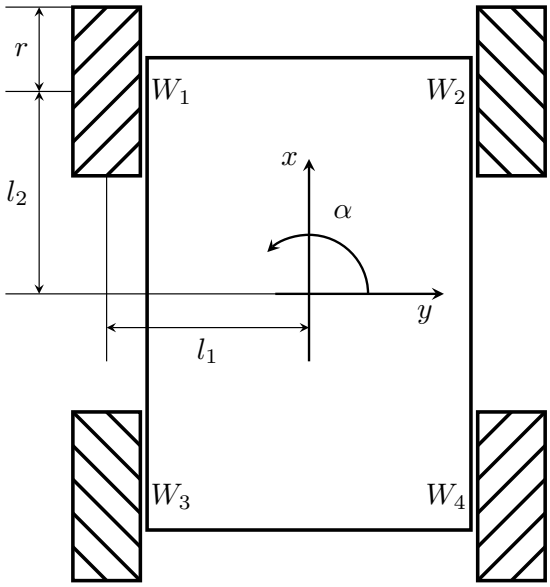


Fig. 4.2: Schematic depiction of an omnidirectional robot with a coordinate system attached. Positive direction of rotation of the wheels is that which moves the robot in the positive direction of the x axis.

Figure 4.2 shows a schematic depiction of an omnidirectional robot with the important dimensions and coordinate system attached. As the robot can freely move in any direction and perform the rotational motion at the same time, its locomotion needs to be controlled with a vector of three variables for the velocity in each degree of freedom. [70] provides a direct calculation of the wheel speeds, corresponding to the given control input:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & 1 & -(l_1 + l_2) \\ 1 & -1 & l_1 + l_2 \\ 1 & -1 & -(l_1 + l_2) \\ 1 & 1 & l_1 + l_2 \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ \omega_\alpha \end{bmatrix} \tag{4.1}$$

where ω_i is the angular speed of corresponding wheel W_i and the control vector v_x, v_y and ω_α represents requested speeds in robots local coordinate system.

Aside from the locomotion system, the mechanical construction is very unsophisticated. 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.

4.1.2 Electronic power and control systems

Like majority of other mobile robots for laboratory purposes, Hermes is powered by a set of electrical accumulators. The wheel frame is designed for two lead-acid batteries, providing 24-28 V in series at the capacity of 26 Ah, which is enough for the required operation time. DC motors for each wheel are powered directly from the batteries. The rest of the electronics works either with 5 V or 12 V input, which is provided by a self made, dual channel switched-mode power supply. It is capable to deliver 10 A on both channels and is equipped with voltage and current measurement for monitoring and feedback regulation.

Motors are regulated using the RoboClaw ST 2x45A controllers[1], each taking care of two motors as depicted in Figure 4.3. The motors are equipped with quadrature encoders mounted directly on the motor shaft and planetary gearboxes. The RoboClaw controllers are capable to exploit the information from encoders for speed control, so the equation (4.1) is directly applicable. The controllers also provide variety other control modes and feedbacks (e.g. operational current).

[1]The detailed technical manual for the RoboClaw drivers by Ion Motion Control is available from: `http://www.ionmc.com/downloads` (25.7.2017).

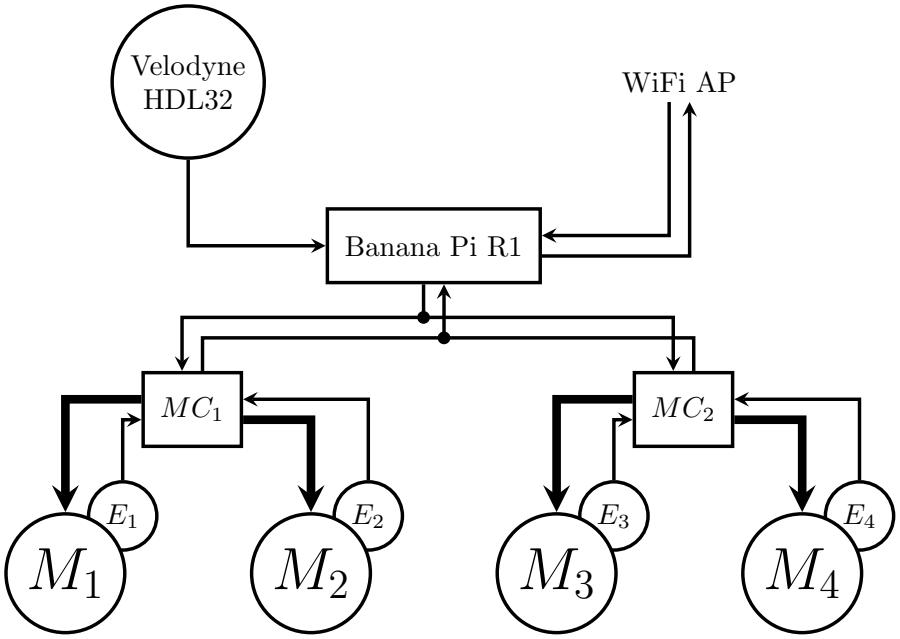


Fig. 4.3: General electrical scheme of the Hermes robot. The emphasis is put on the communication, data flow and control structure. Power distribution is described in the accompanying text. (M_i - motor, E_i - encoder, MC_i - motor controller)

As an embedded platform for communication and computation was selected the BananaPi R1 router-board[2]. It contains a decent dual-core ARM processor and 1 GB of RAM, which should be enough for basic operation. Much more important is the connectivity, where the board is among the top products in its category. 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. This is especially beneficial for larger data acquisition systems. The board also contains a SATA port for hard disc connection. For the mobile robot, a solid state drive was used due to it higher tolerance to mechanical vibrations and shocks. Various operating systems can be run on the board, e.g. Android, OpenWrt and many Linux distributions. A Debian based distribution was selected for the robot, because of its stability and community support.

Communication with the rest of the world is solved using the WiFi connectivity. The robot is configured as an access point to which the other devices can connect. The on-board adapter does not support neither high speed, nor the range, but for laboratory experiments is was sufficient. Of course, a better device can be connected through USB or Ethernet in case of any problems.

[2]The BananaPi R1 board is developed by the Sinovoip company. For further information see: `http://www.banana-pi.org/r1-download.html` (25.7.2017).

4.1.3 Sensory equipment

Sensory equipment is probably the most important hardware for SLAM. This work is focused on point cloud processing, therefore the Hermes robot is equipped with a laser scanner (range finder, LiDAR[3]). Various other means of acquisition of geometrical data are available, but none of them provides direct measurements of the quality comparable to light emitting range finders.

The laser scanner actually used is the Velodyne HDL-32 model [4]. 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 contains 32 lasers firing in a vertical range of $[-30.67, 10.67]$ degrees. As the head of the sensor rotates the lasers point to a different direction and a point cloud is generated. The scanner provides approximately two thousand of measurements per laser per turn, which, with the speed of ten revolutions of the head per second, results in a data flow of 700 000 points per second. It is connected using the Ethernet interface to the BananaPi board, where the raw data from the scanner can be stored on the hard disc. 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, where the vectorization algorithm for point cloud processing is optimized.

4.2 Software tools

Aside from a physical form, each robot is defined by its AI. In the context of this work, the usage of the phrase *artificial intelligence* is quite stilted, but still, a lot of software tools and libraries were developed during the years, either as the means for running experiments, or as a practical implementation of a theoretical thoughts.

4.2.1 Distributed control system

The first project developed for the purposes of this thesis was the control system for the robot Hermes. Its GUI (graphical user interface) part runs on a Windows machine (but can be compiled for Linux as well) and the control routines run on the BananaPi board with Linux. The C++ programming language was chosen as a main development tool because of its portability, flexibility and wide usage in practice.

[3] LiDAR was originally proposed as a combination of words *light* and *radar*, but today an acronym *Light Detection And Ranging* is usually used.

[4] The device was developed by Velodyne LiDAR Inc and the detailed technical informations are available from `http://velodynelidar.com/hdl-32e.html` (25.7.2017).

For system specific operations and graphical interface, the Qt5 library[5] was selected, as it is freely usable (among other possibilities, the most suitable seems the GNU LGPL v.3 license [6]) and covers a wide variety of platform specific matters under the unified interface. Many more tools were used during the development (shell scripts, C, C#, Matlab, ...), but their importance is lower.

Clearly it was possible to quickly put together a disposable piece of code, which would provide a remote control and a basic data logging functionality. On the other hand, such programs are hard to maintain and because different features are added and removed during the development, it is advisable to make the software reasonably modular, especially if there is a chance, that it will grow in complexity by the time. The architecture of the control system was therefore made reasonably flexible.

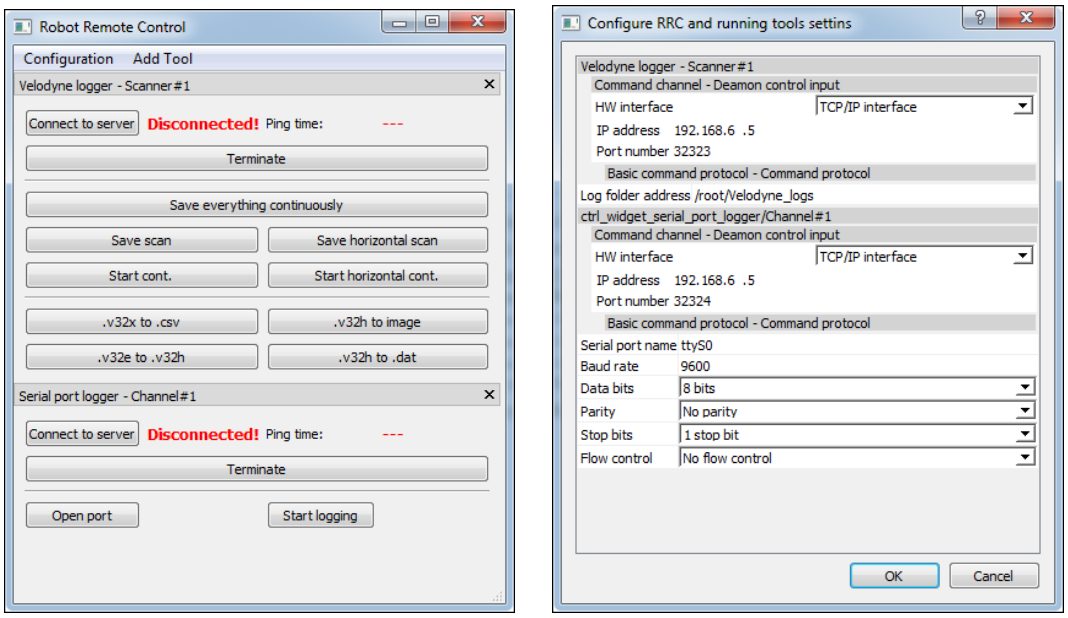
The key concept is separation of tasks into a set of standalone programs (either GUI or background running daemon processes), which communicate through predefined communication channels. This allows to run different parts of the system on different machines, communicate using various hardware interfaces and implementation 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.

So far, there are only the interfaces for TCP/IP and RS232 and the command protocol for control of the background programs. It is sufficient for the current needs and the flexible nature of the approach already proved itself to be useful, when the robot was operated from different devices. It is expectable to be even more successful, once the communication network will become more complicated. So far, there is the control application (see Figure 4.4a) and four daemons (MotorControl, SoftwareTrigger, SerialLogger and VelodyneLogger). The GUI application can manage arbitrary number of daemon processes and for each of them a separate entry in the setting tool is generated (Figure 4.4b).

The distributed control system was used in other experiments performed by our laboratory as well, for example in precise georeferencing [71].

[5]Qt5 library at the time of writing (25.7.2017) was developed and maintained by The Qt Company and available from: `https://www.qt.io/developers/`.

[6]The GNU Lesser General Public License is one of the free software licences published by the Free Software Foundation. The version 3 is available from `https://www.gnu.org/copyleft/lesser.html` (25.7.2017).



(a) Control user interface.

(b) Settings user interface.

Fig. 4.4: Screenshot of the application for remote control of the robot Hermes and its internal processes.

4.2.2 Vector processing library

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 the rest is very different. First, it does not use any special high level libraries due to portability reasons. Only the *standard template library* was used, as it is widely available and provides highly optimized and helpful code.

The library itself builds on templates as well 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. The library was also used in the benchmarks comparing processing times of various algorithms and implementations in the following chapters. 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.

The following overview presents the most important classes in the vector processing library. Although it works well, it is far from maturity. Many thoughts on architecture and a lot of programming craftsmanship will be necessary to promote

it from its rather experimental state.

Basic geometry

Vector2D is a simple class representing a vector in a two-dimensional space. All functions used in vector algebra are defined and some extra useful routines are provided as well. It is also extensively used for representation of a point in a 2D space.

LineSegment2D naturally represents a line segment. It is defined by the begin and the end points and can be can be manipulated using various transformations and tested for overlaps, crossings etc. Although it is redundant, the class also contains a unit direction vector of the line segment, because it significantly speeds up several computations, which seems to justify the extra memory footprint.

Transformation2D represents a rigid transformation limited to rotation and translation. Similar to the previous, it is optimized for speed, so the internal sine/cosine functions are precomputed in advance. The angular parameter therefore occupies the space of two floating point numbers instead of one.

Point cloud processing

EAR_Segmenter is a functor implementing the point cloud segmentation algorithm described in Chapter 5. The prefix EAR stands for *Euclidean* (distance metric), *Adaptive* (examined point surroundings), *Retroactive* (rewrites pertinence on merging clusters).

TLS_Vectorizer provides implementation of the classic total least square vectorization of given data set. Only a single regression line is fitted into the whole point cloud, so it is not advisable to use it on clouds with unknown shape.

FTLS_Vectorizer is a functor written in accordance to the fast vectorization algorithm, which will be described in Section 6.3. The class contains buffer arrays for internal variables, data structure for user settings and straightforward interface for usage with the rest of the library.

AFTLS_Vectorizer is internally the same as the FTLS_Vectorizer, but adds the augmentation described in Section 6.4, which leads to better results in certain situations. The interface of the class is identical to the previous one to allow easy interchangeability.

TLS_Structures is a support class, which covers internal structures of the previous three total least squares algorithms. It does not provide any standalone functionality on its own.

SLAM specific data structures

EdgeRich2D is an aggregate class containing all data relevant to a single edge in a map, including raw point cloud, cumulative sums (see Chapter 6 for details), line segment approximation, pointer to an observation structure and possibly a user defined annotation.

EdgeRaw2D can be created from the EdgeRich2D object as its lighter alternative for further processing. It contains only the cumulative sums, line segment approximation and a pointer to the parent EdgeRich2D. It is much more compact and contains all information for the total least squares computations.

Pose2D is a simple structure containing information on position and azimuth in given coordinate system.

Observation2D contains a data from a single point cloud measurement. Aside from the cloud itself, there is an information on expected pose of the robot at the time of data acquisition, time stamp, an array of EdgeRich2D objects obtained through segmentation and vectorization and a set of outliers, which were measured, but were not suitable for further processing.

Vector data sets similarity and registration

LS2D_Observer is dedicated to generation of an observation in a given virtual environment. The class currently does not contain any parameters and the observation precisely follows the input data. With appropriate expansion, the class could be used for simulation of laser scanning in artificial virtual terrain.

EdgeSetComparator implements the line segment similarity criterion described in Section 7.2. The object stores the precomputed sums and provides interface for sequential addition and removal of line segment pairs, batch loading and of course, evaluation of the criterion.

MatchLineSegments2D has a similar user interface as the previous class, but the internal functionality is implemented in accordance to Section 7.3.

LS2D_CorrFinder is a functor, which searches for corresponding line segments in data sets using the algorithm described in Section 7.4. All functions described are fully implemented and work as expected. On the other hand, the development is still in progress, so some changes are possible in the future.

Import-export

LaTeXExport is a class dedicated to generation of LaTeX output from the experiments. Many graphs, point cloud renderings and other diagrams in this work were

obtained using this tool. It turned out to be extremely useful during debugging of the methods as well.

CSV_ImportExport is currently not implemented, because due to an old design decision, the CSV[7] interface is scattered around the classes, which implement the data structure to be saved or loaded. It will be extracted and merged into a new class in the future revision of the library.

4.3 The ATEROS robotic system

The acronym ATEROS stands for *Autonomous and Telepresence Robotic System*, which is developed by our laboratory. 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. This thesis is aimed to enhance autonomy of the robots even more.

The system can be used in a variety of situations, which require different robots. The outdoor reconnaissance robots are depicted in Figure 4.5. The possible missions are: autonomous 3D map building, environment contamination measurement, urban search and rescue (US&R) and other military and civilian applications.

Fig. 4.5: ATEROS - the outdoor reconnaissance robots and an operator with a virtual reality head set.

[7]CSV refers to the coma separated value file type, frequently used to store data in table-like, human readable format.

5 POINT CLOUD SEGMENTATION AND FILTRATION

Since the data are obtained in an arbitrary environment and 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. The notions *segmentation* and *clustering* are used interchangeably in further text[1].

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. Nevertheless, the methods are applicable as well and provide a good inspiration for development of a new method.

Surveys [72] and [73] provide a useful taxonomy of segmentation methods available. *Edge-based* algorithms are not easily convertible to work with two dimensional data, so will be omitted in the following review. The same applies for *model fitting* methods, because presence of outliers would endanger their functionality and line segment fitting is the matter of further processing.

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, seed start to expand to the points in the neighbourhood and as long as a condition is met, the cluster grows. Stopping condition can be either given by quality of approximation by some kind of geometrical primitive (typically a plane) [74], surface curvature [75], or anything else [76]. Initial distribution of points is crucial for the good results and does not seem to be completely reliable in highly structured environments. Opposite is the top-down method [77], which starts with a single cluster for all points and than divides it into smaller ones, until each of them meets a condition from those listed previously. Obvious difficulty is the question, whereto divide the unsatisfying cluster. A hybrid approach utilizing both principles is possible as well [78]. Iterative nature makes these algorithms rather unpredictable. If working

[1] *Segmentation* is mostly used in the computer vision domain and refers to identification of regions of an image with some common property. *Clustering* is more used in statistics and machine learning, where grouping of high-dimensional data is studied. Point cloud processing lies somewhere between those fields and techniques from both can be successfully applied. Using these words as synonyms thus should not be inappropriate in this case.

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 is the K-means [79]. The original method works with minimization of the squared distances between the center of the cluster and the points belonging to it. Since points in a point clouds do not concentrate around a single center, but around lines or surfaces, the method has to be partially adjusted. Nevertheless, its popularity is very high [80], [81], [82]. Another possibility is hierarchical clustering, which is similar to region growing, but the initial seed distribution stems from different principles and probably provides better results [83], [84].

Many other interesting approaches exist [85], [86], [87], [88], but these are focused on 3D clouds as well and propose elaborated algorithms, which could not be justified in a much simpler 2D case. An interesting subject is the possibility of parallelization of the procedure as shown in [89] and [90]. Though greatly beneficial for large data streams, 2D scans probably should not need such optimization.

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 and lead to over-segmentation[2]. On this basis, an algorithm, independent on line fitting and working sequentially on an ordered point cloud, was devised.

5.2.1 Sequential segmentation

The overall diagram of the algorithm function is depicted in Fig. 5.1. At input, there is a raw cloud of N points as obtained through a laser scanner and several parameters for setting of the working conditions. Distance threshold δ is a maximal spacing between points belonging to a single cluster and corresponds to an amplitude of the noise expected in measurements. Constant K 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.

The algorithm proceeds from one point in an ordered point cloud to another and checks if the K of the previous points are closer or farther than the distance δ. If K is larger than i, it stops at $i = 1$. After the comparison finishes, three

[2] *Over-segmentation* refers to a situation, where too many segments are found and generalization process was terminated too soon. *Under-segmentation* is the opposite and generally means loss of significant information

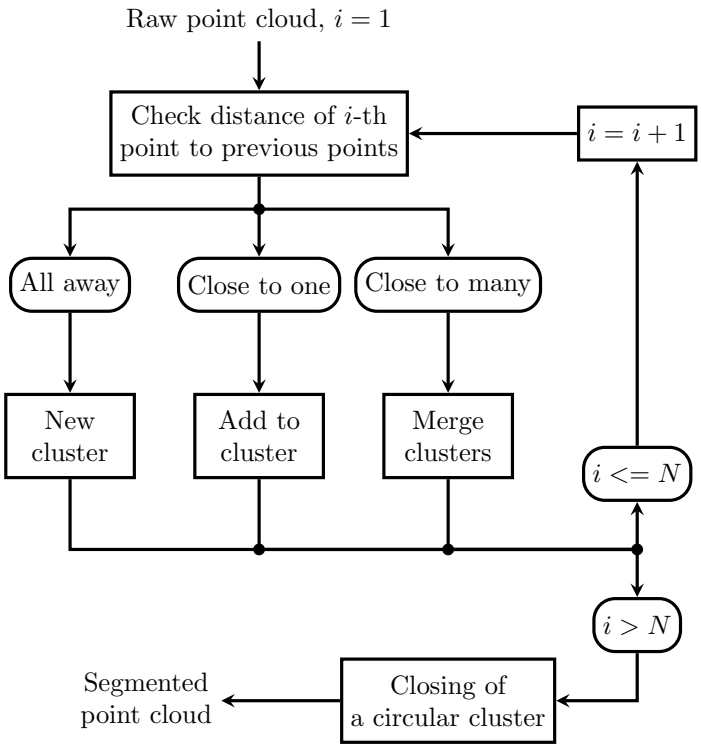


Fig. 5.1: Schematic diagram of the segmentation algorithm.

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 the 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 the given clusters and the working point are joined together and form a new bigger cluster unifying all of them. Once any of these operations is finished, the algorithm proceeds to the next point and the process continues in a next iteration.

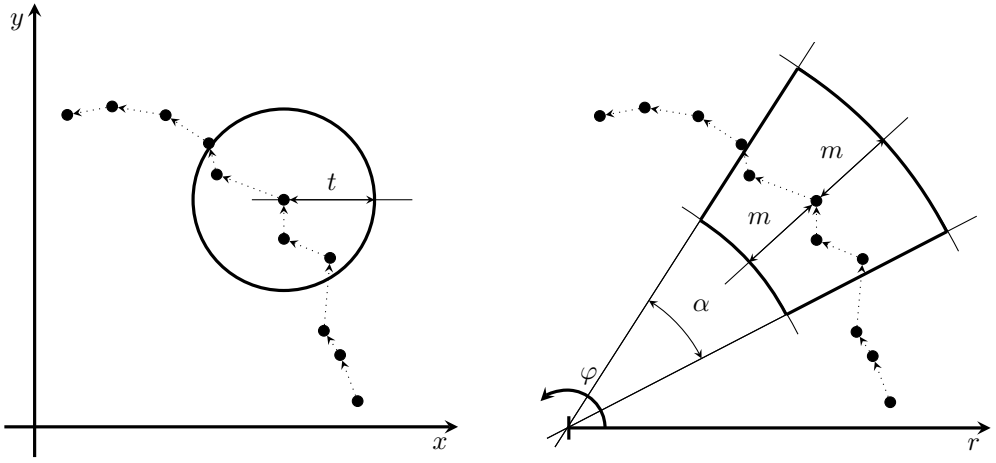
In the end, once the last point is processed, there is a final check, if the cloud is circular and the points from the beginning and the end may belong to the same cluster. In case this is true, the first K points are examined again, now taking the negative indices from the end of the cloud. This precaution ensures, that no cluster is split due to the sequential storage of a circular cloud.

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, ready for the vectorization process. Though the algorithm is quite simple and computationally efficient, there are some subtleties worth special attention.

5.2.2 Caveats of the clustering process

The first issue stems from an unspecified format of the point cloud coordinates. In Section 3.1 an importance of unification was already mentioned, but the exact decision is implementation dependant and may vary. In practice, the coordinate system will be either Cartesian or polar and for each of them a different distance function is efficient. Cartesian coordinates are familiar and regular Euclidean distance will suffice (see Fig. 5.2a). For direct computation in the polar system, a bare comparison of angular and radial distance with thresholds is sufficient and saves the costly computations involving trigonometric functions (Fig. 5.2b). The algorithm is otherwise very light-weight and coordinate conversion or similar operations would have significant impact on performance.



(a) Neighbourhood definition for point clouds in Cartesian system of coordinates.

(b) An adapted neighbourhood for polar coordinates.

Fig. 5.2: Differences in proximity evaluation in Cartesian and polar coordinates.

In practical measurements, there is another effect to be taken into account. Principle of operation of the laser scanner implies, that the points measured on a flat surface are not evenly spaced and become sparse with growing distance from the scanner. An adaptive threshold was found to be useful to partially suppress this effect. In this application, the threshold was scaled linearly with the distance between the working point and the position of the scanner. For the Cartesian

coordinate case in Fig. 5.2a this means the t length. In polar coordinates (Fig. 5.2b) only the m distance is scaled, since α is an angular measure and is not influenced by the effect described. Different scales and thresholds are possible, a widely applicable example for Cartesian coordinates is as follows:

$$t_i = 2\pi \frac{K}{N} |r_i|, \tag{5.1}$$

where N is the number of points in the cloud and $|r_i|$ is the distance of the particular point from the beginning.

A third issue is related to the too close or too distant measurements and the linear scaling of the distance threshold. In practical situation, a robot is usually large enough to prevent scanning of obstacles from a very close proximity and a large distances in the indoor environments are rarely encountered as well.

If the laser scanner is near a surface it measures, the scaling mechanism can reduce the threshold to be lesser than the noise of the measurement (for consequences see Figure 5.3). An opposite happens at large distances, where clearly distinguishable clusters are falsely merged, because the threshold have grown enormously (Fig. 5.4). Both can be suppressed by introduction of lower and upper constraint to the threshold and perform the scaling operation only in these margins. Neither of these numbers is some arbitrary constant for manual fine-tuning of the algorithm. The lower bound directly corresponds to the accuracy of the distance measurement device and the upper bound is given by the minimal required point density in a cluster, which should be specified in advance anyway.

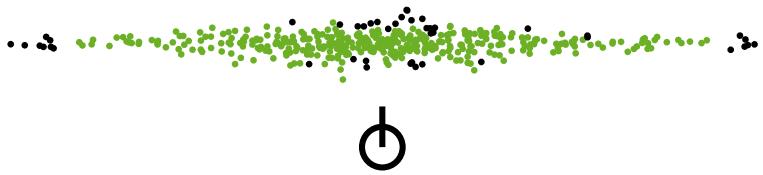


Fig. 5.3: Segmentation with an adaptive threshold - too close objects. If no bottom limitation for the adaptive threshold is specified, it can become lower than the sensor noise for very near objects. If this happens, the points at the periphery of the cluster are detected as false outliers.

The final remark origins at the same phenomenon and is related to sampling, i.e. spatial density of measurements[3]. This fact has a huge impact on security of the mobile robot operation, because too much generalization in scan segmentation

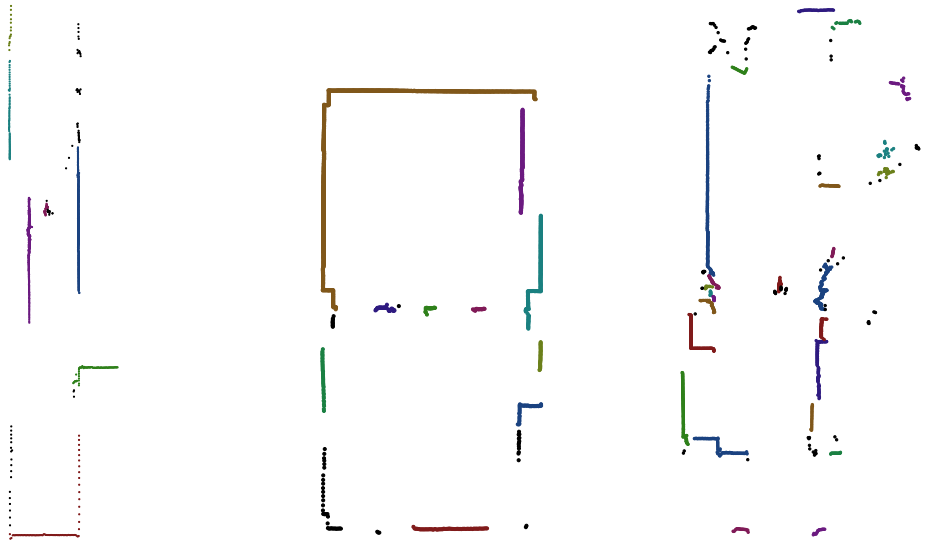
[3]Direct connection to the Nyquist–Shannon sampling theorem from the field of digital signal processing is evident, but unfortunately, no dedicated work on this theme in conjunction with point cloud segmentation was found during the literature exploration.

Fig. 5.4: Segmentation with an adaptive threshold - too distant objects. If no upper limitation for the adaptive threshold is specified, it becomes too permissive at large distances as depicted in the right side of the image. In such a situation, outliers may not be properly filtered out and cluster separation may fail as well.

and vectorization can lead to omission of narrow objects (chair legs, steel netting) and result in collision. This work does not address the issue any further, but since safety in robotics is of great importance [41], it is definitely a worthy direction of research.

5.3 Experimental verification

Any algorithm considered for practical usage needs an extensive testing. 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



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

Fig. 5.5: Three scans from the real indoor environment. Clusters are depicted in colors, while outliers are jointly rendered in black.

experiments were processed in Cartesian coordinates using the same parameter settings. The number of retroactively tested neighbours K was set to ten, the threshold value t was adaptively scaled using the formula (5.1) and the clusters were considered outlying if they contained less than fifteen points. The lower bound of the threshold was five centimetres and upper one twenty centimetres.

The first set of experiments depicted in Fig. 5.5 was held in an indoor environment of Faculty of Electrical Engineering and Communication in Brno, using the equipment described in Chapter 4. The narrow corridor in Fig 5.5a 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

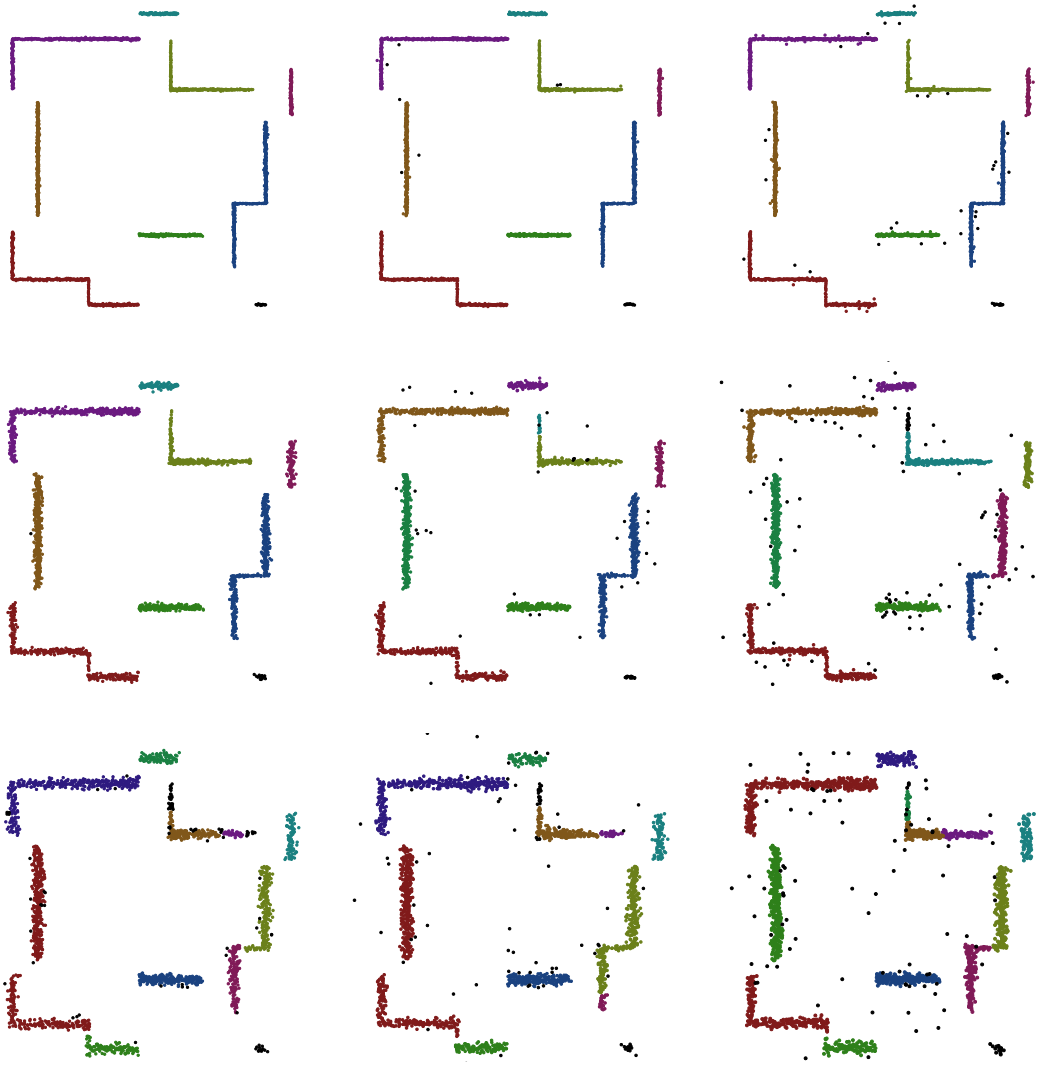


Fig. 5.6: Simulated scans of idealised room with different levels of noise (grows from top to bottom) and portion of outliers (grows from left to right).

continuous and father away, they fall apart into solitary outliers. Figures 5.5b and 5.5c 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 set of synthetic experiments in Figures 5.6 and 5.7 was performed to explore an influence of the noise level and the ratio of outliers in a scan on the quality of segmentation. In practise, several laser scanners would be required and differences in one parameter only would be hardly achievable. Simulation is therefore very beneficial in this case.

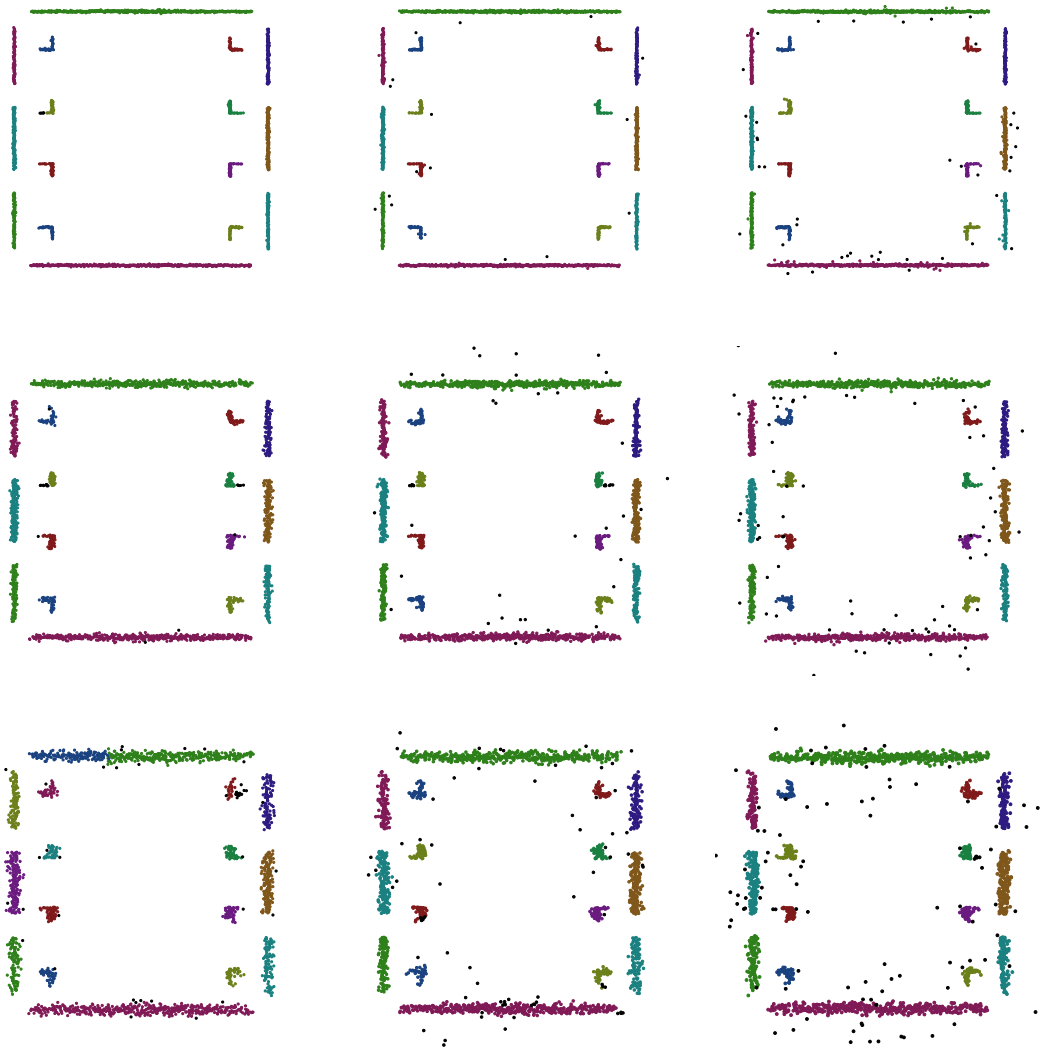


Fig. 5.7: Simulated scans of a room with small objects. Again, different levels of noise (grows from top to bottom) and portion of outliers (grows from left to right) are tested.

The lowest noise level (top row) corresponds to the real laser scanner used in the practical experiments and the segmentation is flawless in this case. As the level of the noise increases (middle and bottom row), false splits of the clusters emerge. This happens, when the noise is so significant, that a real order of the points as seen in the picture (given by the order of their perpendicular projections on the edge they belong to) is different to that in which they were acquired (e.g. given by the angular coordinate for the rotating scanners). The segmentation algorithm works with the order of acquisition and there the cloud could really seem interrupted. In practise, this kind of problems is rarely present, because the recent scanners have much smaller noise, than that in the simulation, which limits probability of this issue to arise.

Outlier rejection, on the other hand, is satisfying and works steadily at all noise levels, even the distant fault points are correctly separated and did not result into false cluster merging (see the bottom right scenario in Fig. 5.7). Small cluster are segmented properly, only in some cases the peripheral points were rejected and marked as noise.

5.4 Résumé of the segmentation algorithm

As illustrated above, 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. Its parameters are an estimate of an amplitude of the noise present in the point cloud, a maximal permitted distance of points in a cluster and a number of retroactively tested neighbours K, used during the search for pertinence of the point to the already existing clusters. Experimental verification in the previous section showed good results in situations with proper algorithm settings, highly noised scans had the K parameter deliberately low to illustrate possible problems with false splits of evidently continuous clusters. 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 [91], [92] and [93]] written by the author of this thesis on the subject. For the purpose of this thesis, 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 [94], [95], [96], [97]. Closely related is polyline simplification [98], a procedure widely used in cartography [99], [100] to obtain maps with different levels of details [101] and effective storage of the data [102]. Another application, situated on the boundary between robotics and cartography, is motion tracking [103] of vehicles [104], people [105] or animals [106]. The last major domain in the given context then consists in image processing [107], [108], where vectors are used to extract geometrical information from a bitmap image [109], [110]. Besides these main fields, there are also other special applications, such as cloud detection in meteorology [111] and trajectory teaching to industrial manipulators [112].

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; for example, the above statement does not hold true for a point cloud composed of several scans acquired from different locations, but one scan acquired by a laser scanner from one location already exhibits this property. If it is possible to treat each scan separately, the order of points in the list can be exploited. Naturally, borders in cartography also act as an ordered list of points. Edges in a bitmap image are usually formed by a chain of adjacent pixels. Intersecting edges can be always split into a set of such chains, namely ordered lists. It is clear that the "ordered list" assumption is somewhat constraining, but a wide range of practical applications either directly satisfy this condition or can be converted into a form which does so.

6.1.1 General vectorization algorithms

There is a large number of algorithms to facilitate both the vectorization of point clouds or bitmaps and the simplification of polylines. Based on their input data, all of these methods can be roughly divided into two groups. Algorithms within the former group are general enough to be capable of processing any input data, regardless of whether such data are ordered or not.

A widely used algorithm of this type is the *Hough transform* (HT) [113], which finds application in image processing and, occasionally, robotics. Despite its more than fifty-year history, the algorithm is still being developed, and many innovations can be traced. Interesting topics related to high quality vectorization are connectivity-enforcing HT, described in [114], and the 3D point clouds to 2D maps matching system [115].

The *RANSAC* method [116] 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; however, it constitutes an integral part of more complex systems for robust point cloud processing, [94] and [95].

Some recent algorithms for line segment extraction are based on genetic algorithms [117], neural networks [118], Delaunay triangulation [119], and multiscale edges [120]; importantly, many other procedures can be found in literature, especially in that focused on image processing.

A comparison of the traditional HT with new algorithms used in this field is available in [121]. Moreover, a number of methods have been particularly developed for line extraction from the arbitrary point clouds resulting from 3D laser scanning. Very promising results are presented in [122], where the authors used the truncated Fourier series to extract information about the curvature of a surface and then employed the obtained data to identify the feature lines. Algorithms in this category are generally computationally intensive and, as such, rather unsuitable for real-time vectorization.

6.1.2 Point eliminating algorithms

The second category of algorithms needs ordered input data to work correctly. A wide family of algorithms based on *point elimination* (PE), which are mainly used in cartography to simplify maps with a high scale, belongs to this category. 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. The computational complexity and quality of the approximation of these point eliminating algorithms is examined in depth in the survey paper [98], where the authors present several metrics to enumerate distortion, displacement and even the visual differences between the original and the simplified vector image.

Probably the best known point eliminating algorithm is the *Douglas-Peucker algorithm* (DP, also known as split-and-merge) [123], further improved in [124]. In [96], it was the fastest method among others and in [98], it was the best approximating method in the point eliminating algorithms family. Its expected computational complexity is $O(n \log n)$, which, in combination with the low computational costs of each step, makes it fast enough for online point cloud processing. The algorithm was also parallelized to perform even faster [125].

Less computationally complex, but also less accurate, is the *Zhao-Saalfeld sleeve-fitting algorithm* [126]. One of the fastest algorithms used in practice is the *Reumann-Witkam algorithm* (RW), which was published in [127], only a year after the DP algorithm. Computational complexity is $O(n)$ and each iteration consists of only one point to line distance computation, which makes it very fast, although the quality of an approximation is not nearly as good as for the DP algorithm, as shown in [98].

6.1.3 Least squares vectorization algorithms

To overcome the problem of losing information when using point eliminating algorithms, linear regression is often used. This method also needs ordered data on input. The most common case is a standard least squares regression, which provides good results at a reasonable computational cost. It is frequently applied in feature tracking systems in conjunction with other algorithms, such as the extended Kalman filter in [97]. It is also often used to improve the precision of inaccurate results obtained by other means, for example, the paper [6] describes a combination of line regression and HT. Computation is usually done in Cartesian coordinates, but the authors of the algorithm [128] showed a way of computing line regression directly from the point cloud data in polar coordinates, which might be useful under specific conditions.

The *Incremental algorithm* (INC), as described in [96], was the best known representative of linear regression based algorithms for a long time. Orthogonal least squares fitting is used to find the regression line. The algorithm gradually iterates through the ordered data and in every iteration it computes a linear approximation of the tested range of points. If the error metric is in the limit, the algorithm continues to another point. 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 point eliminating algorithms. Performance comparison in [96] confirms this expectation.

Correctness and precision evaluation in [96] gives similar results for the DP algorithm and the least squares fitting. The point clouds used here are clearly obtained with a laser scanner with low noise, which probably leads to the presented results. In a less favorable situation, with more noise present, the linear regression approximates the given points better, as reported in [129], [130] and [131].

Contrary to point eliminating algorithms, the least squares approach enables a description of the uncertainty of the approximating line with respect to the uncertainty of measurement of the points in the cloud. Firstly, the results are in [132] and a more general treatment of this topic is described in [133].

6.1.4 The problem in threshold based vectorization

Every vectorization algorithm has an error function evaluating quality of the approximation it produces. The general algorithms usually search each line segment in the data set separately, which leads to minimization of the error function in each case. Conversely, the algorithms for the sorted data, exploit the continuity of the given point cloud, traverses through the data set and set up the break points of the approximating polyline. This approach is prone to give suboptimal results as shows the following example.

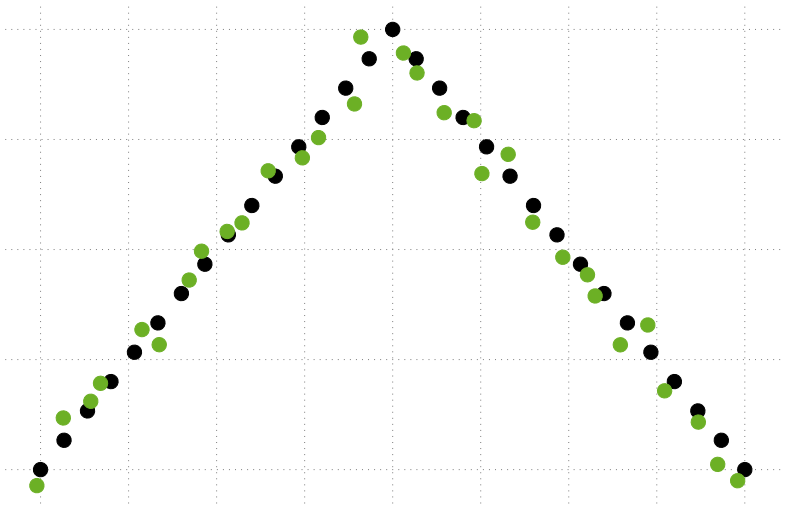


Fig. 6.1: An example of two simple point clouds. The black one is precisely generated to form a sharp corner, while the green one has a realistic noise added.

Fig. 6.1 displays two point clouds: one precise (black) and one noised (red). Both point clouds are ordered, as if they were obtained by a laser scanner. Fig. 6.2

shows the characteristics of the error function generated by a TLS method for a given number of points, counted from the beginning of the cloud. As expected, the first half of the precise point cloud is vectorized with a zero error, since the points lie exactly in a line. After the midpoint, the error starts to grow monotonically. Though the whole shape is quite similar, the first half of the green characteristics is corrugated, contains local minima and the break point, where the error starts to grow rapidly, is not defined as well as in the previous case.

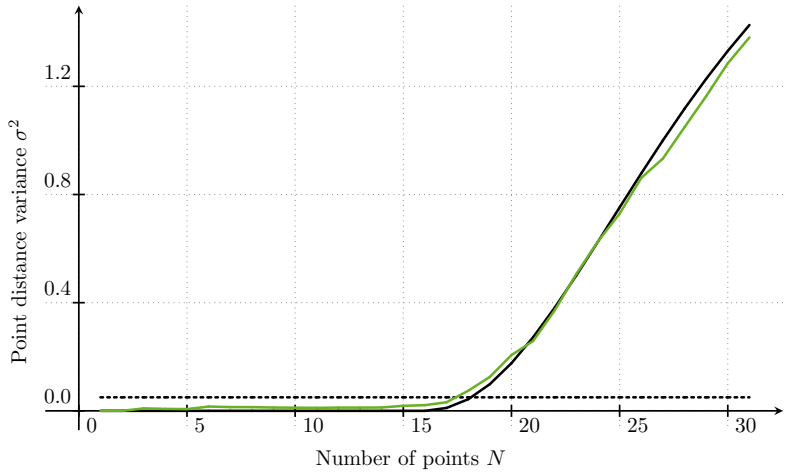


Fig. 6.2: The error functions generated for the various number of points from example point clouds in Fig. 6.1, obtained by the TLS vectorization algorithm. The black characteristics belongs to the precise data set and the green chart to the noised one. The dashed line illustrates a defensive choice of a threshold level for a robust vectorization.

All of the fast vectorization methods (whether PE or TLS) employ a threshold, which is compared to the error function in each iteration and when exceeded, the break point is set up and a new approximation is started. Due to the fact, that the error characteristics is not ideal, the threshold must be set with appropriate reserve, to ensure robustness and immunity to the noise. Such a defensive choice of the threshold leads to higher error than necessary for every line segment in the approximating polyline, except the last one, as can be seen in Fig. 6.3.

The negative influence of the defensively chosen threshold was slightly magnified for the illustrational purposes in this case, but even without that, it is a serious drawback of all of the fast vectorization algorithms. This effect manifests regardless the algorithms traverse the point cloud incrementally (INC and RW), or perform some sort of an adapted binary search (FTLS and DP). It is also impossible to suppress it by calibration [134] or identification [135] of the laser scanner, because the described problem is an inherent feature of the TLS and PE algorithms, not a

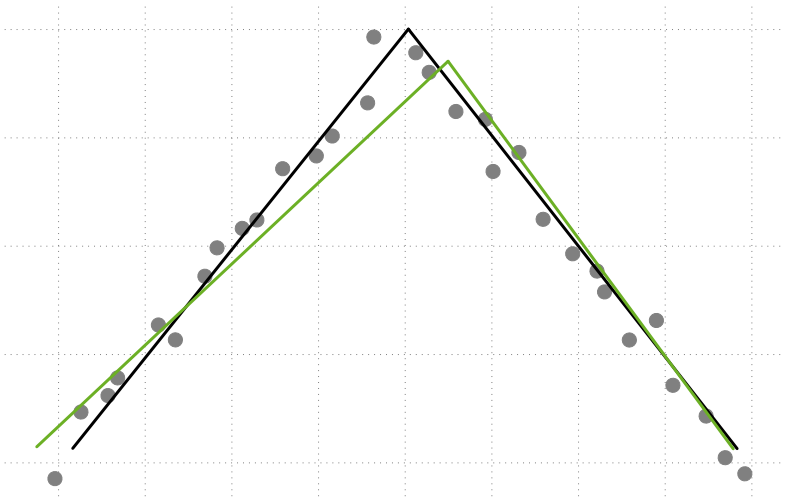


Fig. 6.3: Optimal (green) and suboptimal (black) vectorization of the noised point cloud (gray). The green and black line segments approximating the second half of the point cloud nearly overlap each other, while the first part is flawed by the threshold problem.

flaw of the input data.

6.1.5 Solution to the accuracy versus speed dilemma

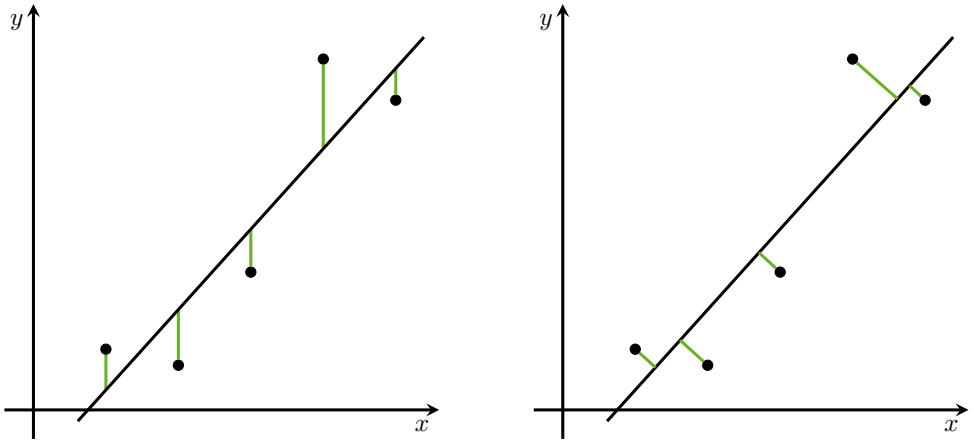
According to the facts discussed above, there are basically two categories of algorithms suitable for real-time operation: point eliminating methods, which are fast and not so accurate, and linear regression methods, which provide the best approximation, but are generally slower. Even worse, both of 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.

All examples are related to point clouds, but the algorithm itself can process any given ordered list of points, regardless of its origin.

6.2 Theoretical background of the total least squares fitting techniques

As mentioned above, the least squares method for the vectorization of point clouds is an accurate and widely used approach, however, a simple linear regression is insufficient. It presumes only one variable to have an error in observation and the other to be precisely known, as shown in Fig. 6.4a. This is definitely not true for a 2D laser scan, where both coordinates have a certain variance. Also, the approximation line in a simple regression cannot be parallel with the y-axis and nearly parallel lines are highly inaccurate, which is another significant downside of this method when considered for use in point cloud vectorization.



(a) Simple linear regression. (b) Orthogonal linear regression.

Fig. 6.4: Linear regression methods.

In the following text, Cartesian coordinates (x, y) are going to be used, and measured points in the cloud are presumed to belong to an ordered list of N pairs in the form $(x_i, y_i), i = 1 \dots N$, where N is the number of processed points.

6.2.1 Traditional approach

The solution to the problem of variance in both coordinates is the total least squares (TLS) method, which minimizes the square of the shortest Euclidean distance between a point and the regression line (see Fig. 6.4b). The main idea has been known for a long time [136] and the theory behind it is well described, but is very general [137] and too complex for the purposes of a fast vectorization algorithm. A special case of TLS is the *Deming regression* [138], and an even more special case (for equal

variances of both variables) is the *orthogonal regression*, which corresponds well to point cloud vectorization. This was used, for example, in [96], [139] and [133].

Both the Deming regression and orthogonal regression are usually derived for the equation of a line in the form $y = kx + q$, which is mostly sufficient, but for point cloud vectorization it is not, because it is not able to correctly describe a line parallel with the y-axis. This is the reason for derivation of the orthogonal regression for a neutral format of the equation of a line.

Probably, the first paper where this method is used for point cloud processing is [139]. The authors took the following equation to describe a line:

$$x \sin\theta - y \sin\theta + \rho = 0 \tag{6.1}$$

and derived the solution:

$$\theta = \frac{1}{2} \arctan \left(\frac{2 \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N} (x_i - \bar{x})^2 \sum_{i=1}^{N} (y_i - \bar{y})^2} \right), \tag{6.2}$$

$$\rho = \bar{y} \cos\theta - \bar{x} \sin\theta, \tag{6.3}$$

where:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{N} x_i, \qquad \bar{y} = \frac{1}{n} \sum_{i=1}^{N} y_i. \tag{6.4}$$

This form of equations is the base of the traditional incremental algorithm for TLS vectorization and is used in papers [132] and [133] focused on uncertainty enumeration. The equations are also suggested in [128] as a computationally more effective alternative to a direct computation in polar coordinates. The proposed equations are definitely correct and well explored, but for further optimization of the computational costs there is a huge complication, because they contain nested sums.

6.2.2 Alternative equations for an optimized algorithm

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:

$$ax + by + c = 0. \tag{6.5}$$

Since each line has an infinite number of possible normal vectors, an additional condition ensuring its unit length is used:

$$a^2 + b^2 = 1. \tag{6.6}$$

The squared distance $l_i^2(a, c)$ between an arbitrary point $P_i(x_i, y_i)$ and the regression line defined by equation (6.5) and condition (6.6) is:

$$l_i^2(a, c) = \left(ax_i + \sqrt{1 - a^2}y_i + c\right)^2. \tag{6.7}$$

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

$$\sum_{i=1}^{N} \frac{\partial l_i^2(a, c)}{\partial a} = 0, \qquad \sum_{i=1}^{N} \frac{\partial l_i^2(a, c)}{\partial c} = 0. \tag{6.8}$$

By substituting (6.7) into (6.8), we get:

$$\sum_{i=1}^{N} \frac{\partial \left(ax_i + \sqrt{1 - a^2}y_i + c\right)^2}{\partial a} = 0, \tag{6.9}$$

$$\sum_{i=1}^{N} \frac{\partial \left(ax_i + \sqrt{1 - a^2}y_i + c\right)^2}{\partial c} = 0. \tag{6.10}$$

After differentiation, the first equation is as follows:

$$2\sum_{i=1}^{N} \left(ax_i^2 - ay_i^2 + \frac{1 - 2a^2}{\sqrt{1 - a^2}}x_i y_i + cx_i - c\frac{a}{\sqrt{1 - a^2}}y_i\right) = 0. \tag{6.11}$$

The equation (6.10) gives a simple result:

$$2\sum_{i=1}^{N} \left(ax_i + \sqrt{1 - a^2}y_i + c\right) = 0, \tag{6.12}$$

which can be rewritten into the form:

$$c = \frac{-1}{N}\left(a\sum_{i=1}^{N} x_i + \sqrt{1 - a^2}\sum_{i=1}^{N} y_i\right). \tag{6.13}$$

Similar rearrangement of (6.11) and substitution of (6.13) yields the equation:

$$\begin{aligned} a\sum_{i=1}^{N} x_i^2 - a\sum_{i=1}^{N} y_i^2 + \frac{1 - 2a^2}{\sqrt{1 - a^2}}\sum_{i=1}^{N} x_i y_i - \frac{a}{N}\left(\sum_{i=1}^{N} x_i\right)^2 - \\ - \frac{\sqrt{1 - a^2}}{N}\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i + \frac{a^2}{N\sqrt{1 - a^2}}\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i + \frac{a}{N}\left(\sum_{i=1}^{N} y_i\right)^2 = 0, \end{aligned} \tag{6.14}$$

which gives the following identity:

$$a\left(\sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} y_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)^2 + \frac{1}{N}\left(\sum_{i=1}^{N} y_i\right)^2\right) = \\ = \frac{1-2a^2}{\sqrt{1-a^2}}\left(\frac{1}{N}\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} x_i y_i\right). \quad (6.15)$$

Simple rearrangement gives the equation:

$$n = \frac{p}{q} = \frac{a\sqrt{1-a^2}}{1-2a^2} = \frac{\sum\limits_{i=1}^{N} x_i \sum\limits_{i=1}^{N} y_i - N \sum\limits_{i=1}^{N} x_i y_i}{N \sum\limits_{i=1}^{N} x_i^2 - N \sum\limits_{i=1}^{N} y_i^2 - \left(\sum\limits_{i=1}^{N} x_i\right)^2 + \left(\sum\limits_{i=1}^{N} y_i\right)^2}, \quad (6.16)$$

where n is the whole fraction, p is the numerator and q the denominator. For a, there is the equation:

$$2a^4(4n^2+1) - a^2(4n^2+1) + n^2 = 0. \quad (6.17)$$

Parameters of the regression line a, b are as follows:

$$a = \pm\sqrt{\frac{1}{2} \pm \frac{1}{2\sqrt{4n^2+1}}}, \quad b = \pm\sqrt{1-a^2}. \quad (6.18)$$

Signs of the square roots in the latter equations can be derived using the signs of numerator p and denominator q of the fraction n. There are two equivalent sets of correct signs (two normal vectors are possible), one of them being:

$$a = \begin{cases} \sqrt{\frac{1}{2} - \frac{1}{2\sqrt{4n^2+1}}}, & q > 0, \\ \sqrt{\frac{1}{2} + \frac{1}{2\sqrt{4n^2+1}}}, & q \leq 0, \end{cases} \quad (6.19)$$

$$b = \begin{cases} \sqrt{1-a^2}, & p > 0, \\ -\sqrt{1-a^2}, & p \leq 0. \end{cases} \quad (6.20)$$

The signum function cannot be used in these equations, because it defines $\text{sign}(0) = 0$. On the other hand, it does not matter which sign $q = 0$ or $p = 0$ are assigned to, because the resulting normal vectors for both cases are only mutually opposite, but still perfectly valid. Coefficient c is obtained from the simple equation:

$$c = -\frac{1}{N}\left(a\sum_{i=1}^{N} x_i + b\sum_{i=1}^{N} y_i\right) \quad (6.21)$$

The last adjustment of the results is optional, but was found to be useful for further processing of the vectorized point cloud. Multiplying all the coefficients

by sign(c) ensures the normal vector points towards the origin of the coordinates - thus, from an obstacle to the free space, when talking about laser scanning - and c coefficient is the shortest distance between the line and the origin. If $c = 0$, the line goes through the origin and the following adjustment cannot be used. In a point cloud vectorization, a problem like this should never happen, because the laser scanner is in the origin of the coordinates, therefore such a situation is physically impossible.

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 previous results, the variance is given by the following formula:

$$\sigma^2 = \frac{1}{N}\left(a^2\sum_{i=1}^{N}x_i^2 + 2ab\sum_{i=1}^{N}x_iy_i + b^2\sum_{i=1}^{N}y_i^2\right) - c^2. \tag{6.22}$$

Equations (6.5)-(6.21) solve the same problem, give the same results as traditional formulas (6.1)-(6.4), can be converted to each other, but do not contain any nested sums. This means that the uncertainty theory from [132] and [133] can still be used and we are allowed to propose new optimizations for the linear regression algorithm.

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* is dedicated to the data preparation, the *line fitting process* generates a set of lines approximating the point cloud and finally the *post-processing stage* reduces the extracted lines to a set of connected line segments and eventually refines bad connections if any have appeared.

There are three inputs to the algorithm. The first one is a continuous cluster (an ordered list of points) to be vectorized. The other two are the maximal permitted variance σ_{th}, representing the required precision, and the maximal intersection distance δ_{th} for output error checking. Both of these parameters will be discussed deeper within an appropriate part of the text. Output is a polyline approximation of all the input points.

6.3.1 Preprocessing stage

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

$$\boldsymbol{a}_i = \left[x_i, y_i, \sum_{k=1}^{i} x_k, \sum_{k=1}^{i} y_k, \sum_{k=1}^{i} x_k^2, \sum_{k=1}^{i} y_k^2, \sum_{k=1}^{i} x_k y_k \right], \tag{6.23}$$

for $i = 1 \dots N$. For further purposes, an additional vector $\boldsymbol{a}_0 = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]$ is defined and an augmented data matrix is formed:

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{a}_1 & \dots & \boldsymbol{a}_N \end{bmatrix}^T. \tag{6.24}$$

A submatrix of $\mathbf{A}$ is defined as:

$$\mathbf{A}[j,k] = \begin{bmatrix} \boldsymbol{a}_j & \dots & \boldsymbol{a}_k \end{bmatrix}^T, \tag{6.25}$$

determined by the row indices j and k, which follow the condition $0 \leq j < k \leq N$.

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.

6.3.2 The line fitting process

An input for this stage of the line fitting process is the augmented matrix $\mathbf{A}$ and a user defined threshold σ_{th}, which determines the maximal permissible standard deviation of points along the line. The process of finding the regression line approximating the highest possible number of points is based on a binary search approach, which is the reason why this algorithm is only capable of working on an ordered list of points. The algorithm is described by the diagram in Fig. 6.5.

There are three control variables in the algorithm. j and k demarcate the interval in the matrix $\mathbf{A}$ 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.

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 $\Sigma x, \Sigma y, \Sigma x^2, \Sigma y^2, \Sigma xy$, belonging to the k point, and appropriate

sums belonging to the j 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. Where the standard deviation σ is within the threshold σ_{th}, three situations may occur. If k points to the last entry in the matrix $\mathbf{A}$, the last line in the cluster was found and the vectorization algorithm is finished. If k is not equal to N and s equals to zero, the binary search is finished, a new line is ready to be added to the output set and all control variables are adjusted to start a new search. The third case occurs during a regular binary search and results in an expansion of the examined interval.

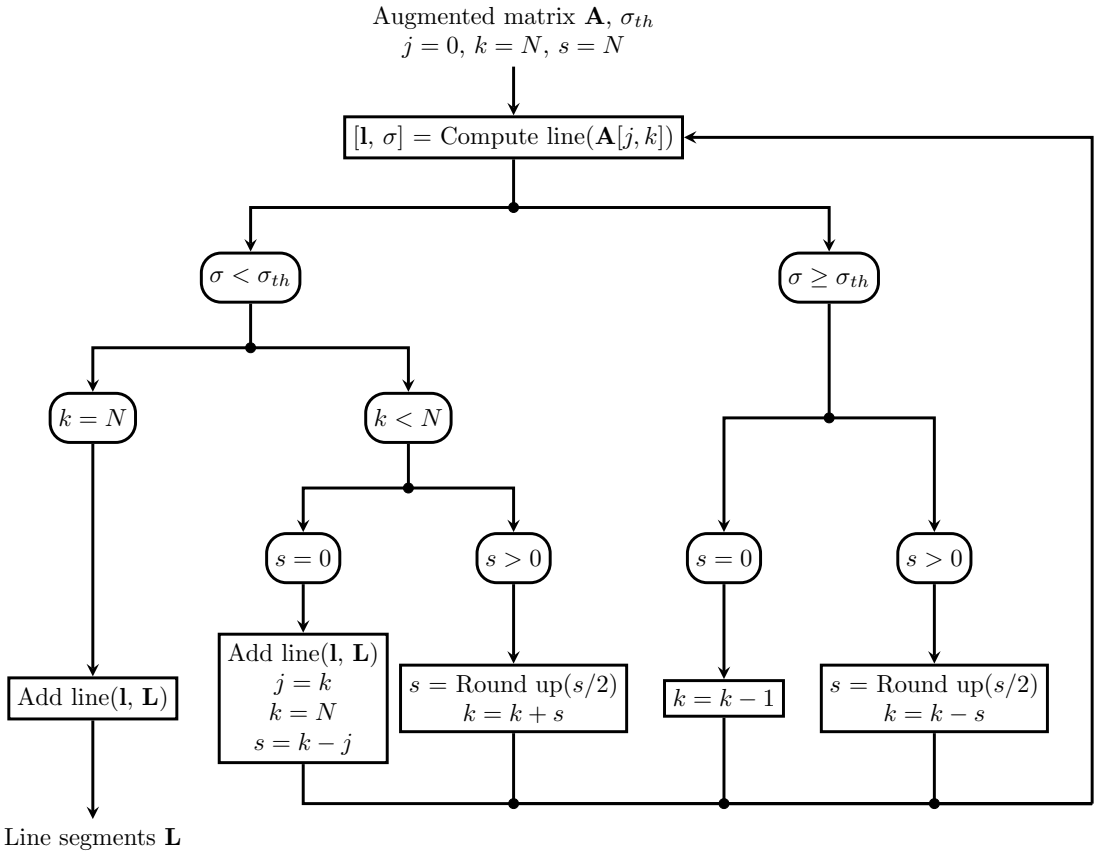


Fig. 6.5: Diagram of the line fitting stage of the FTLS vectorization algorithm. Input: $\mathbf{A}$ - augmented data matrix according to equation (6.24), σ_{th} - maximal permitted standard deviation. Execution control variables: j and k - beginning and end of the currently tested interval in the $\mathbf{A}$ matrix, s - variable for the testing interval adjustment. Temporary variables for the currently tested interval: $\mathbf{l}$ - vector of the best fitting line parameters, σ - standard deviation of the points along the line, Output: $\mathbf{L}$ - set of extracted lines.

When σ is larger than σ_{th} there are only two possibilities. Because the size of

the input array of points can be anything between 2 (minimum to define a line) to infinity (theoretically), a binary search is not always capable of dividing the interval into two equal length parts, which may result in the situation where the search is finished and the variance is still too high. In that case, the algorithm shrinks the tested interval point by point until an acceptable line is found. If the tested interval has the size of power of two, shrinking never happens, otherwise less than $\log_2 N$ steps are necessary. The second possibility represents a standard shrinking of the interval during the search.

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 linear algorithms.

6.3.3 Postprocessing stage

Lines extracted in the previous step might be truncated using their end points and returned as an output and the algorithm finishes. 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.

(a) Failure of the intersection method. (b) Correction using an additional line.

Fig. 6.6: The postprocessing of the extracted lines using an intersection method with corrections.

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, as shown in Fig. 6.6a. Such a failure is impossible when using point eliminating methods such as the DP algorithm, but with linear regression it is a rare but serious circumstance.

To detect the problem, the algorithm checks the distance between the intersection of lines and the point in the cluster where one approximation changes to another. If the distance is greater than a predefined threshold δ, a correction is made. Points related to both consecutive linear approximations are divided into three intervals instead of the original two, and a new regression is calculated. The inflection point is usually bridged by the intermediate line and the curvature is better approximated. The correction is depicted in Fig. 6.6b.

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. The number of corrections depends on the input data and cannot be easily predicted. In the worst case scenario the range of N points can always be approximated by $N - 1$ lines with zero error, which means no simplification have occurred. This is not a flaw of the FTLS algorithm, but a sign of a too low value of the maximal intersection distance parameter supplied to the algorithm. In such a situation, the algorithm loses its generalization ability and increasing the parameter should be considered. The optimal value depends on the noise present in the input data.

After this stage is complete, the vectorization process is finished and the extracted lines are ready for further utilization.

6.3.4 Implementation considerations

Although, as described above, the FTLS algorithm works precisely from the mathematical point of view, there are several practical issues that deserve special attention.

In this context, the first problem relates to the selection of an appropriate data type for the representation of a single point coordinates. The chosen format should be able to represent the given coordinates without any loss of precision, but, due to a large number of processed points, it should use the smallest possible quantity of bits to save memory and bandwidth. 16-bit integers or 32-bit floating point numbers suffice to store coordinates with five (or six) significant digits, which is satisfactory in most practical situations.

The choice of the data type to represent the sums, as defined in equation (6.23), is directly dependent on the numerical type used for a single coordinate and on the number of points to be processed. The summation has to be finished without any loss of precision to prevent numerical instability of the equation (6.16). The

subtraction in the denominator is prone to catastrophic cancellation, which may lead to a critical computation error; such behavior is prevented by sufficient precision of the data types. Common choices are 32-bit and 64-bit integers or, if needed, 64-bit floating point numbers.

Further computation should employ the same data type as the previous phase. Of course, the results from equations (6.19), (6.20) and (6.21) can be truncated to any desired precision.

6.4 Augmentation for the globally optimal approximations

The FTLS algorithm outputs an ordered set of extracted lines, which approximate the whole cluster being processed, but the approximation suffers the threshold related suboptimality as described in Section 6.1.4. 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 reference by the AFTLS abbreviation.

Every extracted line produced by the second stage of the FTLS algorithm has an exactly defined interval of points, which belong to it. This information is used to build a break point vector, where the index of the last point of every line is stored:

$$\boldsymbol{b} = [b_0, b_1, \ldots, b_L]\,. \tag{6.26}$$

The $b_0 = 0$ is a dummy element representing a virtual point before the beginning of the point cloud. The last break point always refers to the end of the point cloud, therefore $b_L = N$. All the other break points satisfy the condition: $b_{i-1} < b_i$ for $i = 1 \ldots L$.

Every extracted line has also attached an error value given by equation 6.22, which describes, how well does it approximate the given data. Since the coefficients a, b and c can be expressed as a function of $\mathbf{A}[p, q]$, the variance can be also denoted: $\sigma^2\left(\mathbf{A}[p, q]\right)$.

This adjustment of the notation allows an expression of the error E of the approximation of the whole cluster. It is defined as follows:

$$E(\boldsymbol{b}) = \sum_{i=1}^{L} \sigma^2\left(\mathbf{A}[b_{i-1}, b_i]\right) \frac{b_i - b_{i-1}}{N}. \tag{6.27}$$

The original FTLS algorithm does not evaluate any global error metrics, its third stage consists only in turning the extracted lines into the polyline, which describes

the general shape of the cluster. To get the globally optimal results analytically, the following set of the equations would have to be solved:

$$\frac{\partial E(\boldsymbol{b})}{\partial b_1} = 0 \quad \dots \quad \frac{\partial E(\boldsymbol{b})}{\partial b_{L-1}} = 0. \tag{6.28}$$

There is only $L-1$ of equations for L extracted lines, because b_0 and b_L coefficients are defined to be constant, when building the break point vector $\boldsymbol{b}$ according to the equation (6.26). Unfortunately, the analytic solution of the system (6.28) is extremely complicated or even impossible, because the error function $\sigma^2(\mathbf{A}[p,q])$ 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) [140] was employed. It 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. The NM method is designed for localization of extremes of the non-linear functions with arbitrary number of arguments. The method is based on a simplex composed of $D+1$ vertices in a D-dimensional space of possible arguments of the optimized function. Each vertex has its function value attached and every iteration the least suitable one is replaced by a newly computed successor. The function value of the new vertex is usually better than the previous one, so as the iterations proceed, the simplex shrinks and closes to the set of arguments giving the optimal value.

In the original NM method, the simplex is described by a set of $D+1$ vectors with D elements. In case of AFTLS, the search for the optimal break point positions, the situation is slightly more complex, because every argument vector for the optimized error function (6.27) has constants at the beginning and at the end. At the beginning of the optimization process, there is only one initial guess on position of the break points produced by the second stage of the FTLS algorithm, therefore $L-1$ new vertices have to be generated. The resulting simplex has the structure:

$$\mathbf{S} = \begin{bmatrix} \boldsymbol{s}_0 & \boldsymbol{s}_1 & \dots & \boldsymbol{s}_{L-1} \end{bmatrix}^T, \tag{6.29}$$

generated by the following rules:

$$\begin{aligned} &\boldsymbol{s}_0 = \boldsymbol{b}, \\ &s_{i,j} = b_j \qquad i \neq j, \\ &s_{i,j} = b_j - \tau \quad i = j, \end{aligned} \tag{6.30}$$

for $i = 1 \dots (L-1)$ and $j = 0 \dots L$. As the first vertex, the initial guess is adopted without any changes. The remaining $L-1$ vertices are generated by subtracting a

user defined value τ from a single element of the vector $\boldsymbol{b}$ (different element for every new vertex). The τ parameter is an integral variable and should fulfil the condition $\tau < b_0$ to ensure, that all the elements of the vertex will satisfy the constraints discussed already beside the equation (6.26). τ should correspond to an expected number of extra points assigned to the line segment, but the NM method is robust enough to handle even very rough estimates.

The vertex manipulating operations, which compose the core of every iteration of the NM method, were implemented in our algorithm according to [141]. The structure of the iteration is well described, therefore the details are not reproduced here. The error function (6.27) and the simplex (6.29) are enough to start the optimization process.

There are two major differences to the original approach [140]. The first is the introduction of constraints to otherwise unconstrained method. In general, adding constraints to the NM method can be very complicated [142], but in our case, adding limits cannot lead to worse results, than the initial guess, neither can cause a collapse of the method. Because the break points must keep the order determined by the points in the source point cloud, the algorithm does not search through the whole $(L-1)$-dimensional space, but only through its part, limited by the inequality $s_{i,j-1} < s_{i,j}$ for $i = 0 \dots (L-1)$ and $j = 1 \dots L$. This condition is checked every time after a new vertex is generated and proceeds from the end to the beginning of the particular vector. Any break points out of the range are trimmed to fit into the boundaries. A new error function value is computed after this check.

The second difference to the original approach is the discreteness of the arguments of the optimized function. While the original method is designed to be used with real numbers, the break points can be only integers. In combination with the constraints discussed above, this means, that there is a finite number of possible vertices. At the end of the iteration process, as the solution becomes more precise, the vertices inevitably start to overlap and the simplex collapses. If that happens, or if the vertices end up in one hyperplane, the algorithm is stopped and the best solution is taken as the output of the optimization.

The concept described above ensures, that no worse result than the initial guess can appear and in the vast majority of cases a significant improvement can be observed. The postprocessing, after the modified NM optimization is over, is the same as in the original FTLS algorithm.

6.5 Testing and experiments

Both FTLS and AFTLS algorithms were deeply tested in [92] and [93]. This theses summarizes the most important experiments and any further details are possible to

be found in the said papers. Major role in this section have the synthetic tests from simulations, because of higher flexibility and better control over parameters of the environment and "measured" point cloud. Real world experiments were performed as well and conclude the practical evaluation.

Three established algorithms were chosen for comparison with the proposed methods:

The INC algorithm constitutes the fastest known algorithm for TLS vectorization. Precomputed sums to make it $O(n)$ complex were employed to make the INC algorithm comparable to the other algorithms tested. However, there is no decimation of the point cloud as mentioned in [96] (where the authors iterate through the cloud by five points at each step), because all the other algorithms operate at a full resolution.

The DP and RW algorithms were selected to represent the group of point eliminating methods. These are examined because it is useful to observe how the FTLS algorithm compares to the generally fastest, yet less precise, vectorization algorithms. The implementation of the DP is non-recursive as this variant was found to be slightly faster than the usual form.

The FTLS and AFTLS algorithms also include buffer arrays. The allocation of required memory is performed before the benchmark starts running because the memory has to be allocated only once but can be used multiple times later. The advance allocation of resources is usual practice in the real time processing of large data, so it should not violate standards of proper benchmarking.

All of the algorithms are given the same data as the input. The points are represented by two 32-bit floating point numbers as the 32-bit float type is able to represent six significant decimal digits precisely, which reflects the precision of currently produced laser scanners. The DP and RW algorithms rely on point to line distance computation, which can be safely performed using the 32-bit floating point variables. The TLS algorithms are prone to numerical instability, as described in Section 6.3.4. To prevent this problem, all sums are represented by a 64-bit floating point number, which guarantees a precision of no less than 15 significant decimal digits.

The approximation quality metrics of the selected algorithms differ substantially: while in the DP and RW algorithms we consider the maximal permissible distance, in the TLS algorithms the relevant quantity consists in the variance of distances. To unify this measure for both approaches, a maximal standard deviation σ is defined for each experiment. For the DP and RW the 3σ threshold is used, and for the INC and FTLS algorithms the variance is naturally σ^2.

All the tests were implemented in the C++ programming language, using the Microsoft Visual Studio 2012 (v110) compiler with -O2 optimization. The timing

data were obtained using the standard library clock function. The machine running benchmarks was equipped with a six-core, 64-bit AMD FX-6350 CPU, whose base frequency is 3.9 GHz. The CPU comprises a 3x2 MB L2 and an 8 MB L3 caches, and therefore all the data processed by the algorithms certainly fit the cache memory. Although the CPU supports a large variety of SIMD instruction sets, we opted not to use them, because this would reduce the generality of the tests performed.

6.5.1 Synthetic tests

A wide spectrum of diverse synthetic tests were performed to demonstrate different features of the algorithms under different conditions. The most important are discussed in this section.

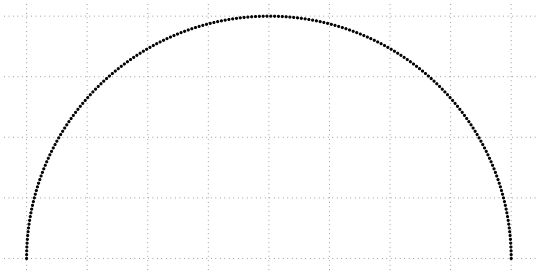
Speed versus different number of extracted line segments

The first set of tests investigates the relationship between the number of extracted line segments and the speed of the algorithms. A simple semicircular cluster with a specified number of points N was generated for this purpose. An example with $N = 100$ is shown in Fig. 6.7a. The number of points to be tested was chosen to reflect the standard, currently achieved point cloud densities, with some overlap towards larger clouds because laser scanners will probably provide even more data in the future. Since the cluster is continuous and no noise is present, no preprocessing in terms of segmentation or filtration was performed. Smooth shape of the cluster also means, that the threshold problem with sharp corners do not manifest as intensively and further refinement using the AFTLS algorithm would barely bring any noticeable improvement at significant performance penalty. Due to this reason, the AFTLS algorithm in omitted in this experiment.

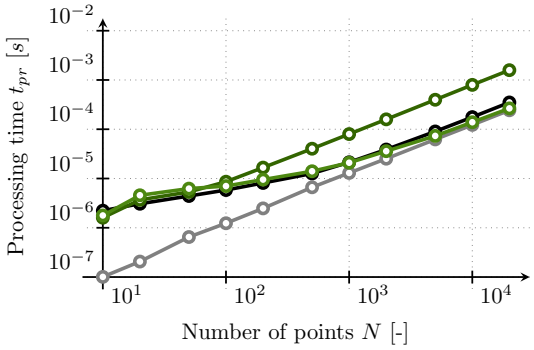
The graphs in Fig. 6.7 show the results of the benchmark for three precision threshold. As the maximal permitted standard deviation σ grows, more lines are required to sufficiently approximate the shape. This noticeably influences performance of some of the tested algorithms.

At the beginning of the characteristics listed in Fig. 6.7, the situation is rather simple. The algorithms lack enough points to satisfy the required precision and the only sticking out is the RW method with its incredibly low computational time per iteration. The others perform nearly the same.

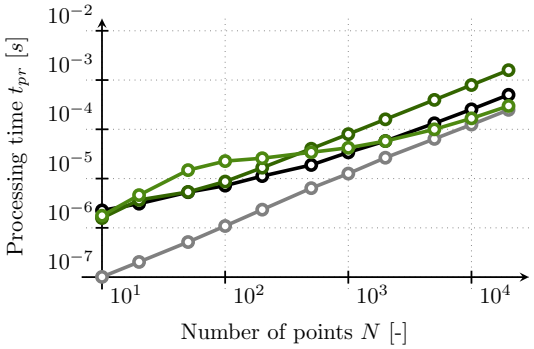
For a slightly larger number of points, the situation changes. The INC and RW exhibit a linear growth of computational time as described in Section 6.1. The DP method proves itself to be more efficient than the INC algorithm and becomes faster at N around one hundred or even lower. The FTLS algorithm performs the worst in this area and is even slower than the traditional INC algorithm.



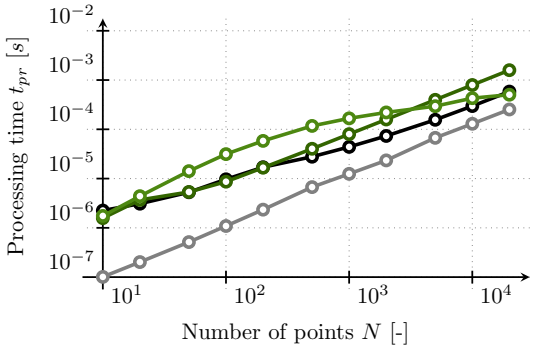
(a) An example of a semicircular point cloud with 100 points.



(b) $\sigma = 0,03$



(c) $\sigma = 0,003$



(d) $\sigma = 0,0003$

Fig. 6.7: The vectorization speed of the tested algorithms with respect to different number of extracted line segments in a semicircular cluster. The growing required precision σ imposes growing number of line segments necessary for approximation, which directly affects performance of the non-incremental algorithms. (FTLS - green, INC - dark green, DP - black, RW - gray)

Third part of the characteristics describing behaviour for larger clusters is where the FTLS proves its qualities. Once there is enough points for elimination, the graph bends down and the performance of the FTLS becomes better than the one of the traditional INC approach. The DP characteristics is already settled at a linear growth similar to RW and INC and for large sets of points FTLS outperforms it as well, closing to the RW characteristics. This is the point where the $O(m \log N)$ complexity starts to be more favourable than $O(N)$. A higher number of extracted lines moves this point to a higher cluster sizes (compare Figures 6.7b, 6.7c and 6.7d), but it always exists. Of course, the preprocessing stage of the FTLS algorithm is $O(N)$ complex, but the absolute amount of computation is very low, which bring it in line with the fastest RW algorithm.

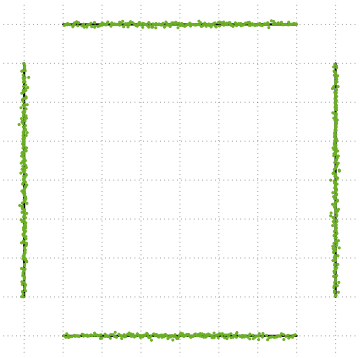
An extraneous result of this set of experiments is also the fact, that for approx-

imation of a smooth continuous point cloud the TLS methods generally need less lines, than the PE ones.

Testing in a virtually simulated environment

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 was obtained using the approach described in Section 7.4.2.

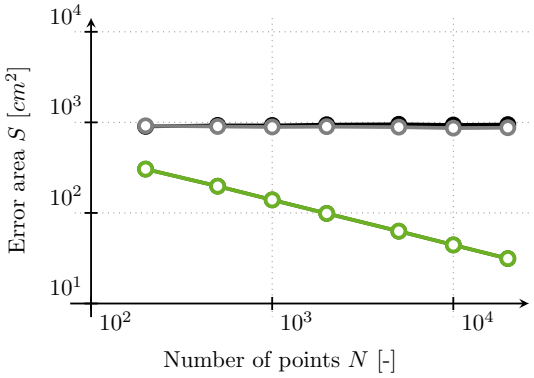
The experiment in Figure 6.8 is focused on a single line approximation. The number of the lines extracted by the algorithms well corresponds to expectations.



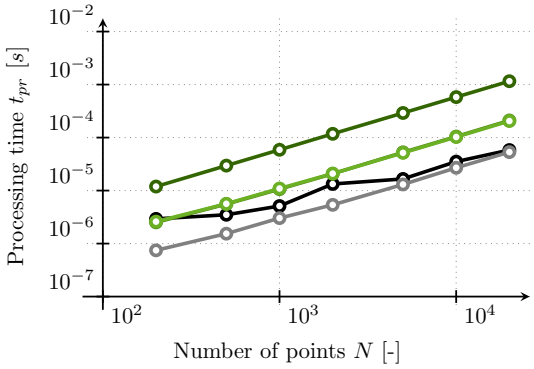
(a) The point cloud. (Grid 50 x 50 cm)

No. of points	No. of lines	Extracted lines [%] TLS	DP	RW
200	4	0	+2,9	+85,3
500	4	0	0	+109,9
1000	4	0	0	+127.5
2000	4	0	0	+139,7
5000	4	0	0	+143,9
10000	4	0	0	+150,5
20000	4	0	0	+155.9

(b) The relative line extraction success-rate.



(c) The precision benchmark. All TLS methods share the same characteristics.

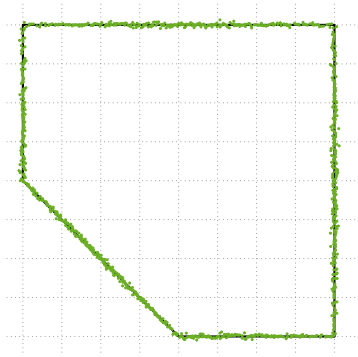


(d) The speed benchmark.

Fig. 6.8: The point cloud shape in this experiment contains four solitary edges. Each of them can be approximated by a single line segment. (AFTLS - light green, FTLS - green, INC - dark green, DP - black, RW - gray)

TLS methods provide exactly the correct number of lines and the DP algorithm with a small exclusion at the sparsest cloud as well. RW, due to its nature, produces more and more lines as the density of the points grows. The precision benchmark reveals gradual improvement in case of TLS methods, which all provide the same results. The point eliminating methods do not exhibit this property. The speed comparison for TLS methods and the RW algorithm show linear time complexity, which should be the case for a single line approximation of the cluster. Contrary to the theory, the DP algorithm behaves in a more complicated manner. Repeated long-term experiments confirmed this observation. The reason probably lies in reallocation of the internal buffer, which manifests significantly for fast computation times and turns negligible for large clusters.

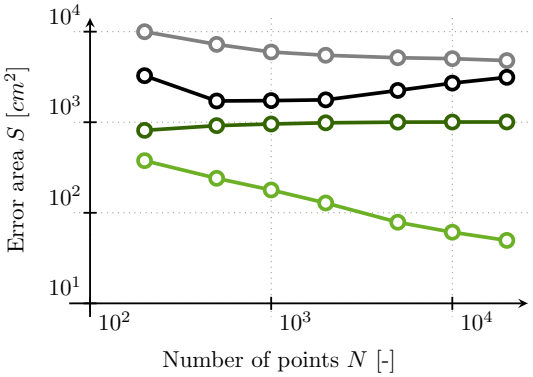
In contrast to the previous one, the experiment in Figure 6.9 is focused on vectorization of a large continuous cluster with several sharp corners. In terms of



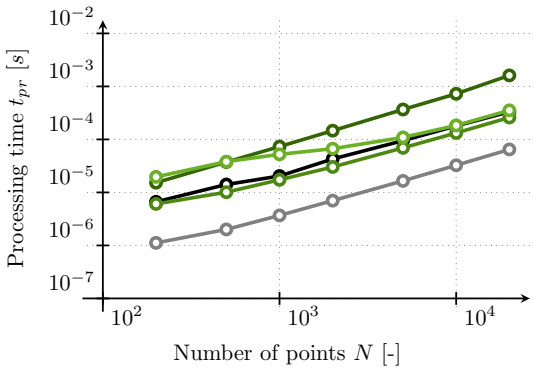
(a) The point cloud. (Grid 50 x 50 cm)

No. of points	No. of lines	Extracted lines [%] TLS	DP	RW
200	5	+24,3	+9,1	+118,5
500	5	+0,4	+7,2	+136,8
1000	5	0	+7,5	+149,1
2000	5	0	+13,4	+156,7
5000	5	0	+15,6	+160,8
10000	5	0	+13,9	+169,0
20000	5	0	+12,2	+176,2

(b) The relative line extraction success-rate.



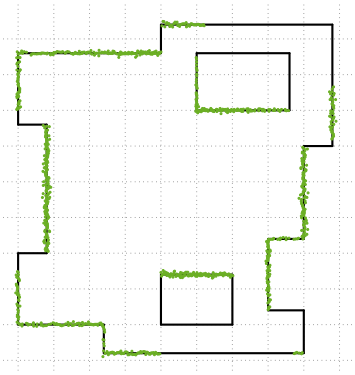
(c) The precision benchmark. FTLS and INC share the same characteristics.



(d) The speed benchmark.

Fig. 6.9: This experiment explores behaviour of the algorithms in case of a large cluster, which needs to be approximated by five consecutive line segments. (AFTLS - light green, FTLS - green, INC - dark green, DP - black, RW - gray)

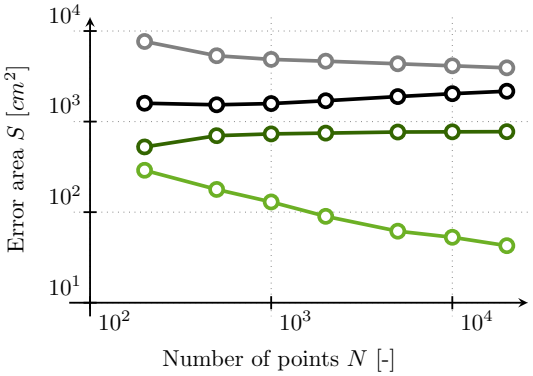
correct number of extracted lines, the TLS methods provide good results again. The DP algorithm sometimes fails and provides more lines than necessary regardless the number of points available. The RW algorithm overestimates the count of the approximation lines in a similar way to the case in Figure 6.8. The precision benchmark clearly illustrates the threshold problem in vectorization of ordered point clouds. The PE methods generally perform poorer than TLS, but FTLS and INC algorithms get stuck on certain precision and do not provide better results with growing number of points, even though the information needed is present. Optimization in the AFTLS addresses this problem and the light green characteristics proves it to work correctly, because the error area decreases steadily, as the number points available grow. The results of the speed benchmark reveal increased computational of the AFTLS optimization, but as it is proportional to the number of extracted



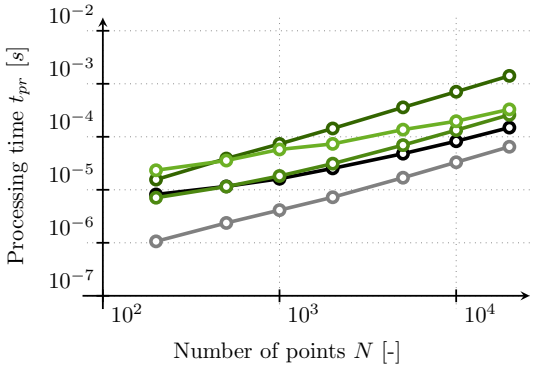
(a) The point cloud. (Grid 50 x 50 cm)

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. 6.10: 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)

lines, the added cost becomes irrelevant at higher cluster densities and the speed is close to the original FTLS algorithm.

The third synthetic test in Figure 6.10 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 exhibits the same tendency as in previous cases, only the smaller cluster sizes impose lower number of additional lines. The precision benchmark again illustrates TLS superiority over the PE methods and AFTLS refinement abilities for large clusters. 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 than in the test in Fig. 6.9. Nevertheless, its performance for larger point clouds is still much better than of the original INC algorithm.

6.5.2 Real data testing

The experiments presented so far were carried out using synthetic data only, and the related practical tests are described in this section. All the data come from the segmentation and filtration experiments in Chapter 5.

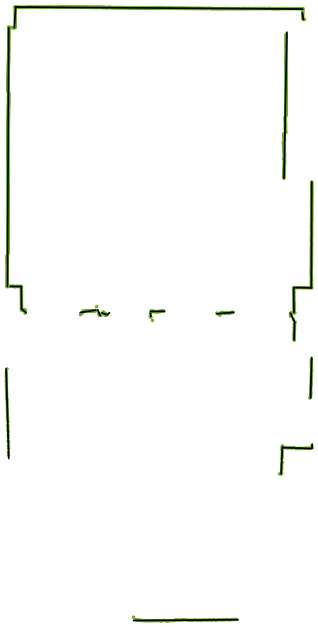
The speed measurements were made on a machine using a four-core / eight threads, 64-bit Intel Core-i7-4790K CPU, running on 4.0 GHz. Processor cache memory is large enough to hold all of the data of every test performed. The software and implementation details remain the same as in the previous section. Precision evaluation of the real laser scans was found to be very complicated. The big issue is the precision of the reference measurement for the benchmarking. Every millimetre in the reference measurement taken in an empty room, had a serious impact on the error enumeration, which could easily cause the misleading results, especially in case of the AFTLS algorithm, where the precision evaluation would be the most important. For this reason, precision evaluation of the real scans was omitted and only the number of the extracted line segments was traced. As will show the third experiment, even this evaluation is not completely reliable.

The point clouds obtained in a man-made environment are generally quite similar, usually composed of a set of polylines, thus for an illustration of the practical performance of the proposed algorithm and for comparison with the other methods, three representative scans in Figures 6.11, 6.12 and 6.13 were selected. The point cloud statistics and the results are summarized in the table next to each vectorized scan. In all cases, the σ parameter of the TLS methods equals to 2 cm, therefore the PE threshold is 6 cm.

The experiments in Figures 6.11 and 6.12 represent an easy environment with

long, clear walls and low amount of small objects. The observation of the number of the extracted lines is interesting due to the different outcomes of the TLS algorithms. In synthetic tests, this had rarely happened, but real scans contain several small details which can cause this behaviour. INC gives the largest count, because it stops every time the given threshold is exceeded during the incremental search. FTLS can skip small fluctuations, so the extraction phase gives lesser or equal number of lines as the INC algorithm. After that, the augmentation of the AFTLS method can take place and refine the results, which directly influences the postprocessing stage, where, in case of bad intersections, another lines can be added. This whole variety of possibilities mostly ends on the same results, but the discussed examples nicely illustrate, how much factors can affect the vectorization process. The point eliminating methods provide more lines than necessary, especially the RW algorithm is very ineffective. On the other hand, the processing times correspond to the expectations very well with FTLS being the fastest TLS algorithm and the PE methods performing approximately twice as better.

The third real world measurement in Figure 6.13 is somewhat different. It originates from a laboratory, where lot of small things were in the field of view of the laser scanner. The result is a highly cluttered point cloud with only a few clear, straight clusters. The correct number of extracted lines is therefore hard to predict



(a) The vectorized scan. (FTLS)

Point cloud statistics	
Points	1944
Lines	25
Extracted lines	
FTLS	27 (+13%)
AFTLS	27 (+13%)
INC	31 (+29%)
DP	35 (+46%)
RW	58 (+142%)
Processing time [μs]	
FTLS	15,6
AFTLS	17,6 (+12%)
INC	83,5 (+432%)
DP	8,6 (-45%)
RW	8,4 (-46%)

(b) The vectorization results.

Fig. 6.11: A real point cloud obtained in an empty room, containing mostly long line segments.

(a) The vectorized scan. (FTLS)

Point cloud statistics	
Points	1918
Lines	10
Extracted lines	
FTLS	15 (+50%)
AFTLS	11 (+10%)
INC	13 (+30%)
DP	14 (+40%)
RW	30 (+200%)
Processing time [μs]	
FTLS	14,6
AFTLS	15,1 (+3%)
INC	77,0 (+427%)
DP	7,4 (-49%)
RW	7,2 (-51%)

(b) The vectorization results.

Fig. 6.12: A scan of a narrow hallway. Sparse points at the distant measurements as well as dense clusters near the scanner are properly vectorized.

(a) The vectorized scan. (FTLS)

Point cloud statistics	
Points	1981
Lines	50
Extracted lines	
FTLS	73 (+46%)
AFTLS	76 (+52%)
INC	76 (+52%)
DP	78 (+56%)
RW	113 (+126%)
Processing time [μs]	
FTLS	21,5
AFTLS	29,1 (+35%)
INC	79,4 (+269%)
DP	31,0 (+44%)
RW	28,9 (+35%)

(b) The vectorization results.

Fig. 6.13: A real point cloud obtained in a laboratory environment with a few clear walls and a lot of small objects.

and the statistics is provided as an overview rather than an exact evaluation. In this case, the processing time is the interesting part of the table. The TLS methods behave as expected, but both PE algorithms perform much worse than in the previous cases, even slower than the FTLS algorithm. Though it may look optimistic at first, a great caution is in place. Similar inconsistency was already observed during the synthetic test in Figure 6.8 in case of the DP algorithm. As far as the author knows, there is no easily identifiable reason in the source code. More suspicious seem to be the CPU with its cache management, dynamic frequency changes and so on. Highly complex hardware with run-time resource management makea the serious benchmarking a challenging task and a comparison of algorithms can easily turn into a comparison of implementations [143]. This experiment shows, that the performance of the algorithms is predictable only to some extent and if the speed is really mandatory, a benchmark with particular implementations on given hardware should be made.

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 to deal with the threshold problem in vectorization was shown as well. The algorithms were tested in many experiments with the following results:

The most important outcomes of the synthetic tests in Figures 6.8, 6.9 and 6.10 are three facts: Firstly, the point eliminating methods are proven to yield less accurate results than the TLS methods. The error area in accordance with the methodology from Section 7.4.2 is several times larger for the DP algorithm and even more extensive for the RW algorithm. Secondly, the INC algorithm is always slower than the FTLS algorithm and the 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 6.1.4 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 as presented in

Section 6.5.2. 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. In the natural environment, line extraction may become more problematic, and thus different approaches are often employed. An in-depth treatment of this topic is provided in [144], and an example of a more recent method can be found in [145].

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. Even a special case, such as point cloud registration, has many applications in practical life, namely object reconstruction, non-contact inspection, medical and surgery support and autonomous vehicle navigation. Of course, the later one, with an emphasis on range measurements, will be in question in the following pages, but many techniques from one domain are perfectly applicable in the others as well, so inspiration from other fields is inevitable [146]. Computer vision methods based on image processing are not covered in the following overview due to different nature of the input data, though a great amount of them is interesting for robotics as well [147], [148].

The registration process requires some measure of quality of the fitting, which is usually called similarity or simply (generalized) distance. The notion of *similarity*[1] and *congruence*[2] in mathematics is quite strict and does not allow any deviations in the shape of the compared objects. On the other hand, every measuring device has some error and the world is not static, so exactly same observations fulfilling the mathematical definition are acquired in rare occasions, if ever at all. Instead, similar values appear in similar situations, and this is where some metrics of similarity becomes essential. A human observer usually spots the similarity unconsciously due to our evolutionarily developed sense for pattern matching, but a machine, which did not have millions of years for its development, is reliant on numerically evaluable algorithms. Some results in this chapter related to similarity were already published in [149] and are adapted for the needs of this thesis.

The actual survey of the point cloud registration techniques in the following section deals only with the methods able to fit the incomplete measurements. 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.

[1] In sense of geometry, two shapes are similar, if one can be obtained from the other using only a uniform scaling, translation, rotation or reflection.

[2] Congruence is slightly stricter than similarity, because it forbids scaling. Only a rigid transformation (translation, rotation, reflection) is allowed.

7.1 Feature matching and shape registration techniques

As discussed in Chapter 2, the robotic mapping was mainly focused on point-like features in the past and fitting of the whole point clouds became popular, when better range finders and hardware with higher computation power got available. Before this change, the registration of point clouds and various other geometrical entities was mainly cultivated in the domain of computer aided design and non-contact inspection. A recent survey [150] on robotics reveals a single method dominating the field called *iterative closest point* (ICP) with its variants. Different approaches exist as well, but are not nearly as popular. The methods working directly with more complex geometrical primitives are even more uncommon, possibly because the ICP is applicable on them as well. The last subsection of the theoretical overview is dedicated to the line segment similarity, which is going to be extensively addressed in the technical sections of this chapter.

7.1.1 The iterative closest point algorithm

The original ICP was first presented in [151]. The method is well elaborated, applicable in both 2D and 3D spaces and according to the introductory statement in [151], 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.

Since its introduction, ICP have been succesfully used in many practical and scientific applications. [150] provides a graph showing an increasing number of papers dealing with ICP from its publication until 2013[3]. Today, the original ICP has dozens of variants [146], which are regularly compared [152] and tested in bechmarks [153].

Generalized ICP [154] is an important addition to the original registration toolkit. The authors introduced a probabilistic model containing locally planar structure of the registered point clouds, which enhances both convergence and accuracy. As always, other upgraded variants are available, for example [155] with approximate surface reconstruction and [156] with 2D image features. A frequently addressed

[3]The authors have identified approximately 1800 papers with *Iterative Closes Point* in a title or abstract in that period of time. Although it is not clear, whether the ICP was really used, challenged with a new algorithm or merely cited as an obligatory item of the state of the art, it is undoubtedly very popular.

problem of the ICP with convergence to local minima resulting in wrong correspondences, is usually solved by a coarse pre-registration using different method (geometric features [157] or with genetic algorithms [158]). A process of matching of multiple scans in one optimization routine is described in [159] and although it is applicable mainly in the off-line data processing, the results are much better, than in the case of sequential registration.

Generally, the ICP is succesfull and widely applied technique [146], [150][152], [153] and [160]. On the other hand, the large variety of modifications suggests, that there are still issues to be addressed and the operation is not always perfect. Despite its dominance in the field, further research in this direction seems to be appropriate.

7.1.2 Alternative approaches

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 does not impose real competition to the ICP. A straightforward example is a simple correlation. Testing of the overlap under various transformations of one of the registered shapes is relatively easy to implement, however computational complexity of $O(n^3)$ in 2D and $O(n^6)$ in 3D makes it very slow for larger data sets.

Random sample consensus (RANSAC) method, which was already mentioned in Section 6.1 in context of vectorization, is also sometimes used for registration of the whole point clouds. Because it draws random possible correspondences, the serch space is not as high dimensional as in the case of bare correlation, but its stochastic nature makes it somewhat unpredictable, when it comes to convergence. Nevertheless, RANSAC can be found as a building block of the registration pipeline in recent literature [161], [162] and [163].

Principal component analysis (PCA) is another possible mean of shape registration. First, a correlation matrix for each point cloud is computed and eigenvectors are identified. In the second step, the orthogonal bases of eigenvectors are aligned, which directly provides a transformation for the whole point cloud. The method is simple and fast, but fails in case of more complex shapes of point clouds and does not deal with incomplete data very well. In case of suitable data, it has clearly its place in the state of the art [153].

A good source of alternatives to ICP are methods adopted from the field of computer vision. Representative examples are *Scale-invariant feature transform* (SIFT) [164] and *Speeded up robust features* (SURF) [165]. Both are based on feature extraction and registration using these special points, which corresponds more to the traditional SLAM solutions rather than modern effort to utilize full geoometry of the scene and possibly the semantics as well. Despite this drawback, the poisitive

results of these methods are undoubtful and can be found in many papers, i.e. [166], with inverse cumulative histogram in [167], or with ICP in [168].

Another possiblity is to model the point cloud as a *mixture of Gaussians* and define a function for a distance between two such mixtures as proposed in [169]. This approach well reflects the probabilistic nature of measurements and provides good robustness and accuracy. Similar method is introduced in [170] for pose estimation in mobile robotics. Again, the results are promising, but unfortunately, the methodology is applicable only to raw point clouds.

7.1.3 Registration using complex primitives

Methods directly operating on more complex primitives, such as line segments, non-uniform rational B-splines, triangle meshes etc. are very rare. This was not always the case and reviewing an old literature on shape registration reveals some attempts in this direction. In [150] is an interesting reference to [171], where critique of usage of advanced geometric primitives is present. Primitives derived from point clouds are claimed to be more sensitive to noise and not able to provide stable, features invariant to rigid transformation. Points are deemed to be more robust and are proposed as a better base objects for shape registration. This opinion in combination with applicability of ICP on pretty much any geometrical primitive probably led researchers to concentrate on a raw point cloud matching and complex primitives became nearly abandoned.

[172], [173] and [174] are sparse remarks on the theme of line segment based shape matching. [172] directly stems from the original ICP algorithm, while [173] and [174] present their own approaches. [173] is especially interesting from robotics point of view, because among other applications it deals with mapping using laser scanners. Unfortunately, a comparison with other techniques is not very elaborated in all three cases, but the presented results seem, at least visually, comparable to the techniques mentioned above.

Second area in the shape registration field, where advanced primitive sometimes appear, are the modifications to the original ICP. For example the generalized ICP [154] mentioned above uses local planar approximations and [175] uses the quadratic curves in 2D or patches in 3D. Put this way, a renaissance of the registration algorithms based on more complex geometrical shapes seems to be slowly drawing in.

From the point of view of robotic mapping, the is clearly a need to leave points and model the environment in a more thorough way. Critique [171] gives a strong reason to be cautious about inaccurate approximations, but on the other hand, possible benefits seem to outweigh the additional effort. Total least squares vecto-

rization with a suppressed threshold related error described in Chapter 6 provides high quality approximation, which could potentially lead to better registration results. Reinvestigation of this area is therefore in place and will be covered in the rest of this chapter, especially in Section 7.3.

7.1.4 Similarity evaluation of the corresponding line segment pairs

The material presented in this subsection was already published in [149] and the following text heavily draws from this source.

Some measure of similarity or distance betweed the shapes is necessary for each registration algorithm. Though the notion of similarity may look easy at first, it is quite complex task to solve in general. Different situations require invariance to different transformations. For example [176] requires invariance to all affine transformations, [177] requires invariance only to a subset of possible transformations and in [178], only the invariance to rigid transformations and scaling is demanded. On the other hand, all transformations are important in motion estimation applications [179], [180], path planing [181] and the SLAM problem in robotics [182]. Many applications exist for 3D object detection [183] and 3D scene matching [184], where incomplete line segments often appear. Dealing with incomplete information is essential, as already mentioned at the beginning of this chapter. Another set of algorithms is used for polygons [185] or polyline curves [186]. Mathematically well described is the *Frechet distance* [187], but its domain of operation are polylines and in case of isolated line segments it simplifies to bare comparison of distances, similar to the algorithms above.

The most relevant criteria to the demands of SLAM are based on the Hausdorff distance. Valuable comparison of the established line segment distance functions [188] evaluates three of these similarity metrics.

The Hausdorff distance function for line segments in its basic form, as described in [185], is defined as:

$$\begin{aligned} d_{l_1,l_2}(l_1,l_2) &= \sup_{\boldsymbol{P}\in l_1} \inf_{\boldsymbol{Q}\in l_2} d(\boldsymbol{P},\boldsymbol{Q}), \\ d_{l_2,l_1}(l_1,l_2) &= \sup_{\boldsymbol{P}\in l_2} \inf_{\boldsymbol{Q}\in l_1} d(\boldsymbol{P},\boldsymbol{Q}), \end{aligned} \tag{7.1}$$

$$d_{crit}(l_1,l_2) = \max(d_{l_1,l_2}, d_{l_2,l_1}), \tag{7.2}$$

where $d(\mathbf{P},\mathbf{Q})$ is some distance metric (Euclidean in most cases), d_{l_1,l_2} is the longest perpendicular distance from l_2 to l_1 and vice versa in the second case. The criterion value is the higher of these two distances.

The *modified Hausdorff line segment distance* originates from [189] and the definition is as follows:

$$d_{crit}(l_1, l_2) = \min(\| l_1 \|, \| l_2 \|)\sin(\alpha), \tag{7.3}$$

where $\| l_x \|$ denotes length of the particular line segment and α is the angle formed by l_1 and l_2.

Modified perpendicular line segment Hausdorff distance [186] relies on the perpendicular distance between an end point of one line segment and the corresponding line segment as a whole. If a line segment is described as $l_x = \{\boldsymbol{P}_{x1}, \boldsymbol{P}_{x2}\}$ then the perpendicular distance can be written as $d_\perp(l_x, \boldsymbol{P}_{yz})$, where $x, y, z \in \{1, 2\}$ are appropriate indices. The criterion is then defined by the equations:

$$\begin{aligned} d_{\perp 1} = \min(&\max(d_\perp(l_1, \boldsymbol{P}_{21}), d_\perp(l_1, \boldsymbol{P}_{22})), \\ &\max(d_\perp(l_2, \boldsymbol{P}_{11}), d_\perp(l_2, \boldsymbol{P}_{12}))), \\ d_{\perp 2} = \min(&\min(d_\perp(l_1, \boldsymbol{P}_{21}), d_\perp(l_1, \boldsymbol{P}_{22})), \\ &\min(d_\perp(l_2, \boldsymbol{P}_{11}), d_\perp(l_2, \boldsymbol{P}_{12}))), \end{aligned} \tag{7.4}$$

$$w_i = \frac{d_{\perp i}}{d_{\perp 1} + d_{\perp 2}} \quad \text{for } i = \{1, 2\}, \tag{7.5}$$

$$d_{crit}(l_1, l_2) = \frac{1}{2}\left(w_1 d_{\perp 1} + w_2 d_{\perp 2}\right). \tag{7.6}$$

Many other criteria can be found in literature, but although they fulfil slightly different requirements, in general, the concept is quite similar - usage of distances between points, rarely the angle of the examined line segment pair, all of them combined using min()/max() functions. Basic Hausdorff distance (7.2), and many others not mentioned here, does not even give zero results for line segments lying on the same line. On the other hand, modified version of this criterion (7.3) gives zero results for every collinear pair of line segments, which is not desirable as well.

The mentioned criteria definitely fulfil the task they were designed for, but the properties do not meet the requirements of robotic mapping and their formulation prevents possible optimizations for better performance. The Section 7.2 describes a novel criterion, which overcomes these limitations.

7.2 Similarity of vector maps with known correspondences

As stated in the previous section, a wide range of requirements is put on the similarity criteria under different conditions. This leads to a set of algorithms with

different properties for particular situations, rather than a single overcomplicated method for everything. Specialization also enables deeper optimization, which is necessary in time and resource critical applications. Solutions mentioned above represent possible approaches, but 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. The method described further in this section satisfies the requirements and solves the issues described. Same as the appropriate part of the introduction, the following text is directly adopted from the already published the paper [149]] written by the author of this thesis.

Mobile robotics and SLAM require 2D similarity criterion variant 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.

7.2.1 Area based criterion function

The main thought behind the design of the presented criterion is the following: If the similarity of two points (zero dimensional objects) in higher dimensional spaces is a distance, then the similarity of two line segments (1D objects) should be defined by an area (in more than one dimensional spaces). For the purpose of the criterion, which should return zero as an extremum for a certain input, the square of the area is going to be the criterion value. This approach deals with a possible negative sign of the area without the need of an absolute value and ensures, that a zero is the minimal possible output of the computation.

In the following computations, the two arbitrary line segments defined by the end-points $\boldsymbol{AB}$ and $\boldsymbol{CD}$ as depicted in the Figure 7.1 are being used. Further equations frequently contain a substitution: $\mathbf{x} = \boldsymbol{B} - \boldsymbol{A}$ and $\boldsymbol{y} = \boldsymbol{D} - \boldsymbol{C}$.

One of the requirements stated above demands zero output, if both examined line segments belong to the same line. This is satisfied by the squared area of the parallelogram defined by the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ (as depicted in Figure 7.1):

$$S^2_{ABCD}(l_1, l_2) = \| \, (\boldsymbol{B} - \boldsymbol{A}) \times (\boldsymbol{D} - \boldsymbol{C}) \, \|^2 = \| \, \boldsymbol{x} \times \boldsymbol{y} \, \|^2 \, . \tag{7.7}$$

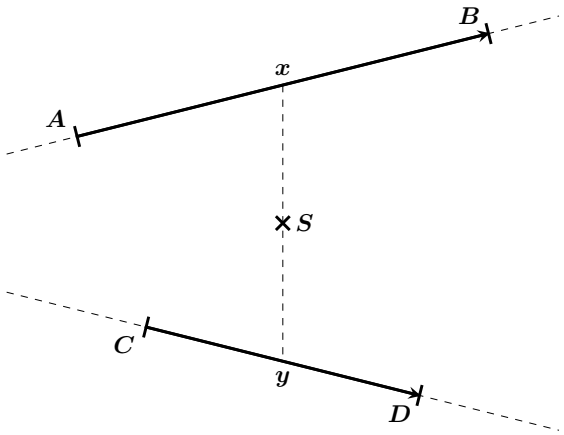


Fig. 7.1: An example of two line segments l_1 defined by points $\boldsymbol{AB}$ and l_2 defined by points $\boldsymbol{CD}$ and the middle point $\boldsymbol{S}$.

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. A brief intuitive examination of the equation (7.7) reveals the same weakness as has the Modified Hausdorff distance criterion (7.3): The result is zero for any collinear pair $\{l_1, l_2\}$. To obtain fully functional criterion, this rule is further narrowed by introduction of the squared area of the triangle $\boldsymbol{ABS}$, which is defined by:

$$S^2_{ABS}(l_1, l_2) = \frac{1}{4} \parallel (\boldsymbol{B} - \boldsymbol{A}) \times (\boldsymbol{S} - \boldsymbol{A}) \parallel^2, \tag{7.8}$$

where $\boldsymbol{S} = (\boldsymbol{A} + \boldsymbol{B} + \boldsymbol{C} + \boldsymbol{D})/4$. This function is zero for any pair $\{l_1, l_2\}$, where $((\boldsymbol{C} + \boldsymbol{D})/2) \in (\boldsymbol{A} + t\boldsymbol{x})$ and $t \in \mathbb{R}$.

By summation of the equations (7.7) and (7.8) the final criterion function is defined:

$$S_{crit}(l_1, l_2) = S^2_{ABCD}(l_1, l_2) + 64 S^2_{ABS}(l_1, l_2). \tag{7.9}$$

The coefficient 64 balances the influence of both parts of the criterion, because S_{ABCD} is significantly larger than S_{ABS}. The exact value of the constant is justified in the following subsection, where the criterion is examined deeper.

7.2.2 Properties of the criterion

So far, the criterion was developed using intuitive understanding of the geometrical properties of the cross product, but this can hardly prove the concept to be working at all conditions.

To provide a mathematical proof of the existence and shape of the minimum of the criterion function, every possible mutual position and length of both line

segments l_1 and l_2 needs to be examined, which means eight variables in total. Situation can be simplified by understanding the geometrical properties of the criterion. Both area functions (7.7) and (7.8) are independent of the position of the origin, because all vectors resulting form the subtractions become invariant to translation. Invariance of the area functions with respect to rotation of both line segments is evident from an alternative form of equation (7.7):

$$S^2_{ABCD}(l_1, l_2) = \| \boldsymbol{x} \times \boldsymbol{y} \|^2 = \| \boldsymbol{x} \|^2 \| \boldsymbol{y} \|^2 \sin^2(\alpha). \tag{7.10}$$

Since lengths of the vectors and the angle between them are not affected by the rotation of the whole pair and formula (7.8) can be rewritten in the exact same way, it is evident, that the criterion provides results independent on that kind of transformation.

The value of the criterion function (7.9) is definitely dependent on the scale, but for the purpose of minimum search, that dependency is irrelevant. Multiplying a parabolic function by a constant does not affect the location of its minimum.

Combination of the previous findings implies a remarkable simplification of the minimum search task. Thanks to the invariance of the criterion to the translation and rotation and omission of the scale factor, we can fix one line segment at a constant length and position and examine only the remaining four independent variables defining position and length of the second one. Minimum of the criterion function (7.9) is then given by the solution of the following system of equations:

$$\begin{aligned}
\frac{\partial S_{crit}}{\partial C_x} &= 0, \\
\frac{\partial S_{crit}}{\partial C_y} &= 2(C_y - D_y) + 2(C_y + D_y) = 0, \\
\frac{\partial S_{crit}}{\partial D_x} &= 0, \\
\frac{\partial S_{crit}}{\partial D_y} &= -2(C_y - D_y) + 2(C_y + D_y) = 0,
\end{aligned} \tag{7.11}$$

all for $l_1 = \{\boldsymbol{A}, \boldsymbol{B}\} = \{\{0, 0\}, \{1, 0\}\}$.

The equations are not simplified, because at this stage we can easily compare magnitudes of the first derivatives of both components of the criterion function (7.9). The coefficient 64 was chosen to equalize these magnitudes, which are now 2 for both components of the non-zero equations.

The system of equations (7.11) directly provides conditions for critical points of the criterion function:

$$\begin{aligned} &C_y = D_y = 0, \\ &C_x, D_x \in \mathbb{R}, \end{aligned} \tag{7.12}$$

which means any line segment lying on the x axis, where l_1 is located as well.

To reveal a nature of the critical points, the Hessian of the criterion function (7.9) is computed. Generally it is defined as:

$$\mathbf{H}_{i,j} = \frac{\partial f(x_1, \ldots, x_n)}{\partial x_i \partial x_j} \qquad \text{for } 1 \leq i, j \leq n. \tag{7.13}$$

As l_1 is set constant at the beginning of this examination, S_{crit} is a function of l_2 only, therefore $S_{crit}(C_x, C_y, D_x, D_y)$ applies. The Hessian is then:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 4 \end{bmatrix}. \tag{7.14}$$

H is positive semi-definite, which implies, that only minima or saddle points can exist in the critical points defined by the conditions (7.12). Since $S_{crit}(C_x, C_y, D_x, D_y)$ under conditions (7.12) is always zero, the continuous subspace of the critical points can only be the minimum of the function (7.9).

Now we can claim, that the criterion (7.9) truly satisfies requirements formulated at the beginning of this section. The only remaining feature to be described is an optimized procedure for similarity evaluation of multiple line segment pairs at once.

7.2.3 Expansion for a set of line segment pairs

In practice, there are a lot of situations, where two sets of line segments are being tested for similarity. An overall similarity is then given by a sum of similarities of all corresponding pairs from both sets. During scan to map matching in robotics, or image to image registration in computer vision applications, one set is often considered static (remains constant during computation) and the other is dynamic (i.e. manipulated using some kind of transformation). The transformation is iteratively adjusted to minimize the overall similarity criterion. The established similarity criteria require transformation of the second set of line segments and recalculation of the output value any time, the transformation changes. The presented criterion allows to precompute the result and then transform it in constant time, regardless the number of pairs being examined.

Let the line segment $l_1 = \{\boldsymbol{A}, \boldsymbol{B}\}$ be static and the $l_2 = \{\boldsymbol{C}, \boldsymbol{D}\}$ belong to the transformed set. The transformation is described by a rotation matrix $\mathbf{R}$ and translation vector $\boldsymbol{t}$:

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \qquad \boldsymbol{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{7.15}$$

where θ is the angle of rotation and t_x and t_y are translations in the direction of the x and y axes and affects an arbitrary point in accordance with equation $\boldsymbol{P'} = \mathbf{R}\boldsymbol{P} + \boldsymbol{t}$. Equations (7.7) and (7.8), with transformation of l_2 included, look as follows:

$$S^2_{ABCD}(l_1, l_2) = \| \, (\boldsymbol{B} - \boldsymbol{A}) \times (\mathbf{R}(\boldsymbol{D} - \boldsymbol{C})) \, \|^2, \tag{7.16}$$

$$S^2_{ABS}(l_1, l_2) = \frac{1}{64} \, \| \, (\boldsymbol{B} - \boldsymbol{A}) \times (\boldsymbol{A} + \boldsymbol{B} + \mathbf{R}(\boldsymbol{C} + \boldsymbol{D}) + 2\boldsymbol{t} - 4\boldsymbol{A}) \, \|^2 \, . \tag{7.17}$$

The overall similarity for a set of N line segment pairs is then given by:

$$\begin{aligned}
S_{tot} = & \sum_{i=1}^{N} S_{crit,i} \\
= \; & s^2 \sum (\boldsymbol{x}_i \cdot \boldsymbol{y}_i)^2 \\
& + c^2 \sum \| \, \boldsymbol{x}_i \times \boldsymbol{y}_i \, \|^2 \\
& + 2cs \sum (\boldsymbol{x}_i \cdot \boldsymbol{y}_i) \| \, \boldsymbol{x}_i \times \boldsymbol{y}_i \, \| \\
& + \sum \| \, \boldsymbol{x}_i \times \boldsymbol{v}_i \, \|^2 \\
& + 2s \sum (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) \| \, \boldsymbol{x}_i \times \boldsymbol{v}_i \, \| \\
& + 2c \sum \| \, \boldsymbol{x}_i \times \boldsymbol{z}_i \, \| \| \, \boldsymbol{x}_i \times \boldsymbol{v}_i \, \| \\
& + 4t_y \sum x_{x,i} \| \, \boldsymbol{x}_i \times \boldsymbol{v}_i \, \| \\
& - 4t_x \sum x_{y,i} \| \, \boldsymbol{x}_i \times \boldsymbol{v}_i \, \| \\
& + s^2 \sum (\boldsymbol{x}_i \cdot \boldsymbol{z}_i)^2 \\
& + 2cs \sum (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) \| \, \boldsymbol{x}_i \times \boldsymbol{z}_i \, \| \\
& + c^2 \sum \| \, \boldsymbol{x}_i \times \boldsymbol{z}_i \, \|^2 \\
& + 4st_y \sum x_{x,i} (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) \\
& - 4st_x \sum x_{y,i} (\boldsymbol{x}_i \cdot \boldsymbol{z}_i) \\
& + 4ct_y \sum x_{x,i} \| \, \boldsymbol{x}_i \times \boldsymbol{z}_i \, \| \\
& - 4ct_x \sum x_{y,i} \| \, \boldsymbol{x}_i \times \boldsymbol{z}_i \, \| \\
& + 4t_y^2 \sum x_{x,i}^2 \\
& - 8t_x t_y \sum x_{x,i} x_{y,i} \\
& + 4t_x^2 \sum x_{y,i}^2,
\end{aligned} \tag{7.18}$$

where $\boldsymbol{x} = \boldsymbol{B} - \boldsymbol{A}$, $\boldsymbol{y} = \boldsymbol{D} - \boldsymbol{C}$, $\boldsymbol{z} = \boldsymbol{D} + \boldsymbol{C}$, $\boldsymbol{v} = \boldsymbol{B} - 3\boldsymbol{A}$ and $c = \cos(\theta)$, $s = \sin(\theta)$, both from the rotation matrix $\mathbf{R}$. 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. The examples are iterative closest line algorithms by [172] and [182].

7.2.4 Experiments and results

Empirical verification is essential, when any new method is being released for practical applications. In this section, we are going to present synthetic tests proving features theoretically described in Sec. 7.2.2 and 7.2.3 and show some performance tests demonstrating advantages of the method, when similarity for a set of line segment pairs under different rigid transformations is to be computed.

All experiments were implemented in the C++ programming language and compiled with Microsoft Visual Studio 2015, using the -O2 optimization setting. No other optimizations were made to keep the tests as general as possible. The machine used for running the experiments had a four-core / eight threads, 64-bit Intel Core-i7-4790K CPU, running on 4.0 GHz. Processor cache memory is large enough to hold all of the data of every test performed. Visualization of the results was done using MATLAB 2015 computing environment.

Feature verification

For feature verification of the criterion, the methodics introduced in [188] was adopted. It clearly visualizes properties of the criterion under wide variety of conditions and an interested reader may find results for other line segment distance functions in the cited paper in the given format. Unification of testing methods should help to compare the methods and choose the right criterion function for the particular needs.

Initial arrangement of the experiment is depicted in Figure 7.2. There is a static line segment $l_1 = \{\boldsymbol{A}, \boldsymbol{B}\} = \{\{0, 0\}, \{100, 0\}\}$ and a dynamic l_2, transformed by various transformations according to the particular test. Figure 7.2 shows the positive direction of the rotation and the x axis. Rotations are always performed around the origin of the coordinate system.

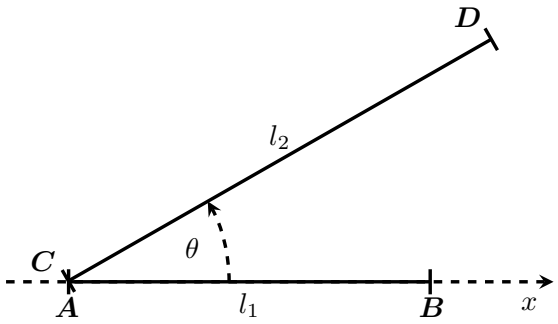


Fig. 7.2: Schematic depiction of the positions of the line segments during the verification process as introduced in [188].

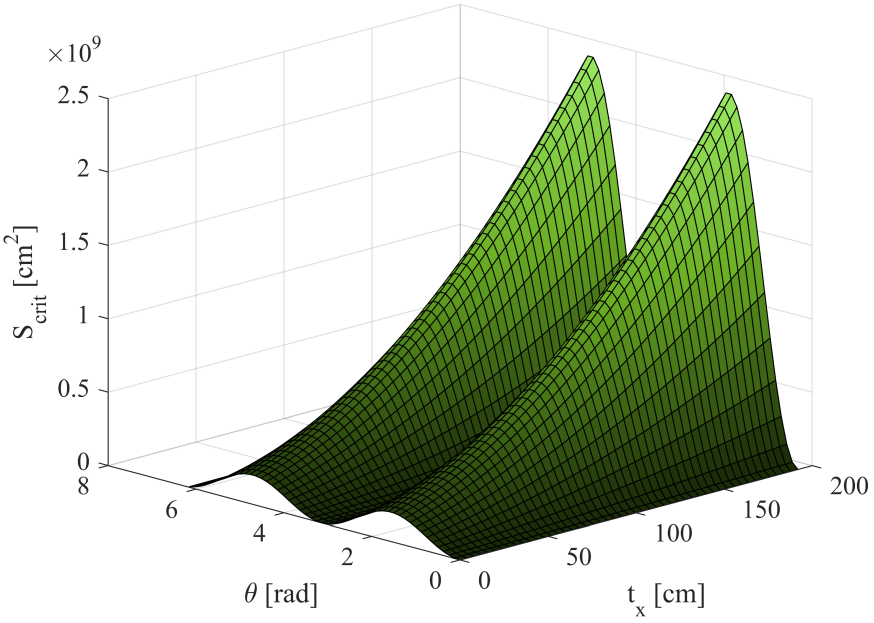


Fig. 7.3: Criterion verification: Translation of l_2 by $t_x \in [0; 200]$ centimetres followed by a rotation by $\theta \in [0; 2\pi]$ radians.

The first two tests directly follow [188]. Figure 7.3 depicts a situation, where l_2 is first translated in the direction of the x axis and then rotated by a given angle. The graph clearly shows, that for l_2 lying on the x axis, the criterion returns zero and as the translation increases and the angle closes to $\frac{\pi}{2}$ and $\frac{3}{2}\pi$ the output value grows rapidly.

Figure 7.4 shows another important case, where the length of l_2 is varied and then the line segment is rotated by the angle θ. Again, the criterion gives zero output for any l_2 coincident with the x axis and increases as the length grows and θ closes to $\frac{\pi}{2}$ or $\frac{3}{2}\pi$, which is desirable. Even for very short l_2 the nature of the criterion is

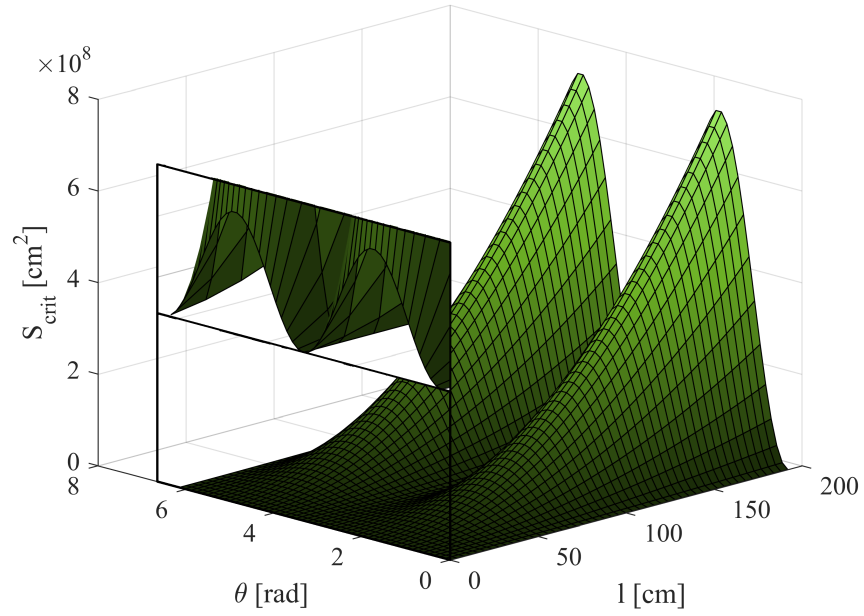


Fig. 7.4: Criterion verification: Length scaling of l_2 in the interval $[1; 200]$ centimetres followed by a rotation by $\theta \in [0; 2\pi]$ radians. Sub-plot shows the detail of the graph for very small lengths of l_2.

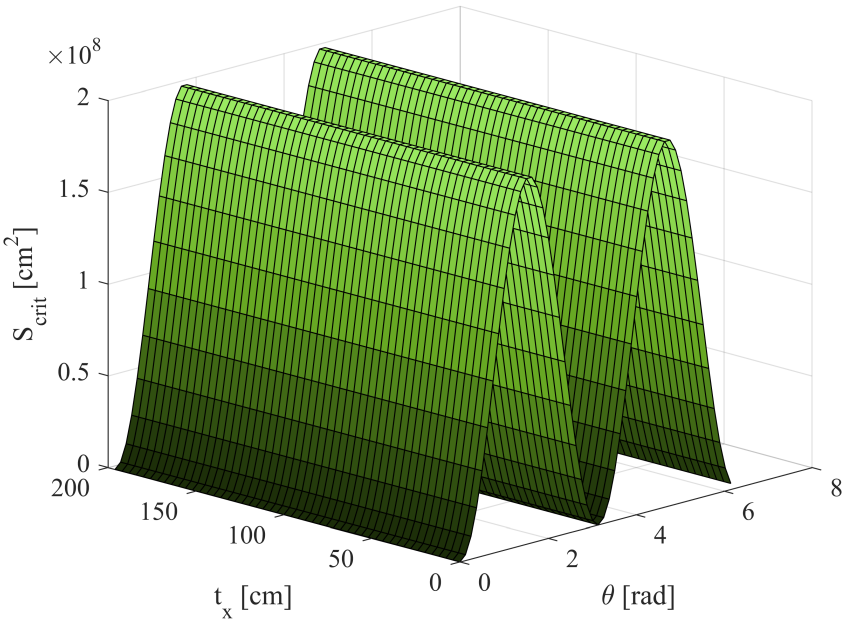


Fig. 7.5: Criterion verification: Rotation of l_2 by $\theta \in [0; 2\pi]$ radians followed by a translation by $t_x \in [0; 200]$ centimetres.

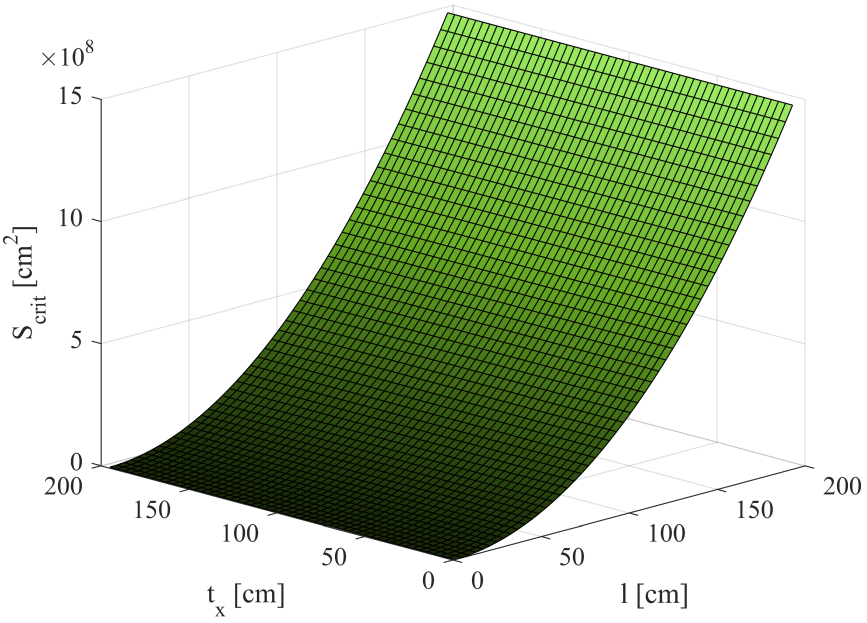


Fig. 7.6: Criterion verification: Length scaling of l_2 in the interval $[1; 200]$ centimetres followed by a rotation by a fixed angle $\pi/4$ radians and translated by $t_x \in [0; 200]$ centimetres at the end.

consistent and the detail magnified in the sub-plot in Figure 7.4 corresponds to the harmonic shape observed in Figure 7.3.

The two following tests are meant to demonstrate, that for any l_2 moving along the l_1 (i.e. x axis in these experiments), the value of the criterion remains the same. Figure 7.5 shows a situation, where l_2 is first rotated and then translated along the x axis. The graph clearly demonstrates, that the translation has no effect and the enumerated similarity remains the same.

Similar behaviour appears, when l_2 is scaled at first, then rotated by the fixed angle $\frac{\pi}{4}$ and translated along l_1. The output of the criterion (see Figure 7.6) is quadratically dependant on the length of l_2, but the translation does not affect it in any way.

All four experiments prove the theoretical findings from Section 7.2.1. If l_2 and l_1 lie on the same line, the criterion returns zero, which corresponds to conditions (7.12). The tests also illustrate the fact, that the value of the criterion is not affected by translation of l_2 in the direction of l_1, which is mathematically proved in the system of equations (7.11).

Performance verification

Though the equation (7.18) might seem enormous at first, there are many repeating terms, so the sums can be precomputed with reasonable amount of additions and multiplications. The same applies to later evaluation for various transformations. In fact, careful examination of equations (7.16) and (7.17) reveals, that the number of basic floating point operations is roughly the same in both cases.

In this set of tests the performance of a naive and the optimized algorithms is compared. The naive implementation of the criterion function for a set of line segment pairs stems from the equations (7.16) and (7.17). First, it computes the transformation of the dynamic line segments and then the errors. Contrary, the optimized algorithm first computes sums of the equation (7.18) and then evaluates the criterion for each particular transformation.

Figures 7.7 and 7.8 show the timings of the proposed criterion function. Both exhibit the predicted behaviour stated in Section 7.2.3. Naive implementation corresponds to $Q(NT)$ computation complexity, while the optimized algorithm is $O(N + T)$. To emphasize the benefits of the optimization, Figure 7.9 shows the speed-up over the basic version. The performance gains are present even for the lowest numbers of transformations and line segment pairs, but the small percen-

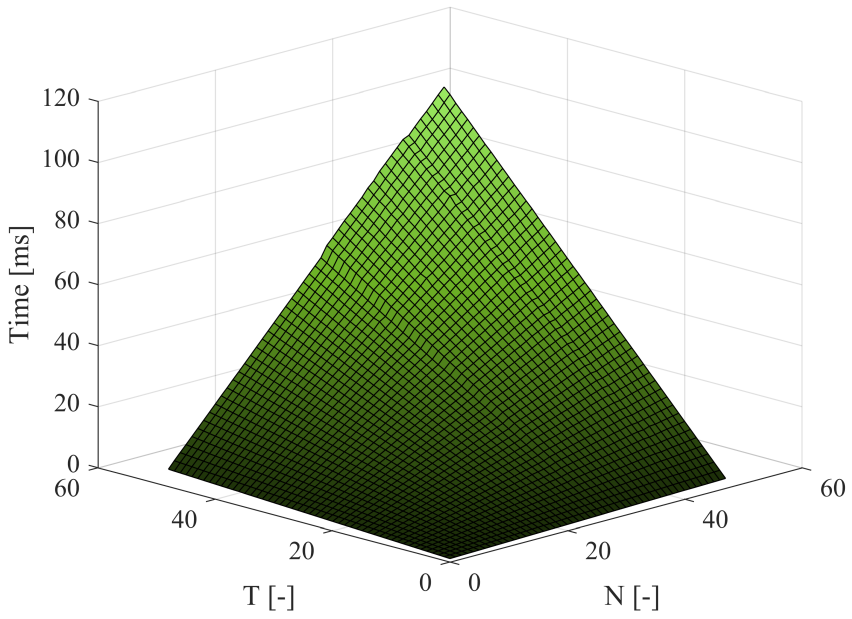


Fig. 7.7: Processing time of a naive implementation of the criterion for various number of line segment pairs being processed (N) and transformations performed (T).

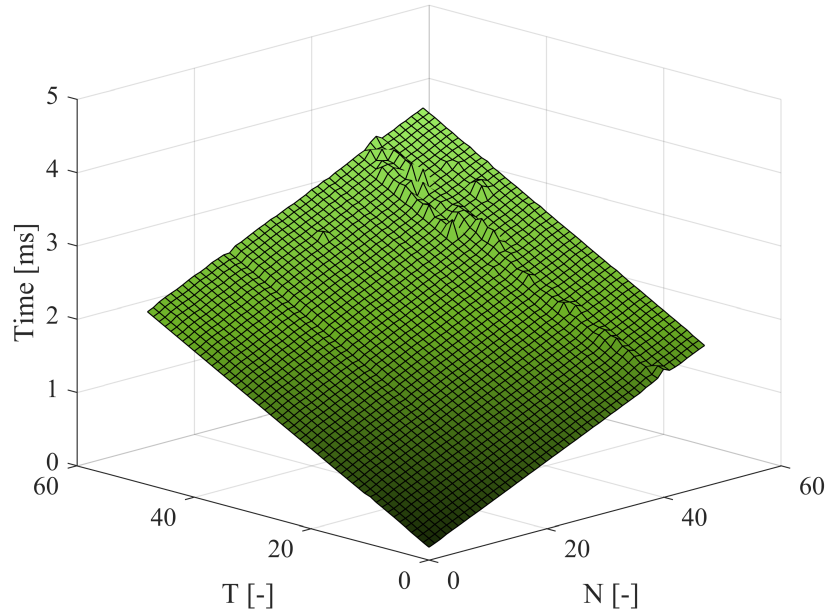


Fig. 7.8: Processing time of the optimized implementation criterion for various number of line segment pairs being processed (N) and transformations performed (T).

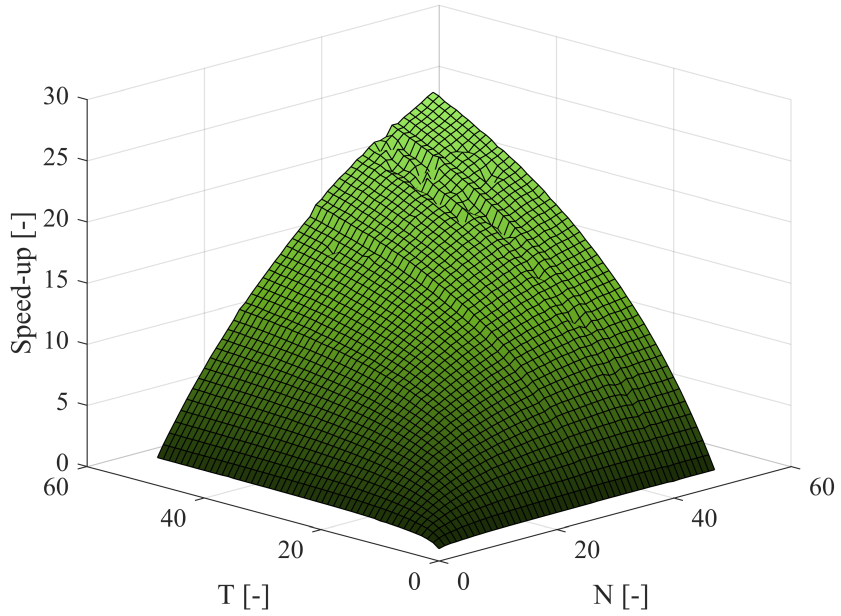


Fig. 7.9: Seed-up of the optimized criterion over a naive implementation for various number of line segment pairs being processed (N) and transformations performed (T).

tage of improvement is highly implementation dependant and redeemed by a larger memory footprint of the optimized algorithm. The most significant improvements can be observed, when both N and T grow up, which is a frequent case. In such situation, the large performance gains are doubtless.

7.3 Registration of the vectorized laser scans

Previous section described a criterion for evaluation of the similarity of line segment pairs with given mutual pose. For minimization of this criterion, an iterative process of fitting would have to be devised, but for known correspondences, there is another possible way. 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. For the one step method in this section, two control mechanisms for failure detection were devised, which should enhance its usability in safety critical projects.

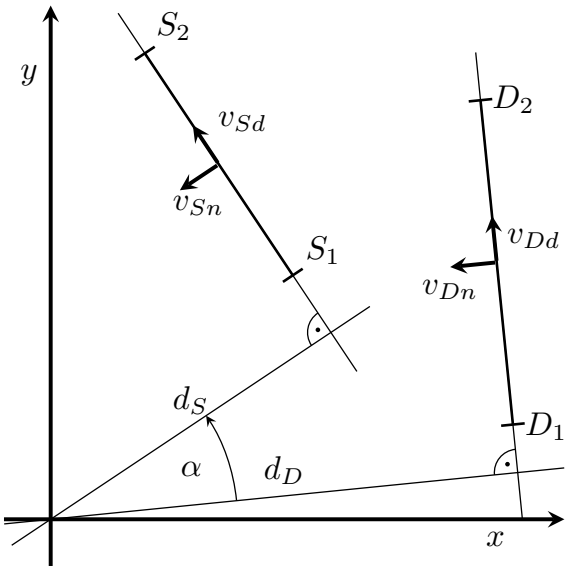


Fig. 7.10: Static (S) and dynamic (D) line segments during the fitting process with directional and normal vectors marked out.

7.3.1 Optimal transformation for vector image fitting

Equations in this section stem from the situation depicted in Figure 7.10. 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:

$$p_i : \begin{bmatrix} (d_{Di} - d_{Si})\boldsymbol{v}_{Sni} \\ \alpha_i \end{bmatrix} + \delta_i \begin{bmatrix} \boldsymbol{v}_{Sdi} \\ 0 \end{bmatrix} = \boldsymbol{P}_i + \delta_i \boldsymbol{v}_i \tag{7.19}$$

where all the variables in the first part correspond to the Fig. 7.10 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. Illustrative example of possible result is shown in Figure 7.11.

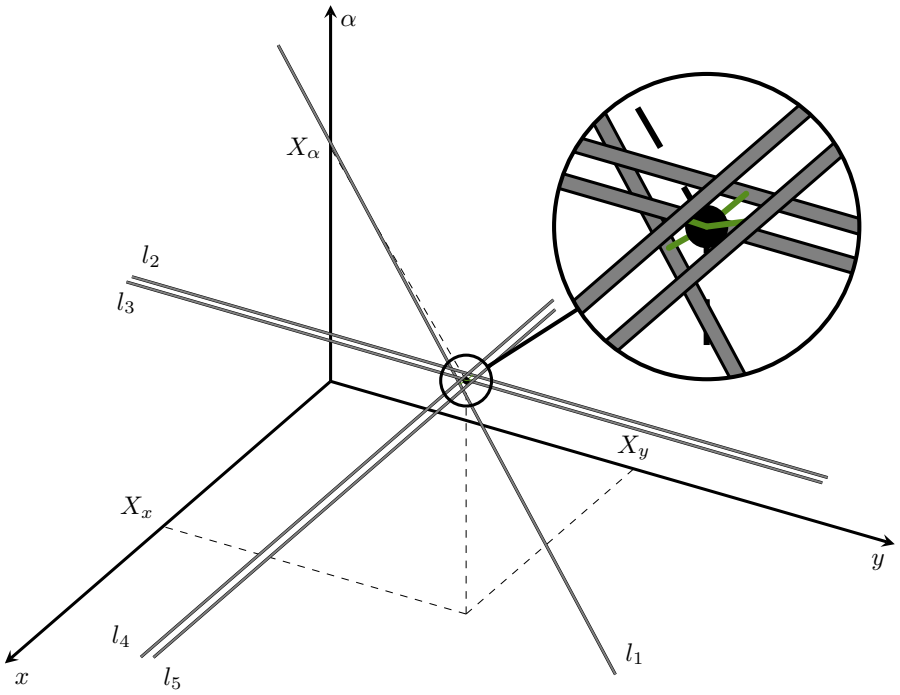


Fig. 7.11: An example of the search for an optimal transformation in the space of all possible transformations x, y, α. The gray lines depict the perfect transformations for particular corresponding line segment pairs. The optimal transformation is shown as a black dot in the magnified area.

The optimal transformation is then simply computed as the closest one to all precise transformations, as shown in the magnified area of the Figure 7.11. Point to line distance in 3D is given by:

$$l_i = \frac{\parallel (\boldsymbol{x} - \boldsymbol{P}_i) \times \boldsymbol{v}_i \parallel}{\parallel \boldsymbol{v}_i \parallel}, \tag{7.20}$$

where $\boldsymbol{x}$ is the wanted transformation and $\boldsymbol{P}_i$ and $\boldsymbol{v}_i$ come from equation (7.19). Searching for $\boldsymbol{x}$ with arbitrary weight w for each correspondence leads to a standard least squares problem:

$$\sum_{i=1}^{N} w_i \frac{\partial l_i^2(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial(\parallel \boldsymbol{x} - \boldsymbol{P}_i \parallel^2 - (\boldsymbol{x} - \boldsymbol{P}_i) \cdot \boldsymbol{v}_i)^2}{\partial \boldsymbol{x}} = 0. \tag{7.21}$$

Partial differentiation of the previous formula gives the following vector equation:

$$\sum_{i=1}^{N} w_i((\boldsymbol{x} - \boldsymbol{P}_i - \boldsymbol{v}_i((\boldsymbol{x} - \boldsymbol{P}_i) \cdot \boldsymbol{v}_i)) = 0, \tag{7.22}$$

which can be rewritten into the form:

$$\sum_{i=1}^{N} w_i \boldsymbol{x} - \sum_{i=1}^{N} w_i \boldsymbol{v}_i (\boldsymbol{x} \cdot \boldsymbol{v}_i) = \sum_{i=1}^{N} w_i \boldsymbol{P}_i - \sum_{i=1}^{N} w_i \boldsymbol{v}_i (\boldsymbol{P}_i \cdot \boldsymbol{v}_i). \tag{7.23}$$

Rearranged into a set of linear equations, (7.23) is as follows:

$$\begin{bmatrix} \sum w_i v_{yi}^2 & -\sum w_i v_{xi} v_{yi} & 0 \\ -\sum w_i v_{xi} v_{yi} & \sum w_i v_{xi}^2 & 0 \\ 0 & 0 & \sum w_i \end{bmatrix} \begin{bmatrix} x_x \\ x_y \\ x_\alpha \end{bmatrix} = \begin{bmatrix} \sum w_i P_{xi} v_{yi}^2 - \sum w_i P_{yi} v_{xi} v_{yi} \\ \sum w_i P_{yi} v_{xi}^2 - \sum w_i P_{xi} v_{xi} v_{yi} \\ \sum w_i P_{\alpha i} \end{bmatrix}. \tag{7.24}$$

The equation (7.1) is rather long, so an abbreviated form is going to be used in further text. The simplified notation is:

$$\begin{bmatrix} S_{y2} & -S_{xy} & 0 \\ -S_{xy} & S_{x2} & 0 \\ 0 & 0 & S_w \end{bmatrix} \begin{bmatrix} x_x \\ x_y \\ x_\alpha \end{bmatrix} = \begin{bmatrix} S_{Px} \\ S_{Py} \\ S_{P\alpha} \end{bmatrix}. \tag{7.25}$$

Solution is obtained in a standard way using the determinants:

$$\begin{aligned} D &= S_{x2} S_{y2} - S_{xy}^2, \\ D_1 &= S_{Px} S_{x2} - S_{xy} S_{Py}, \\ D_2 &= S_{Py} S_{y2} - S_{xy} S_{Px}. \end{aligned} \tag{7.26}$$

The optimal translation is directly given by D, D_1 and D_2:

$$\begin{aligned} x_x &= \frac{D_1}{D}, \\ x_y &= \frac{D_2}{D}, \end{aligned} \tag{7.27}$$

while the rotation computation is not as straightforward as it might seem. Since the angle is a circular quantity, a classical average computation:

$$x_\alpha = \frac{S_{P\alpha}}{S_w} \tag{7.28}$$

might lead to an unexpected result. When averaging two angles, there are always two possible outcomes due to circularity, and there is no guarantee, that the right one will be obtained. To obtain the correct results, the Mitsuta's averaging method [190] is usable. The implementation in the testing code copies the algorithm in [191]. Deeper treatment of the statistics of the circular quantities from a programmer's point of view can be found in [192].

At this point, an optimal transformation for a given set of corresponding line segments is successfully computed. The only possibility of failure stems from the division by the discriminant in (7.27), because if all static line segments are collinear, the discriminant D equals zero. The system of equations is not solvable and an error should be reported.

7.3.2 Reliability evaluation

Though mathematically is everything all right, reliability of the computation does not get immediately hundred percent, once the determinant D is not zero. An underdetermined system of equations is only an extreme case of a wider problem of *nearly* collinear lines, which happen in practice much more often than an absolute collinearity. Consider an example in Figure 7.12. There are two static line segments S_1 and S_2 and their corresponding counterparts D_1 and D_2, both nearly but not exactly collinear. Such a situation can easily happen, if a robot maps a long narrow corridor. The deviations from collinearity are caused by a measurement noise. The optimal transformation in this case means shift of the dynamic line segments by three units to the right. Although the solution is mathematically sound, a human observer can clearly spot what is wrong: A tiny deviation in pose of the line segments caused a huge translation. Maybe, this was the issue with approximations, which led the author of [171] to the claim about their susceptibility to noise.

In fact, the problem in Figure 7.12 does not refer to a bad approximation, but to an inadequate set of line segments chosen for registration. A need for some

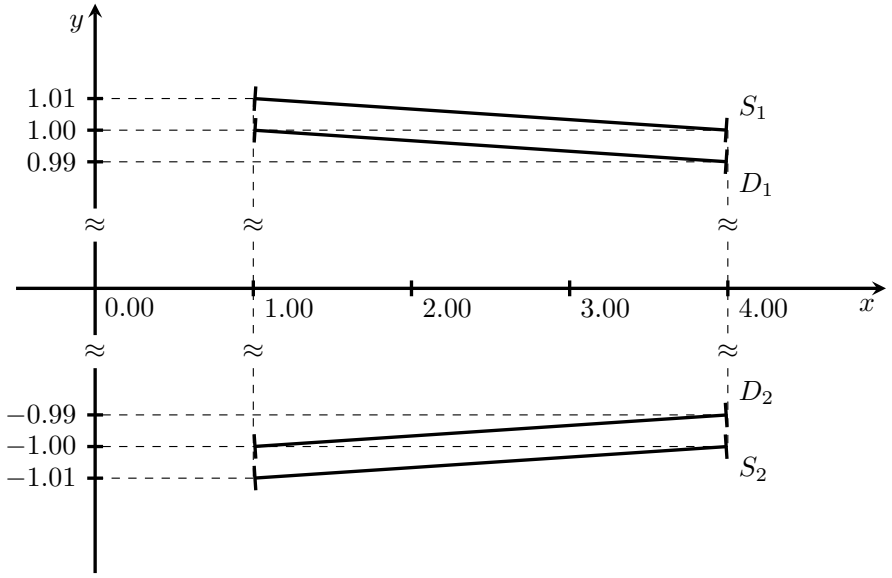


Fig. 7.12: Nearly collinear static (S_i) and dynamic (D_i) line segments during the registration. Because the pairs 1 and 2 are nearly collinear, a tiny deviation in direction leads to a large optimal transformation.

continuous measure of reliability for the transformation computed in the previous section is obvious. One extreme of the metric should correspond to a completely unsolvable situation arising if the line segments happen to be exactly collinear and the opposite case should cover the perfectly solvable cases, when there are two perpendicular pairs. Such a metric is derived in the rest of this section.

As a base for the reliability evaluation, the unit direction vectors of the lines in the transformation space from the equation (7.19) are used. If these vectors point mostly in the same direction, the reliability should be low an vice versa. The main principle of the computation is shown in Figure 7.13. The end points of the direction vectors are shown on the unit circle and an opposite of each of them is added as well. This operation ensures, that the mean of their coordinates remains zero under any circumstances. For the resulting set of points, the correlation matrix is computed. An angle ϵ formed using the eigenvectors $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ then can be used to express the reliability, because if the problem is well solvable, the eigenvectors are similar in length and the ϵ is close to $\pi/4$. In case of nearly collinear lines, one eigenvector will be significantly longer than the other one and ϵ will be close to either $\pi/2$ or zero.

Sums of the direction vectors are already available from the computations described in Section 7.3.1. Because of the additional opposite vectors the sums in the

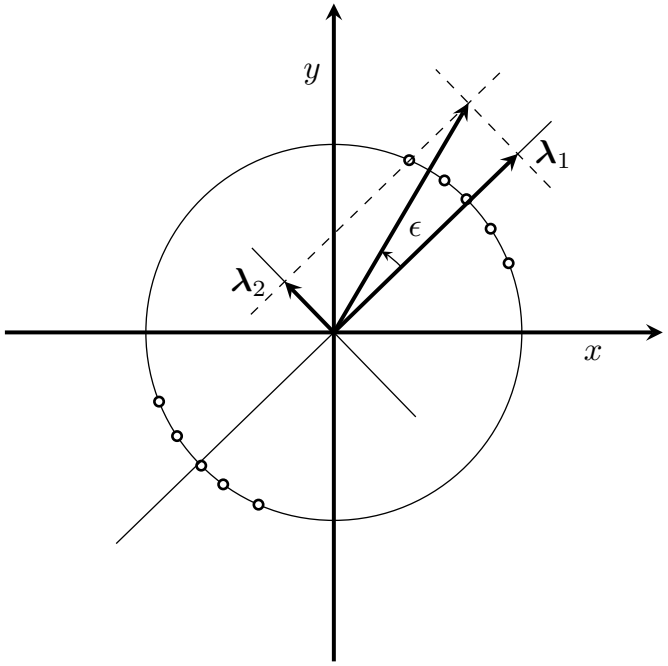


Fig. 7.13: A principal component analysis is used for reliability evaluation of the optimal fitting computation. Small circles represent direction vectors of the static line segments and their opposite counterparts. Eigenvectors $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ represent the linearly uncorrelated variations and the angle ϵ is used to define the actual reliability.

reliability evaluation change as follows:

$$\begin{array}{ccc} \sum_{i=1}^{N} w_i x_i \Rightarrow 0, & \sum_{i=1}^{N} w_i y_i \Rightarrow 0, & \sum_{i=1}^{N} w_i \Rightarrow 2\sum_{i=1}^{N} w_i, \\ \sum_{i=1}^{N} w_i x_i^2 \Rightarrow 2\sum_{i=1}^{N} w_i x_i^2, & \sum_{i=1}^{N} w_i y_i^2 \Rightarrow 2\sum_{i=1}^{N} w_i y_i^2, & \sum_{i=1}^{N} w_i x_i y_i \Rightarrow 2\sum_{i=1}^{N} w_i x_i y_i. \end{array} \tag{7.29}$$

Elements of the covariance matrix are:

$$\begin{aligned} \mathrm{Var}(x) &= \tfrac{1}{\sum w_i} \sum_{i=1}^{N} w_i v_{xi}^2, \\ \mathrm{Var}(y) &= \tfrac{1}{\sum w_i} \sum_{i=1}^{N} w_i v_{yi}^2, \\ \mathrm{Cov}(x,y) &= \tfrac{1}{\sum w_i} \sum_{i=1}^{N} w_i v_{xi} v_{yi}. \end{aligned} \tag{7.30}$$

Using identities (7.30), the covariance matrix $\mathbf{E}$ is then:

$$\mathbf{E} = \begin{bmatrix} \mathrm{Var}(x) & \mathrm{Cov}(x,y) \\ \mathrm{Cov}(x,y) & \mathrm{Var}(y) \end{bmatrix}. \tag{7.31}$$

Eigenvalues are computed using the standard formula $\mathrm{Det}(\mathbf{E} - \lambda\mathbf{I}) = 0$, which leads to a quadratic equation:

$$\lambda^2 - \lambda\frac{\left(\sum w_i v_{xi}^2 + \sum w_i v_{yi}^2\right)}{\sum w_i} + \frac{\left(\sum w_i v_{xi}^2 \sum w_i v_{yi}^2 - \left(\sum w_i v_{xi} v_{yi}\right)^2\right)}{\left(\sum w_i\right)^2} = 0. \tag{7.32}$$

Since:

$$\sum w_i v_{xi}^2 + \sum w_i v_{yi}^2 = \sum w_i (v_{xi}^2 + v_{yi}^2) = \sum w_i, \tag{7.33}$$

and

$$\frac{\left(\sum w_i v_{xi}^2 \sum w_i v_{yi}^2 - \left(\sum w_i v_{xi} v_{yi}\right)^2\right)}{\left(\sum w_i\right)^2} = \mathrm{Det}(\mathbf{E}), \tag{7.34}$$

the equation (7.32) can be simplified in the following manner:

$$\lambda^2 - 2\lambda + \mathrm{Det}(\mathbf{E}) = 0. \tag{7.35}$$

The coefficient 2 before $\lambda_{1,2}$ is present because of the additional opposite vectors. The actual eigenvalues can be easily computed from the equation:

$$\lambda_{1,2} = 1 \pm \sqrt{1 - \mathrm{Det}(\mathbf{E})}. \tag{7.36}$$

Now since $\mathrm{Det}(\mathbf{E}) = \lambda_1\lambda_2$, the following identities apply:

$$\begin{aligned} \lambda_{1,2} &= 1 \pm \sqrt{1 - \lambda_1\lambda_2}, \\ \lambda_{1,2}^2 - 2\lambda_{1,2} + 1 &= 1 - \lambda_1\lambda_2, \\ \lambda_1^2 - 2\lambda_1 = \lambda_1\lambda_2 &= \lambda_2^2 - 2\lambda_2, \\ \tfrac{\lambda_1}{2} + \tfrac{\lambda_2}{2} &= 1. \end{aligned} \tag{7.37}$$

For the angle ϵ from Figure 7.13, the well known formulas $\sin^2\epsilon + \cos^2\epsilon = 1$ and $\sin(2\epsilon) = 2\sin\epsilon\cos\epsilon$ applies, which means that the reliability R can be defined as:

$$R = \sin(2\epsilon) = 2\sqrt{\frac{\lambda_1}{2} \cdot \frac{\lambda_2}{2}} = \sqrt{\lambda_1\lambda_2} = \sqrt{\mathrm{Det}(\mathbf{E})}. \tag{7.38}$$

The definition of the reliability directly shows its properties. Because the positive result of the square root is taken, the value of R can change in the interval $[0; 1]$, where zero demarcates an unsolvable situation (determinant D in equations (7.26) will be zero) and one means the highest reliability possible. There is a smooth transition between these two states (corresponds to the shape of $\sin(2\epsilon)$), so the unreliable results caused by nearly collinear line segments can be easily detected.

7.3.3 Ambiguity evaluation

Aside from reliability, there is another marker describing the results of registration process, which is useful in practice. The computation above provides an optimal transformation minimizing its internal criterion for any given set of line segment pairs, but there is no measure describing, how close is the output to the ideal transformations identified at the beginning. For illustration, see an artificial example in Figure 7.14. The invalid correspondences lead to a large trade-off in the optimal transformation computation, which can be detected and reported to the user of the algorithm. In principle, this means an exposure of the internal criterion function with some additional options.

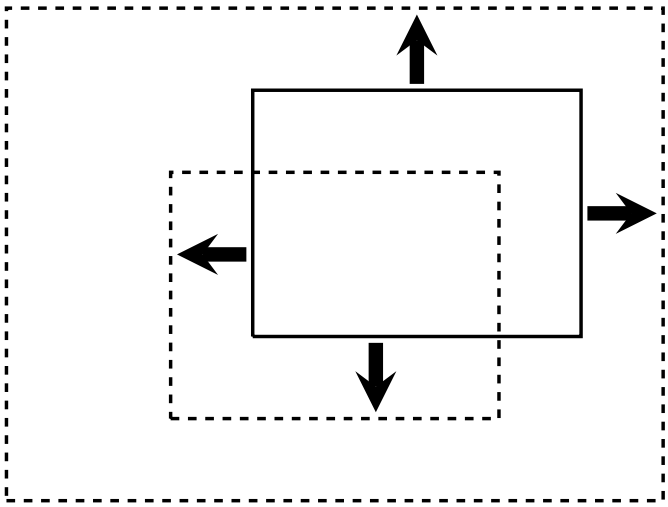


Fig. 7.14: The dashed line represents a static map and the rectangle being registered is depicted using a continuous line. The correspondences (bold arrows) are clearly set in a wrong way, so the optimal transformation is a highly compromise solution. This ambiguity should be reported along with the actual transformation and the reliability metric.

A natural choice is therefore the sum of squared distances differentiated in the equation (7.21). Because the optimal transformation is already found, the sum of squared distances can be expressed as follows:

$$\begin{aligned}\sum l_i^2(\boldsymbol{x}) = \sum w_i((x_\alpha - P_{\alpha i})^2 + (x_x - P_{xi})^2 v_{yi}^2 \\ + (x_y - P_{yi})^2 v_{xi}^2 - 2(x_x - P_{xi})(x_y - P_{yi}) v_{xi} v_{yi}),\end{aligned} \tag{7.39}$$

where x_x, x_y and x_α are parameters of the optimal transformation and the rest of the variables comes from the definition (7.19). The formula above can be directly

expanded to enable precomputation of the sums:

$$\begin{aligned}\sum l_i^2(\boldsymbol{x}) = {} & x_\alpha^2 \sum w_i - 2x_\alpha \sum w_i P_{\alpha i} + \sum w_i P_{\alpha i}^2 \\ & + x_x^2 \sum w_i v_{yi}^2 + x_y^2 \sum w_i v_{xi}^2 \\ & - 2x_x x_y \sum w_i v_{xi} v_{yi} \\ & - 2x_x (\sum w_i P_{xi} v_{yi}^2 - \sum w_i P_{yi} v_{xi} v_{yi}) \\ & - 2x_y (\sum w_i P_{yi} v_{xi}^2 - \sum w_i P_{yi} v_{xi} v_{yi}) \\ & + \sum w_i (P_{xi}^2 v_{yi}^2 - 2P_{xi} P_{yi} v_{xi} v_{yi} + P_{yi}^2 v_{xi}^2),\end{aligned} \tag{7.40}$$

but there are two issues with the straightforward approach. First, there are several sums, which would have to be evaluated in addition to those needed in previous computations. Second, the distance in rotational dimension is invariant to the scale of the shape being registered, but the distance in translation is not. To allow to independently scale the contributions of the rotational and translational distances, it is useful to think about them as if they were the tensed strings. After this imaginary conversion, both can be given a separate stiffness coefficients and freely summed back together. This operation also polishes up the physical inconsistency in units, because the angular and linear quantities no longer mix up together.

Total potential energy of rotational part of transformation can be expressed as:

$$E_\alpha = k_\alpha \left(\sum w_i P_{\alpha i}^2 - x_\alpha \sum w_i P_{\alpha i} \right), \tag{7.41}$$

where k_α is a stiffness coefficient for the rotational movement and $\sum w_i P_{\alpha i}^2$ is a new sum, which needs to be precomputed. Methodology for averaging circular quantities in [191] covers this matter as well.

Total potential energy of the translational part of the transformation is somewhat more complicated to derive, but it can be simplified down to the from:

$$E_{xy} = k_{xy} \left(\sum w_i (P_{xi} v_{yi} - P_{yi} v_{xi})^2 - \sum w_i (x_x v_{yi} - x_y v_{xi})^2 \right), \tag{7.42}$$

where k_{xy} is a stiffness coefficient for the translational part of the transformation. The second sum in the equation is not suitable for precomputation, because it contains the transformation parameters x_x and x_y inside. For practical operation, a slightly longer rearrangement is more beneficial:

$$E_{xy} = k_{xy} \left(\sum w_i (P_{xi} v_{yi} - P_{yi} v_{xi})^2 - x_x^2 \sum w_i v_{yi}^2 + 2x_x x_y \sum w_i v_{xi} v_{yi} - x_y^2 \sum w_i v_{xi}^2 \right). \tag{7.43}$$

Similar to the equation (7.41), there is only one additional sum to be precomputed. Ambiguity A of the solution is then expressed by a sum of the angular and linear energy in a simple formula:

$$A = E_\alpha + E_{xy}. \tag{7.44}$$

Ambiguity evaluation provides a control mechanism expressing, how many compromises were needed to find the optimal transformation. If the match is perfect, the lines of optimal transformations intersect at one point and the ambivalence is zero. Noised measurements from practical experiments exhibit some ambivalence, but it stays limited. If the limit is exceeded, a strong suspicion of badly set correspondences is in place.

7.4 Correspondences extracting algorithm

The matter presented so far in this chapter described some theoretical findings on registration of line segment sets. An important assumption employed in both methods in Sections 7.2 and 7.3 was a prior knowledge of the correspondences between the registered line segments. This is obviously not a frequent case in practice and especially in laser scanning, the vectorized point clouds do not contain any data connecting them with other observations. The methods above provide a straightforward functionality presented so far, but alongside, they can be also exploited in an inverse way.

The idea of the correspondence extracting algorithm is based on testing of the various sets of possible correspondences with the methods above. Gradual process of searching through the entire space of all possible correspondences should lead to minimization of the internal criterion of a given method and result into final set of highly probable correspondences. Although the methods support addition and removal of the line segment pairs in constant time, a naive testing of all possible combinations of the possible correspondences would be extremely time consuming. The following algorithm is not yet fully tested, but the in-build heuristic rules help to reduce the complexity of the process and provide the results in a reasonable time.

The technique from Section 7.3 was used as a base fitting tool in this case. Its application is in the middle two steps of the algorithm scheme depicted in Figure 7.15. All but the last step are applicable on vector images in general and the *expected view test* is an additional filtration specific to robotic mapping. The following subsections go through the whole process and present some preliminary evaluation.

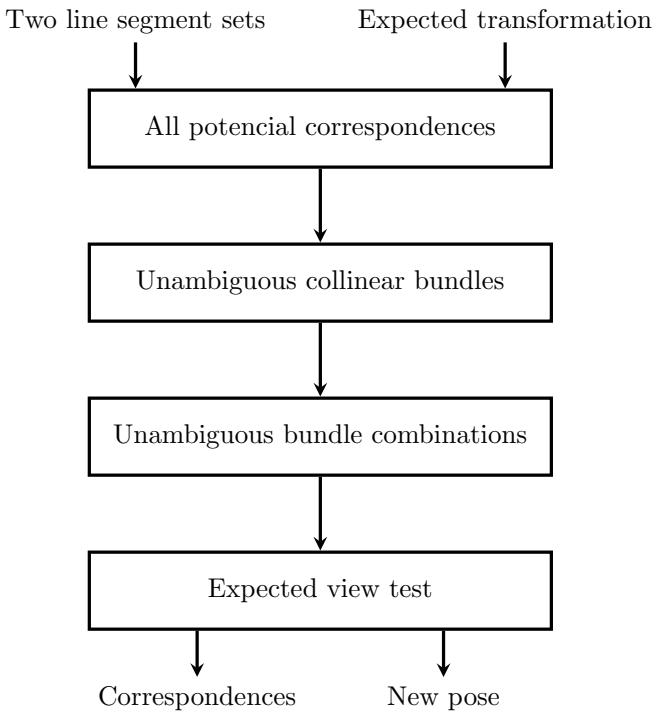


Fig. 7.15: Simplified schematic depiction of the proposed registration algorithm. The framed text describes individual algorithmic steps, while the bare text is used for the input and output data. The *expected view test* is specific for robotic mapping and should not be used in general vector image registration.

7.4.1 Extracting collision-free correspondence sets

As shown in Figure 7.15, the input for the algorithm are two sets of line segments, where one is considered static and the other is meant to be dynamically moved during the registration. The third input is their expected mutual pose, i.e. rigid transformation $T_{exp} : T_{exp}(\boldsymbol{P}) = \mathbf{R}\boldsymbol{P}+\boldsymbol{t}$ needed to move the static coordinate system to the dynamic one. An example from Figure 7.16 will be referred in the following explanation.

All potential 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 centred at T_{exp}. For every pair composed of one line segment from the static and one from the dynamic set, a set of transformations leading to perfect match is computed according to the formula (7.19). Then the set is limited to contain only the transformations, which make the line segments overlap, i.e. share at least one point. If there is an intersection of the

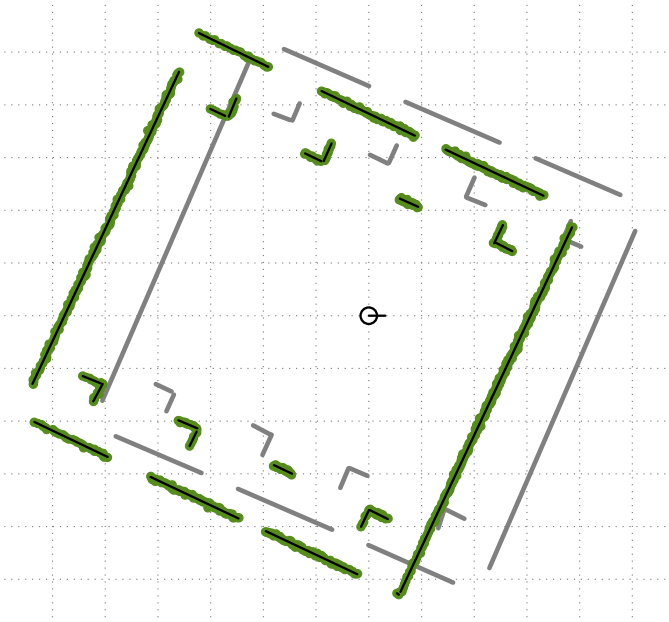
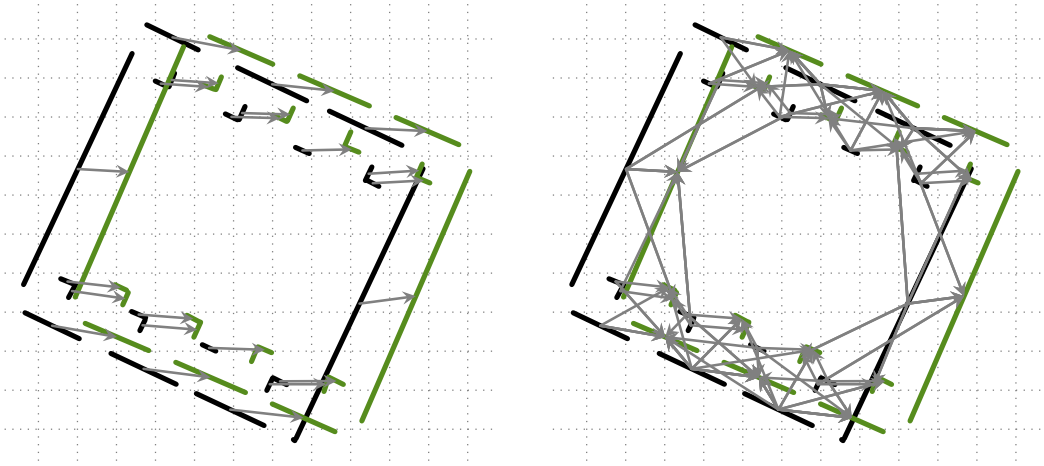


Fig. 7.16: An example of two vectorized laser scans to be registered. The gray lines demarcate an old scan, green dots represent a new point cloud, the black lines stand for its vector approximation and the mark in the middle is the last known position of the robot. The grid resolution is 50x50 centimetres.

set of permitted transformations and the set of perfect match transformations, than the examined line segments are stored as a potentially corresponding pair. The process is repeated for every combination of line segments drawn from the input sets.

Computational complexity of this step in a straightforward implementation corresponds to the product of the number of line segments in both sets. This approach can easily lead to extreme processing times, if even a single set is large enough. In robotic mapping, this is a frequent case, because the map usually contains a lot of data and its size grows in time, while the scans contain similar amount of data in every measurement. On the other hand, the set of permitted transformations defines a limited region in a map, where correspondences may occur and if a density of the map is homogeneous, than all subsequent operations deal with approximately the same amount of data, regardless the true size of the map. This was already discussed in Chapter 2 as an important optimization technique used in various SLAM systems. The benefits are evident, because the overall complexity of the registration process is then given by the algorithm used to extract the region of interest from the map. This obviously does not mean, that the subsequent operations are not worth optimization, but the extraction of the static line segment set is crucial.

Initial pose estimate and its accuracy directly impact the results of this stage



(a) High accuracy pose estimate.

(b) Low accuracy pose estimate.

Fig. 7.17: On an influence of the accuracy of the pose estimate on the number of potentially corresponding line segment pairs. Accurate estimation results in much less conflicting correspondences and makes the subsequent operations less time-consuming. (map - green, scan - black, correspondences - gray arrows)

of the registration. If the pose estimate is accurate, then the subset of all permitted transformations is small and less potential correspondences is found (see Figure 7.17a). Lower accuracy results in larger region for exploration and with more possibly matching line segments. The situation can easily lead to multiple correspondences of one line segment, which collide with each other, because they do not lie on the same line. A typical example is shown in Figure 7.17b. Although it makes the following parts of the algorithm more challenging, the later case is more frequent in practice, because accuracy of both pose estimate and the scan measurements is limited.

Unambiguous collinear bundles

The second stage of the registration process sorts the potential correspondences into unambiguous bundles of *nearly* collinear line segments. The main purpose of this operation is to separate the correspondences which do not collide with each other and which are able to be freely translated in one direction. Both properties are watched using the additional metrics *reliability* and *ambiguity* derived in Section 7.3. Reliability of a correctly assembled bundle is therefore close to zero and its ambiguity as well. 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.

Each bundle stores appropriate sums for the line segment pairs it contains. The algorithm iterates through a list of all possible correspondences an tries to add them to the already existing bundles. If the thresholds are not exceeded after the insertion, the correspondence is associated with the given bundle. In an opposite case, the sums are reverted to the previous state and another bundle is tested. If no bundle is sufficient, a new one is established. This process is linearly dependant on the number of potential correspondences and diversity of line segment directions. In a man made environment, where orthogonal structures are quite often, this diversity is usually very low (two directions in the simplest cases). The process of separation into bundles greatly reduces amount of combinations in the following assembling stage of the algorithm.

Unambiguous bundle combinations

In the last stage of the basic registration process, the bundles are combined into 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

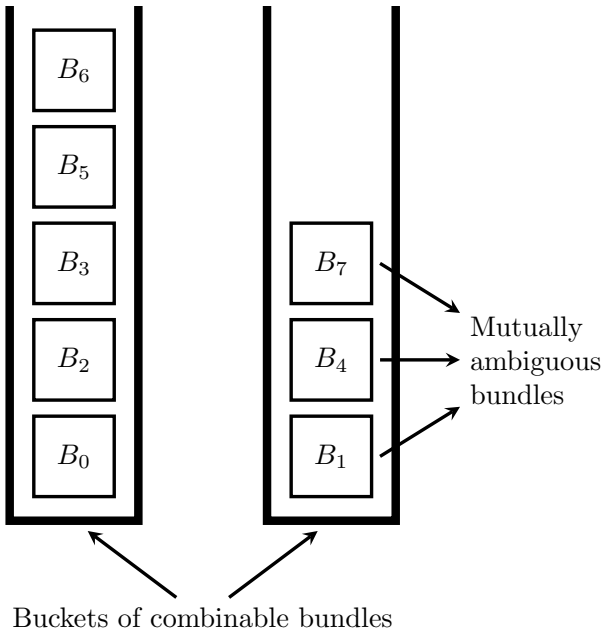


Fig. 7.18: Potential correspondences from an example in Figure 7.17b are sorted into collision-less bundles and two buckets representing the two major directions of the line segments present.

ambiguity and signs a valid combination.

To separate the bundles appropriately, a set of buckets is used in accordance to Figure 7.18. Each bucket contains only the bundles, which would collide with each other. A reliable and unambiguous solutions are then obtained by combining the single bundles from at least two buckets. If more buckets are available, a bundle from each of them can be used, but always one at most. Combining of two bundles should always result into a valid combination, while in case of three or more, the ambiguity can rise a lot. Such situation obviously means collision of correspondences and the solution becomes invalid. Each combination of bundles should be tested if it satisfies the threshold limits.

The example in Figure 7.18 refers to the situation depicted in Figure 7.17b. Careful examination of the correspondences that there are five positions in the north-west direction and three in the north-east, where at least some of them overlap reasonably well. The algorithm have successfully identified this fact and five bundles of correspondences appear in one bucket and three in the second one. Each of the fifteen possible combinations represents a valid combination. The optimal transformation, reliability and ambiguity are provided as well as a by-product of the correspondence extraction process.

The example discussed is somewhat simple in the fact, that there are only two major directions of the line segments. In a more complicated case, more buckets would have appeared, possibly containing less bundles. Complexity of this stage is highly dependant on this distribution. The important thing is, that the reliability threshold should be low enough to enable safe separation of bundles.

7.4.2 Similarity of vector maps with common view-pose

The previous subsection described the first three steps of the registration algorithm, which are applicable on any arbitrary set of line segments, without any additional structure or rules. This means, that every reliable and unambiguous combination of bundles results in a transformation, which makes the related corresponding line segments overlap with no additional constrains being put on them, or on the unmatched remainders in the input sets.

On the other hand, the nature of laser scanning and robotic mapping brings specific rules, which can be used to further narrow the range of solutions extracted so far. First, the objects from the real world always have certain volume and a continuous surface, which has an outer observable side. Even if one of the dimensions of the object is negligible (sheet of paper), it is better to treat it as a real 3D body and avoid the nasty topological issues arising around a bare curved plane in 3D

space[4]. Second, objects from the real world cannot intersect and exist "both at one place" and their surfaces cannot to do so as well. Collisions and embedding objects into each other is clearly possible, but there is always either a distinguishable touch of their surfaces or the surfaces blend together and their parts, which ended up inside the new object just disappear. Third rule is quite simple and emphasises the fact, that a range measurement providing a distance to the closest object also implies, that there is no other observable thing between the sensor and that object. Although this work is focused on 2D problems, the discussion provided mainly 3D examples in an effort to present the illustrations close to our everyday experience. All of those rules apply in the 2D world as well.

The *Expected view test* from the registration scheme in Figure 7.15 is built on these assumptions and provides further criterion metric to distinguish the real correspondences from the practically impossible ones. The test starts with the static line segment set and the optimal transformation, obtained from the correspondences extracted in the previous step. A virtual observer is placed into the static set at the position given by the transformation. A simulated observation is then acquired to obtain an expected view of the known environment from the new location. During this operation several hazardous situations can appear. Many obstacles, one after another, can be present in a single direction, so obviously the closest one should clip the view of those farer from the virtual observer. Second, incomplete measurements can be present in the set, violating the rule of the closed continuous surface. If the line segment is visible from the wrong side, it serves only as an obstacle and should not appear in the expected view. Finally, it is not possible to make any measurements through an unexplored area, since there can be obstacles, so it is treated as if it was fully occupied. On the other hand, unexplored area neither appears in the expected view.

An illustration of the construction of an expected view is shown in Figure 7.19. The static set (all line segments) was obtained from the lower left location. The close obstacle induces a large unknown area behind it, so even though the virtual observer is placed to a position, from which it could "see" behind it, the unknown area still obstructs its view and the known region in the upper left part of the picture cannot be added to the expected view. The only valid observation is demarcated by the green color.

[4]Aside from the inconsistency, that some surfaces would be visible from both sides, while the others belonging to 3D bodies would be observable only from one side, there are other problems ready to arise. The Möbius strip is a good example, as it is *unorientable*, i.e. an inner and outer sides cannot be distinguished, which has a great potential to confuse many mapping algorithms. In the real world, any Möbius strip constructed will have some "thickness". As such, it becomes to be homeomorfic (continuously deformable) to a torus, which is an easily describable 3D body and so will be the strip.

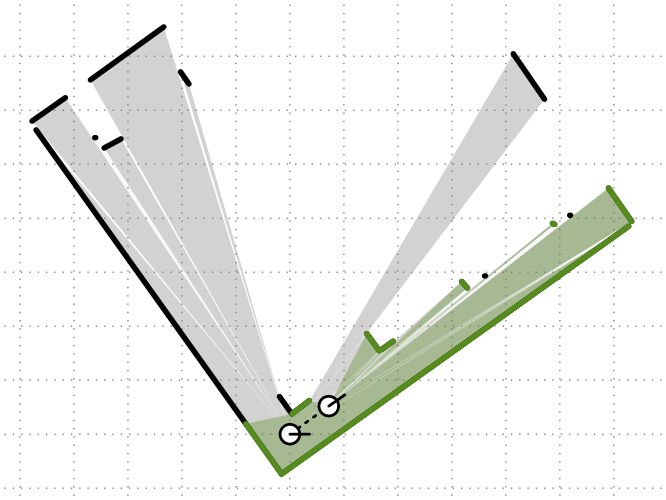
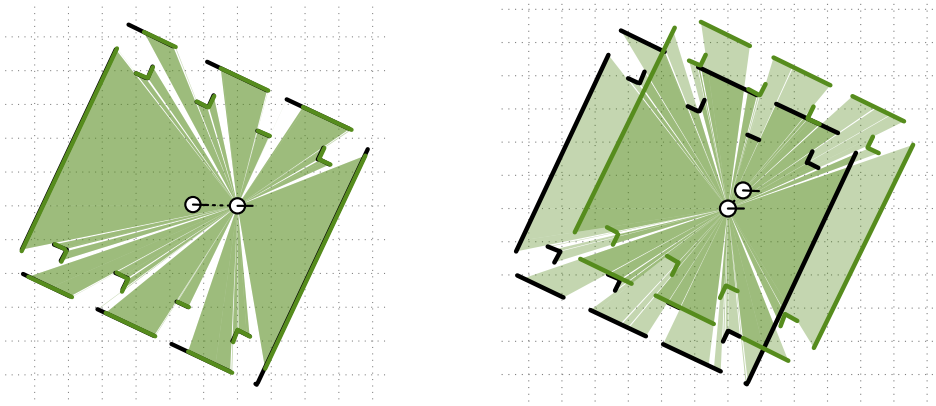


Fig. 7.19: Generation of an expected view for discrepancy evaluation. All line segments (black + green) represent the static set, on which the test is performed. Green segments demarcate the parts of the set visible from the second position (top right mark). Corresponding colouring is used for the original and a new fields of view.

Once the expected view is obtained, the algorithm proceeds to the second stage and compares it with the dynamic set registered by the optimal transformation. At first, the angular intervals, where fields of view of both observation overlap, are extracted. Then, for each of these intervals, the triangular areas denoted by the observer's position and the line segments from both sets are computed. The discrepancy error metric is given by the sum of differences of these two areas for all intervals identified. The lower the discrepancy is, the better the sets fit to each other.

Figure 7.20 shows two examples of the final evaluation of the solutions coming from the model example in Figure 7.16. If the correspondences are properly established (as is the case in Fig. 7.20a), both sets will overlap and the error area is very small. Bad correspondences cause violation of the rules formulated at the beginning of this subsection and result in high error area as can be seen in Figure 7.20b.

The discrepancy metric clearly distinguishes the well registered scans from the false ones. The method works reliably for static environment, which is somewhat limiting. SLAM in a dynamic environment would require identification of moving objects in the scene and application of the described method only on the static part of the data. Computation complexity of the procedure is highly dependant on the algorithm used for the expected view extraction, which refers to occlusion culling frequently solved in the field computer graphics. On the other side, mathematical formulation of the problem consists of computations of line-line intersections and areas of the triangles, which is very straightforward.



(a) Correct correspondences, low discrepacy.

(b) Wrong correspondences, high discrepacy.

Fig. 7.20: Discrepancy visualization for correct and wrong sets of correspondences. Black segments demarcate the dynamic set being registered and the green line segments correspond to the expected view generated from the static set. The darker green area is an unobstructed space visible from both positions and the light green area is visible only in one set and corresponds to the discrepancy error metric.

7.4.3 Preliminary experiments on the correspondence search success ratio

The algorithm in its current state is most probably conceptually finished, but the internal heuristics for selection of the compatible line segment pairs will be tweaked in the subsequent development, mainly to speed up the the whole process. For evaluation of the results during the development, a set of experiments in a virtual environment was performed. Using the same data allows to exactly compare the results after any change and gradually improve the functionality. All reports from the experiments reflecting the actual state of the algorithm are available on the accompanying CD due to their excessive length. The Appendices A, B, C and D are the representative selection for illustrational purposes in this summary. Two types of reports are used: *Case report* for the particular two line segment sets being fitted and the *overview report* summarizing accepted solutions from the particular cases for the whole data set.

Reports A and B come from the *pillars* environment depicted in the respective overview report. It was designed to challenge the registration algorithm, while the robot has significantly obstructed view. The four walls around form a reliable reference and the eight pillars act as the obstacles to be mapped. The trajectory is intentionally designed to lead behind the pillars, so that the surrounding walls

are only partially observable. The case report A nicely shows the importance of the additional discrepancy metric, because all solutions presented satisfy the "high reliability, low ambiguity" condition, but only the confrontation with the real-world constraints allows to find a true set of correspondences. The table of results in the overview report B shows mostly very satisfying statistics. There are basically only two erroneous situations, which can be easily explained with the help of particular case report. First, there are some lines with missing results, which is caused by the discrepancy threshold. The case reports always contain a solution with a low discrepancy, only slightly exceeding the threshold. Smarter way of the optimal solution extraction than a simple thresholding should be probably considered. The second issue arises in the solutions, where the expected view was obtained from a position in the static scan, from which nothing could be observed. It can be easily identified by the zero discrepancy, but this is not a systematic solution and it should be addressed in the algorithm already. Otherwise the results are very good and aside from these two problems, the algorithm looks promising.

The second environment called *quadratic* was used for another set of experiments illustrated by the reports C and D. It was designed to find out, how much detailed pattern is the registration algorithm able to deal with. The quadratic slope of the stairs, makes the steps gradually better and better distinguishable, while the remaining three walls provide a strong reference frame. The case report C nicely shows, how bad can be an influence of small line segments with deviations under the level of noise, if they enter the registration process. Wrong associations cannot be detected in this case, as can be seen in the first (#0) solution. The optimal transformation is not affected too much by this problem, but the correspondences extracted can easily become very misleading. The overall report D shows, that the algorithm was able to find a reasonable transformation for each registration task, but statistics of correspondence search success ration is not very convincing. The lesson learned from this experiment is clear. The maximal allowed ambiguity roughly corresponds to the level of noise present in the original data and the line segments which are too inaccurate with respect to this number, should be rejected from the registration process. Another option would be introduction of a new metric evaluating reliability of the the particular correspondence.

The two experiments presented should be enough to illustrate the outcomes of the algorithm in its current state of development. More tests were performed and can be found (with a brief description) on the attached CD with electronic documents. Though the algorithm gives promising results, there are still some issues to be addressed, before it will be prepared to successfully challenge the state of the art methods.

7.5 Summary of the work on vector map registration

This chapter is dedicated to similarity evaluation and registration of the vectorized laser scans. Several methods were presented for various tasks and evaluated to reveal their true functionality and usability in practice. This section summarizes the most important results.

Section 7.2 presents a novel area-based line segment similarity criterion. Contrary to the usual alternatives, the criterion function is fully differentiable and the derivatives are continuous in the whole domain of definition. The criterion is designed to give zero output for any two line segments lying on the same line, which makes it well applicable, wherever a small vector image is to be fitted into a larger one (e.g. SLAM in robotics). On the other hand, correspondences between line segments must be established in advance. The criterion also supports precomputation. Many algorithms iteratively transform a line segment set by a rigid transformation and compare it to a static set. Precomputation reduces computational complexity of such 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. Testing procedure was selected to correspond with other publications to provide the reader with consistent information.

Section 7.3 describes an analytical approach to the vectorized scan registration with the known correspondences. The optimal transformation, which leads to the best registration with given criterion, is computed in a single step. The method also provides a metric for evaluation of a *reliability* of the computation, since not all sets of line segments are possible to be exactly and unfailingly registered. Second metric is the *ambiguity*, which corresponds to the internal criterion function and reflects, how many compromises were necessary to align the corresponding line segment pairs. Similar to the previous method, all of these values can be obtained from the precomputed sums, which means constant time evaluation and recomputation, 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 was added, which stems from the physical constraints of the real-world measurement. The new metric is called *discrepancy* and it is the last parameter tested during the decision process, which either accepts or declines given set of correspondences as a likely solution. The decision process as described before

is a well working attempt in the direction of a robust data association and the experiments with the synthetic data give a very promising results. Nevertheless, further testing and refinements will be definitely carried out and it is likely, that the algorithm will be further improved.

8 CONCLUSION AND FUTURE WORK

This thesis is focused on processing of raw point clouds for further utilization in SLAM in robotics. Preparation of the data, from filtration and segmentation up to the association with the data already known, is usually referred to as the SLAM front-end and currently seems to be a larger research challenge than the probabilistic inference mechanism for the actual fusion of all measurements into a single, coherent map. Demands put on such maps are growing and simple solutions, distinguishing only the free space - obstacle difference, do not satisfy the needs of modern artificial intelligence. The brief summary of the state of the art in Chapter 2 revealed a strong tendency towards using semantic maps that contain various information about the objects, including their physical dimensions [41]. Since 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 iterative closest point method [151], seem to utilize approximations as well [154], [175]. 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 a straightforward preparation for further algorithms, but, as the approach is quite different from methods found during the literature review, it can be considered novel. Both filtration and segmentation of a raw point cloud is performed at the same time and the output is composed of a set of clusters of points, which can potentially form a continuous edge in the real world. This method is robust, its execution time low and the parameters of the algorithm directly reflect the physical essence of the measurement.

Chapter 6 is dedicated to vectorization of continuous clusters using the total least squares method. In contrast to the point eliminating methods, TLS is able to utilize information from all points and gives significantly more precise results at cost of higher computational time. Contribution of this chapter is twofold. First, an optimized algorithm (FTLS) is introduced, which significantly reduced computational time of the traditional incremental TLS algorithm and now it probably represents the fastest available way of computation of this kind of approximation. Especially for large point clouds (which will probably get even larger as the measurement devices will advance) it proves nearly as fast as the point eliminating methods, while keeping the supreme quality of TLS results. Second contribution lies in an augmentation of this algorithm (AFTLS) addressing the threshold related error of approximation, which is common for de facto 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 already published in [91], [92] and [93].

Registration and data association is covered in Chapter 7. Firstly, similarity of the line segment pairs is discussed. It introduces a novel criterion function, that provides several useful features such as differentiability and continuity in the whole domain of definition. It is also designed to meet requirements encountered during scan to map registration in SLAM applications. The method supports precomputation, which means that if a set of line segment pairs is being examined, addition or removal of a single pair of them is performed in constant time, regardless of the number of pairs totally involved in the computation. The result of the criterion can be also transformed in constant time, providing the similarity enumeration, as if one set of the input line segments would have been transformed. The time needed to examine similarity of two sets of line segments in different mutual pose is therefore constant and not proportional to the number of pairs involved. The criterion was published in [149]

Chapter 7 also presents a method for registration of two sets of line segments with known correspondences, which can be performed in a single step. Apart from the optimal transformation, the computation process provides reliability and ambiguity metrics, which refer to stability of the solution and amount of compromises needed to find the result. Based on this theoretical 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 summarized above provide a good base for further research. When looking back, it did not seem possible that line segments in two-dimensional space would provide so many opportunities for research and useful discoveries. 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] K. Čapek, *R.U.R.* Praha: Aventinum, 1920.

[2] I. Asimov, *I, Robot.* New York, USA: Gnome Press, 1950.

[3] 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*, pp. 116–125, 1985.

[4] 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*, vol. 3, pp. 1615–1621, 1986.

[5] B. Kuipers and Y. T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and Autonomous Systems*, vol. 8, no. 1-2, pp. 47–63, 1991.

[6] J. Yin, L. Carlone, S. Rosa, and B. Bona, "Graph-based robust localization and mapping for autonomous mobile robotic navigation," in *2014 IEEE International Conference on Mechatronics and Automation*, pp. 1680–1685, IEEE, aug 2014.

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

[8] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, p. 35, 1960.

[9] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2003.

[10] Z. Wang, S. Huang, and G. Dissanayake, "Tradeoffs in SLAM with sparse information filters," *Springer Tracts in Advanced Robotics*, vol. 42, pp. 339–348, 2008.

[11] G. Smith, S. Schmidt, and L. McGee, "Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle," tech. rep., Ames Research Center, 1962.

[12] P. Maybeck, *Stochastic Models, Estimation, and Control, Volume1.* Academic Press, 1979.

[13] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," *Int Symp AerospaceDefense Sensing Simul and Controls*, vol. 3, p. 182, 1997.

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* The MIT Press, 2005.

[15] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, no. 91, pp. 1442–1447, IEEE, 1991.

[16] J.-l. Blanco, "Derivation and Implementation of a Full 6D EKF-based Solution to Bearing-Range SLAM," 2008.

[17] S. Holmes, G. Klein, and D. Murray, "An O(N^2) Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1251–1263, 2009.

[18] A. Dempster, N. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society Series B Methodological*, vol. 39, no. 1, pp. 1–38, 1977.

[19] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[20] S. Thrun and M. Montemerlo, "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[21] M. Bosse, "Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework," *The International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[22] S. J. Davey, "Simultaneous Localization and Map Building Using the Probabilistic Multi-Hypothesis Tracker," *IEEE Transactions on Robotics*, vol. 23, pp. 271–280, apr 2007.

[23] P. Del Moral, "Nonlinear Filtering: Interacting Particle Resolution," *Markov Processes and Related Fields*, vol. 2, no. 4, pp. 555–580, 1996.

[24] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks," *IET Signal Processing*, vol. 2, p. 169, jan 2013.

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, vol. 68, pp. 593–598, 2002.

[26] M. Montemerlo, S. Thrun, D. Roller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1151–1156, 2003.

[27] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the FastSLAM algorithm," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, vol. 2006, pp. 424–429, IEEE, 2006.

[28] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, feb 2007.

[29] J. Civera, A. Davison, and J. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, pp. 932–945, oct 2008.

[30] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, pp. 609–631, may 2010.

[31] Shoudong Huang, Zhan Wang, and G. Dissanayake, "Sparse Local Submap Joining Filter for Building Large-Scale Maps," *IEEE Transactions on Robotics*, vol. 24, pp. 1121–1130, oct 2008.

[32] C. Roussillon, A. Gonzalez, J. Solà, J.-M. Codol, N. Mansard, S. Lacroix, and M. Devy, "RT-SLAM: A Generic and Real-Time Visual SLAM Implementation," pp. 31–40, 2011.

[33] M. Mirkhani, R. Forsati, A. M. Shahri, and A. Moayedikia, "A novel efficient algorithm for mobile robot localization," *Robotics and Autonomous Systems*, vol. 61, pp. 920–931, sep 2013.

[34] Y. Li, S. Li, and Y. Ge, "A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments," *Neurocomputing*, vol. 104, pp. 170–179, mar 2013.

[35] S. Thrun, "Robotic Mapping: A Survey," in *Exploring artificial intelligence in the new millennium*, pp. 1–35, 2003.

[36] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 99–110, jun 2006.

[37] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 108–117, sep 2006.

[38] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The SLAM problem: A survey," *Frontiers in Artificial Intelligence and Applications*, vol. 184, no. 1, pp. 363–371, 2008.

[39] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe, "A Review of Recent Developments in Simultaneous Localization and Mapping," *6th International Conference on Industrial and Information Systems*, pp. 477–482, 2011.

[40] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System," *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*, 2013.

[41] 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*, vol. 32, pp. 1309–1332, dec 2016.

[42] 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*, vol. 17, pp. 229–241, jun 2001.

[43] D.-H. Kim and J.-H. Kim, "Visual Loop-Closure Detection Method Using Average Feature Descriptors," in *Robot Intelligence Technology and Applications 2*, pp. 113–118, 2014.

[44] J. Sprickerhof and A. Nüchter, "A heuristic loop closing technique for large-scale 6d slam," *Automatika: Journal for Control, Measurement, Electronics, Computing & Communications*, vol. 52, no. 3, pp. 199–222, 2011.

[45] C. Bibby and I. Reid, *Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association*. 2007.

[46] F. Dayoub, G. Cielniak, and T. Duckett, "Long-term experiments with an adaptive spherical view representation for navigation in changing environments," *Robotics and Autonomous Systems*, vol. 59, pp. 285–295, may 2011.

[47] T. Krajnik, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, "Long-term topological localisation for service robots in dynamic environments using spectral maps," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4537–4542, IEEE, sep 2014.

[48] E. W. Nettleton, P. W. Gibbens, and H. F. Durrant-Whyte, "Closed form solutions to the multiple platform simultaneous localisation and map building (SLAM) problem," *Sensor Fusion: Architectures, Algorithms, and Applications IV*, pp. 428–437, 2000.

[49] C. Leung, S. Huang, and G. Dissanayake, "Active SLAM using Model Predictive Control and Attractor based Exploration," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5031, IEEE, oct 2006.

[50] R. Menzel, "The honeybee as a model for understanding the basis of cognition," *Nature Reviews Neuroscience*, vol. 13, pp. 758–768, oct 2012.

[51] A. B. Watson, "A formula for human retinal ganglion cell receptive field density as a function of visual field location," *Journal of Vision*, vol. 14, pp. 15–15, jun 2014.

[52] C. A. Curcio, K. R. Sloan, R. E. Kalina, and A. E. Hendrickson, "Human photoreceptor topography," *The Journal of Comparative Neurology*, vol. 292, pp. 497–523, feb 1990.

[53] R. Masland, "The Neuronal Organization of the Retina," *Neuron*, vol. 76, pp. 266–280, oct 2012.

[54] K. von Frisch, *The Dance Language and Orientation of Bees*. Massachusetts, USA: Belknap Press, 1967.

[55] S. Zhang, "Maze Navigation by Honeybees: Learning Path Regularity," *Learning & Memory*, vol. 7, pp. 363–374, nov 2000.

[56] G. K. Aguirre, E. Zarahn, and M. D'Esposito, "Neural components of topographical representation," *Proceedings of the National Academy of Sciences*, vol. 95, pp. 839–846, feb 1998.

[57] N. Burgess, E. A. Maguire, and J. O'Keefe, "The Human Hippocampus and Spatial and Episodic Memory," *Neuron*, vol. 35, pp. 625–641, aug 2002.

[58] M. M. Chun and Y. Jiang, "Contextual Cueing: Implicit Learning and Memory of Visual Context Guides Spatial Attention," *Cognitive Psychology*, vol. 36, pp. 28–71, jun 1998.

[59] E. L. Newman, J. B. Caplan, M. P. Kirschen, I. O. Korolev, R. Sekuler, and M. J. Kahana, "Learning your way around town: How virtual taxicab drivers learn to use both layout and landmark information," *Cognition*, vol. 104, pp. 231–253, aug 2007.

[60] J. J. Bos, M. Vinck, L. A. van Mourik-Donga, J. C. Jackson, M. P. Witter, and C. M. A. Pennartz, "Perirhinal firing patterns are sustained across large spatial segments of the task environment," *Nature Communications*, vol. 8, p. 15602, may 2017.

[61] F. Cutzu and S. Edelman, "Representation of object similarity in human vision: psychophysics and a computational model," *Vision Research*, vol. 38, pp. 2229–2257, aug 1998.

[62] N. Kriegeskorte and R. A. Kievit, "Representational geometry: integrating cognition, computation, and the brain," *Trends in Cognitive Sciences*, vol. 17, pp. 401–412, aug 2013.

[63] Y. Zhang, Y. Wang, H. Wang, L. Cui, S. Tian, and D. Wang, "Different processes are involved in human brain for shape and face comparisons," *Neuroscience Letters*, vol. 303, pp. 157–160, may 2001.

[64] R. M. Cichy, D. Pantazis, and A. Oliva, "Resolving human object recognition in space and time," *Nature Neuroscience*, vol. 17, pp. 455–462, jan 2014.

[65] S. G. Tzafestas, *Introduction to Mobile Robot Control*. Elsevier, 1 ed., 2014.

[66] K.-L. Han, H. Kim, and J. S. Lee, "The sources of position errors of omnidirectional mobile robot with Mecanum wheel," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, pp. 581–586, IEEE, oct 2010.

[67] L.-C. Lin and H.-Y. Shih, "Modeling and Adaptive Control of an Omni-Mecanum-Wheeled Robot," *Intelligent Control and Automation*, vol. 04, no. 02, pp. 166–179, 2013.

[68] P. Oliveira, A. J. Sousa, a. P. Moreira, and P. J. Costa, "Precise Modeling of a Four Wheeled Omni-directional Robot," in *Proceedings of the 8th Conference on Autonomous Robot Systems and Competitions*, pp. 57–62, 2008.

[69] J. Wu, R. L. Williams, and J. Lew, "Velocity and Acceleration Cones for Kinematic and Dynamic Constraints on Omni-Directional Mobile Robots," 2006.

[70] I. Doroftei, V. Grosu, and V. Spinu, "Omnidirectional Mobile Robot – Design and Implementation," in *Bioinspiration and Robotics: Walking and Climbing Robots*, no. September, ch. 29, 2007.

[71] P. Gabrlik, A. Jelinek, and P. Janata, "Precise Multi-Sensor Georeferencing System for Micro UAVs," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 170–175, 2016.

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

[73] E. Grilli, F. Menna, and F. Remondino, "A Review of Point Clouds Segmentation and Classification Algorithms," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W3, no. March, pp. 339–344, 2017.

[74] G. Vosselman, B. Gorte, G. Sithole, and T. Rabbani, "Recognising Structure in Laser Scanner Point Clouds," *Information Sciences*, vol. 46, no. April 2016, pp. 1–6, 2004.

[75] T. Rabbani, F. a. van den Heuvel, and G. Vosselman, "Segmentation of point clouds using smoothness constraint," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences - Commission V Symposium 'Image Engineering and Vision Metrology'*, vol. 36, no. 5, pp. 248–253, 2006.

[76] M. Awrangjeb and C. S. Fraser, "Rule-based segmentation of LIDAR point cloud for automatic extraction of building roof planes," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-3/W3, pp. 1–6, oct 2013.

[77] B. Parvin and G. G. Medioni, "Segmentation of range images into planar surfaces by split and merge," in *CVPR '86*, pp. 415–417, 1986.

[78] Q. Zhan, L. Yubin, and Y. Xiao, "Color-Based Segmentation of Point Clouds," *Laser scanning 2009, IAPRS*, vol. XXXVIII, P, pp. 248–252, 2009.

[79] J. Macqueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 233, pp. 281–297, 1967.

[80] N. Chehata, N. David, and F. Bretar, "LIDAR Data Classification using Hierarchical K-means clustering," *ISPRS Congress Beijing 2008*, vol. 37, no. B3b, pp. 325–330, 2008.

[81] B.-Q. Shi, J. Liang, and Q. Liu, "Adaptive simplification of point cloud using -means clustering," *Computer-Aided Design*, vol. 43, pp. 910–922, aug 2011.

[82] K. Zhang, W. Bi, X. Zhang, X. Fu, K. Zhou, and L. Zhu, "A New Kmeans Clustering Algorithm For Point Cloud," *International Journal of Hybrid Information Technology*, vol. 8, pp. 157–170, sep 2015.

[83] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6218–6225, IEEE, may 2014.

[84] D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo, "Fast and Robust Edge Extraction in Unorganized Point Clouds," in *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–8, IEEE, nov 2015.

[85] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D LIDAR point clouds," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2798–2805, IEEE, may 2011.

[86] M. Liu and R. Siegwart, "Information theory based validation for point-cloud segmentation aided by tensor voting," in *2013 IEEE International Conference on Information and Automation (ICIA)*, no. August, pp. 168–173, IEEE, aug 2013.

[87] A. Dimitrov, R. Gu, and M. Golparvar-Fard, "Non-Uniform B-Spline Surface Fitting from Unordered 3D Point Clouds for As-Built Modeling," *Computer-Aided Civil and Infrastructure Engineering*, vol. 31, no. 7, pp. 483–498, 2016.

[88] T. Wu, H. Cui, Y. Li, W. Wang, D. Lui, and E. Shang, "A feature matching and fusion-based positive obstacle detection algorithm for field autonomous land vehicles," *International Journal of Advanced Robotic Systems*, vol. 14, p. 172988141769251, apr 2017.

[89] X. Hu, X. Li, and Y. Zhang, "Fast Filtering of LiDAR Point Cloud in Urban Areas Based on Scan Line Segmentation and GPU Acceleration," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, pp. 308–312, mar 2013.

[90] S. P. Baker and R. W. Sadowski, "GPU assisted processing of point cloud data sets for ground segmentation in autonomous vehicles," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pp. 1–6, IEEE, apr 2013.

[91] A. Jelinek, "Practical Aspects of Total Least Squares Vectorization of Point Clouds in Mobile Robotics," *IFAC-PapersOnLine*, vol. 48, no. 4, pp. 193–198, 2015.

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

[93] 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*, pp. 216–223, SCITEPRESS - Science and and Technology Publications, 2016.

[94] Z. Lu, S. Baek, and S. Lee, "Robust 3D line extraction from stereo point clouds," *2008 IEEE International Conference on Robotics, Automation and Mechatronics, RAM 2008*, vol. 00, pp. 1–5, sep 2008.

[95] K. Hirose and H. Saito, "Fast Line Description for Line-based SLAM," in *Procedings of the British Machine Vision Conference 2012*, pp. 83.1–83.11, British Machine Vision Association, 2012.

[96] 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*, vol. 23, pp. 97–111, jun 2007.

[97] N. Pears, "Feature extraction and tracking for scanning range sensors," *Robotics and Autonomous Systems*, vol. 33, pp. 43–58, oct 2000.

[98] W. Shi and C. Cheung, "Performance Evaluation of Line Simplification Algorithms for Vector Generalization," *The Cartographic Journal*, vol. 43, pp. 27–44, mar 2006.

[99] J. Liu, J. Zhang, F. Xu, Z. Huang, and Y. Li, "Adaptive Algorithm for Automated Polygonal Approximation of High Spatial Resolution Remote Sensing Imagery Segmentation Contours," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 52, pp. 1099–1106, feb 2014.

[100] Jiaping Zhao, Suya You, Jing Huang, J. Zhao, S. You, J. Huang, Jiaping Zhao, Suya You, and Jing Huang, "Rapid extraction and updating of road network from airborne LiDAR data," in *2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, no. OCTOBER 2011, pp. 1–7, IEEE, oct 2011.

[101] C. Dyken, M. Dæhlen, and T. Sevaldrud, "Simultaneous curve simplification," *Journal of Geographical Systems*, vol. 11, pp. 273–289, sep 2009.

[102] P. Kandal and S. Karschti, "Method for simplified storage of data representing forms," 2014.

[103] R. Lange, F. Dürr, and K. Rothermel, "Efficient real-time trajectory tracking," *The VLDB Journal*, vol. 20, pp. 671–694, oct 2011.

[104] I. Sandu Popa, K. Zeitouni, V. Oria, and A. Kharrat, "Spatio-temporal compression of trajectories in road networks," *GeoInformatica*, vol. 19, pp. 117–145, jan 2014.

[105] M. Werner, L. Schauer, and A. Scharf, "Reliable trajectory classification using Wi-Fi signal strength in indoor scenarios," in *2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014*, pp. 663–670, IEEE, may 2014.

[106] A. Thiebault and Y. Tremblay, "Splitting animal trajectories into fine-scale behaviorally consistent movement units: breaking points relate to external stimuli in a foraging seabird," *Behavioral Ecology and Sociobiology*, vol. 67, pp. 1013–1026, jun 2013.

[107] M. Romadi, R. Oulah Haj Thami, R. Romadi, and R. Chiheb, "Detection and recognition of road signs in a video stream based on the shape of the panels," in *2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14)*, pp. 1–5, IEEE, may 2014.

[108] G. Danuser and M. Stricker, "Parametric model fitting: from inlier characterization to outlier detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 263–280, mar 1998.

[109] D. Arifoglu, E. Sahin, H. Adiguzel, P. Duygulu, and M. Kalpakli, "Matching Islamic patterns in Kufic images," *Pattern Analysis and Applications*, vol. 18, pp. 601–617, aug 2015.

[110] M. Rizzardi and S. Troisi, "Approximation of irregular polylines by means of a straight-line graph," *Applied Geomatics*, vol. 3, pp. 171–182, sep 2011.

[111] W. Gong, F. Mao, and S. Song, "Signal simplification and cloud detection with an improved Douglas-Peucker algorithm for single-channel lidar," *Meteorology and Atmospheric Physics*, vol. 113, pp. 89–97, jun 2011.

[112] T. Choi, C. Park, H. Do, D. Park, J. Kyung, and G. Chung, "Trajectory correction based on shape peculiarity in direct teaching manipulator," *International Journal of Control, Automation and Systems*, vol. 11, pp. 1009–1017, oct 2013.

[113] P. V. C. Hough, "Method and Means for Recognizing Complex Patterns," 1962.

[114] R. F. C. Guerreiro and P. M. Q. Aguiar, "Connectivity-enforcing Hough transform for the robust extraction of line segments," *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, vol. 21, pp. 4819–29, dec 2012.

[115] K. Ni, N. Armstrong-Crews, and S. Sawyer, "Geo-registering 3D point clouds to 2D maps with scan matching and the Hough Transform," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1864–1868, IEEE, may 2013.

[116] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, jun 1981.

[117] M. Mirmehdi, P. L. Palmer, and J. Kittler, "Robust line segment extraction using genetic algorithms," in *Image Processing and Its Applications, 1997., Sixth International Conference on*, vol. 1, pp. 141 –145 vol.1, IEEE, 1997.

[118] Y. Cai and Q. Guo, "Point set generalization based on the Kohonen Net," *Geo-spatial Information Science*, vol. 11, pp. 221–227, jan 2008.

[119] M. Naouai, M. Narjess, and A. Hamouda, "Line Recognition Algorithm Using Constrained Delaunay Triangulation," in *ELMAR, 2010 PROCEEDINGS*, no. September, pp. 15–17, 2010.

[120] R. F. C. Guerreiro and P. M. Q. Aguiar, "Extraction of line segments in cluttered images via multiscale edges," in *2013 IEEE International Conference on Image Processing*, pp. 3045–3048, IEEE, sep 2013.

[121] L. Wenyin and D. Dori, "From Raster to Vectors: Extracting Visual Information from Line Drawings," *Pattern Analysis & Applications*, vol. 2, pp. 10–21, feb 1999.

[122] E. Altantsetseg, Y. Muraki, K. Matsuyama, and K. Konno, "Feature line extraction from unorganized noisy point clouds using truncated Fourier series," *The Visual Computer*, vol. 29, pp. 617–626, apr 2013.

[123] 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*, vol. 10, pp. 112–122, oct 1973.

[124] A. Saalfeld, "Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm," *Cartography and Geographic Information Science*, vol. 26, pp. 7–18, jan 1999.

[125] J. Ma, S. Xu, Y. Pu, and G. Chen, "A real-time parallel implementation of Douglas-Peucker polyline simplification algorithm on shared memory multi-core processor computers," *ICCASM 2010 - 2010 International Conference on Computer Application and System Modeling, Proceedings*, vol. 4, no. Iccasm, pp. 647–652, 2010.

[126] Z. Zhao and A. Saalfeld, "Linear-time sleeve-fitting polyline simplification algorithms," *Proceedings of AutoCarto*, pp. 214–223, 1997.

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

[128] K. O. Arras and R. Y. Siegwart, "Feature Extraction and Scene Interpretation for Map-Based Navigation and Map Building," in *Proc. SPIE 3210, Mobile Robots XII* (D. W. Gage, ed.), vol. 3210, pp. 42–53, jan 1998.

[129] S. Rippa, "Adaptive Approximation by Piecewise Linear Polynomials on Triangulations of Subsets of Scattered Data," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, pp. 1123–1141, sep 1992.

[130] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*, no. May, (New York, New York, USA), pp. 209–216, ACM Press, 1997.

[131] C. Chen, C. Yan, X. Cao, J. Guo, and H. Dai, "A greedy-based multiquadric method for LiDAR-derived ground data reduction," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 102, pp. 110–121, 2015.

[132] P. Jensfelt, *Approaches to Mobile Robot Localization in Indoor Environments*. Phd thesis, KTH, 2001.

[133] A. Diosi and L. Kleeman, "Uncertainty of Line Segments Extracted from Static SICK PLS Laser Scans," in *Australiasian Conference on Robotics and Automation*, p. 10, 2002.

[134] P. Kocmanova, L. Zalud, and A. Chromy, "3D proximity laser scanner calibration," in *2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR)*, pp. 742–747, IEEE, aug 2013.

[135] L. Zalud, P. Kocmanova, F. Burian, T. Jilek, P. Kalvoda, and L. Kopecny, "Calibration and Evaluation of Parameters in A 3D Proximity Rotating Scanner," *Elektronika ir Elektrotechnika*, vol. 21, pp. 3–12, feb 2015.

[136] R. J. Adcock, "A Problem in Least Squares," *The Analyst*, vol. 5, p. 53, mar 1878.

[137] G. H. Golub and C. F. van Loan, "An Analysis of the Total Least Squares Problem," *SIAM Journal on Numerical Analysis*, vol. 17, pp. 883–893, dec 1980.

[138] E. W. Deming, *Statistical Adjustment of Data*. Mineola, New York: Dover Publications, dover book ed., 2011.

[139] R. Deriche, R. Vaillant, and O. Faugeras, "From Noisy Edges Points to 3D Reconstruction of a Scene : A Robust Approach and Its Uncertainty Analysis," *Series in Machine Perception and Artificial Intelligence*, vol. 2, pp. 71–79, may 1992.

[140] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, pp. 308–313, jan 1965.

[141] J. C. Lagarias, J. a. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions," *SIAM Journal on Optimization*, vol. 9, pp. 112–147, jan 1998.

[142] F. Le Floc'h, "Issues of Nelder-Mead Simplex Optimisation with Constraints," *SSRN Electronic Journal*, pp. 1–7, 2012.

[143] H.-P. Kriegel, E. Schubert, and A. Zimek, "The (black) art of runtime evaluation : Are we comparing algorithms or implementations ?," *Knowledge and Information Systems*, 2016.

[144] T. Bailey, *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. Ph.d., The University of Sydney, 2002.

[145] E. Tsardoulias and L. Petrou, "Critical Rays Scan Match SLAM," *Journal of Intelligent & Robotic Systems*, vol. 72, pp. 441–462, feb 2013.

[146] G. K. L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, F. C. Langbein, Yonghuai Liu, D. Marshall, R. R. Martin, Xian-Fang Sun, and P. L. Rosin, "Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 7, pp. 1199–1217, 2013.

[147] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *International Journal of Computer Vision*, vol. 47, no. 1/3, pp. 7–42, 2002.

[148] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, pp. 977–1000, oct 2003.

[149] 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*, pp. 1–7, 2017.

[150] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[151] 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*, vol. 14, no. 2, pp. 239–256, 1992.

[152] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on. IEEE*, pp. 145–152, 2001.

[153] B. Bellekens, V. Spruyt, R. Berkvens, R. Penne, and M. Weyn, "A Benchmark Survey of Rigid 3D Point Cloud Registration Algorithms," *International Journal on Advances in Intelligent Systems*, vol. 8, no. 1, pp. 118–127, 2015.

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

[155] D. Holz and S. Behnke, "Registration of Non-Uniform Density 3D Point Clouds using Approximate Surface Reconstructione," *Proceedings of the International Symposium on Robotics (ISR) and the German Conference on Robotics (ROBOTIK)*, no. June, pp. 475–481, 2014.

[156] C.-C. Lin, Y.-C. Tai, J.-J. Lee, and Y.-S. Chen, "A novel point cloud registration using 2D image features," *EURASIP Journal on Advances in Signal Processing*, vol. 2017, p. 5, dec 2017.

[157] S. Liu and X. Xie, "Research on algorithm of point cloud MapReduce registration," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pp. 338–341, IEEE, sep 2011.

[158] S. Ji, Y. Ren, Z. Ji, X. Liu, and G. Hong, "An improved method for registration of point cloud," *Optik - International Journal for Light and Electron Optics*, vol. 140, pp. 451–458, 2017.

[159] R. Benjemaa and F. Schmitt, "A solution for the registration of multiple 3D point sets using unit quaternions," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1407, pp. 34–50, 1998.

[160] E. H. Teniente and J. Andrade-Cetto, "Registration of 3D Points Clouds for Urban Robot Mapping," tech. rep., Institut de Robòtica i Informàtica Industrial, 2008.

[161] Bisheng Yang, Zheng Wei, Qingquan Li, and J. Li, "Semiautomated Building Facade Footprint Extraction From Mobile LiDAR Point Clouds," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 4, pp. 766–770, 2013.

[162] J. Li, X. He, and J. Li, "2D LiDAR and camera fusion in 3D modeling of indoor environment," in *2015 National Aerospace and Electronics Conference (NAECON)*, vol. 2016-March, pp. 379–383, IEEE, jun 2015.

[163] K. Ma, F. Lu, and X. Chen, "Robust Planar Surface Extraction from Noisy and Semi-Dense 3D Point Cloud for Augmented Reality," in *2016 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 453–458, IEEE, sep 2016.

[164] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pp. 1150–1157, IEEE, 1999.

[165] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, jun 2008.

[166] J. Li and G. Jiang, "Point clouds registration with surfaces of low curvatures," in *2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1–5, IEEE, sep 2015.

[167] M. Weinmann and B. Jutzi, "Fast and accurate point cloud registration by exploiting inverse cumulative histograms (ICHs)," *Joint Urban Remote Sensing Event 2013, JURSE 2013*, vol. 856, pp. 218–221, 2013.

[168] R. Lemuz-lópez and M. Arias-estrada, "Iterative Closest SIFT Formulation for Robust Feature Matching," in *Advances in Visual Computing*, pp. 502–513, 2006.

[169] Bing Jian and B. Vemuri, "A robust algorithm for point set registration using mixture of Gaussians," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pp. 1246–1251 Vol. 2, IEEE, 2005.

[170] Y. Miao, Y. Liu, H. Ma, and H. Jin, "The pose estimation of mobile robot based on improved point cloud registration," *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, pp. 1–10, 2016.

[171] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[172] M. Alshawa, "ICL, Iterative Closest Line: A novel point cloud registration algorithm based on linear features.," *ISPRS 2nd summer school*, no. 10, pp. 53–59, 2007.

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

[174] Y. P. Kwon, "Line Segment-based Aerial Image Registration," tech. rep., University of California at Berkeley, Berkeley, 2014.

[175] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas, "Registration of point cloud data from a geometric optimization perspective," in *Proceedings of the*

2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing - SGP '04, no. June 2017, (New York, New York, USA), p. 22, ACM Press, 2004.

[176] Y. Li and R. L. Stevenson, "Multimodal Image Registration With Line Segments by Selective Search," *IEEE Transactions on Cybernetics*, pp. 1–14, 2016.

[177] Z. Wang, J. M. Esturo, H. Seidel, and T. Weinkauf, "Pattern Search in Flows based on Similarity of Stream Line Segments Additional Material," in *Vision, Modeling and Visualization*, (Darmstadt, Germany), The Eurographics Association, 2014.

[178] Y. Gao and M. K. Leung, "Line segment Hausdorff distance on face matching," *Pattern Recognition*, vol. 35, pp. 361–371, feb 2002.

[179] Liang Huang, Qing Chang, Shangfeng Chen, and Huadong Dai, "Line segment matching of space target image sequence based on optical flow prediction," in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pp. 148–152, IEEE, dec 2015.

[180] G. Yammine, E. Wige, F. Simmet, D. Niederkorn, and A. Kaup, "Novel Similarity-Invariant Line Descriptor and Matching Algorithm for Global Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, pp. 1323–1335, aug 2014.

[181] K. Z. Haigh and J. R. Shewchuk, "Geometric Similarity Metrics for Case-Based Reasoning," *Case-Based Reasoning: Working Notes from the AAAI-94 Workshop*, pp. 182–187, 1994.

[182] J. Witt and U. Weltin, "Robust stereo visual odometry using iterative closest multiple lines," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4164–4171, IEEE, nov 2013.

[183] W. Xiao-yu, H. Bing, S. Fang, and C. Xi, "3-D object detection based on line sets matching," in *2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pp. 270–275, IEEE, dec 2015.

[184] M. Poreba and F. Goulette, "Line segment based approach for accuracy assessment of MLS point cloud in urban areas," *International Symposium on Mobile Mapping Technology (MMT)*, 2013.

[185] H. Alt, B. Behrends, and J. Blömer, "Approximate matching of polygonal shapes," *Annals of Mathematics and Artificial Intelligence*, vol. 13, pp. 251–265, sep 1995.

[186] X. Yu, M. Leung, and Y. Gao, "Hausdorff distance for shape matching," in *The 4th LASTED International Conference on visualization, image, and image processing*, 2004.

[187] K. Shahbaz, *Applied Similarity Problems Using Frechet Distance*. PhD thesis, Carleton University Ottawa, 2013.

[188] S. Wirtz and D. Paulus, "Evaluation of established line segment distance functions," *Pattern Recognition and Image Analysis*, vol. 26, pp. 354–359, apr 2016.

[189] J. Chen, M. K. Leung, and Y. Gao, "Noisy logo recognition using line segment Hausdorff distance," *Pattern Recognition*, vol. 36, pp. 943–955, apr 2003.

[190] Y. Mori, "Evaluation of Several "Single-Pass" Estimators of the Mean and the Standard Deviation of Wind Direction," *Journal of Climate and Applied Meteorology*, vol. 25, pp. 1387–1397, oct 1986.

[191] United States Environmental Protection Agency, "Meteorological Monitoring Guidance for Regulatory Modeling Applications," *Epa-454/R-99-005*, p. 171, 2000.

[192] L. Kogan, "Circular Values Math and Statistics with C ++ 11," 2012.

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

SLAM	Simultaneous localization and mapping
OG	Occupancy grid
AI	Artificial intelligence
KF	Kalman filter
EM	Expectation maximization
PF	Particle filter
GA	Genetic algorithm
CC	Computational complexity
RANSAC	Random sample consensus method
ATEROS	Autonomous telepresence robotic system developed by our laboratory
HT	Hough transformation
PE	Point eliminating methods of algorithms
DP	Douglas-Peucker algortihm [123]
RW	Reumann-Witkam algorithm [127]
TLS	Total least squares method or algorithms
INC	Incremental implementation of the algorithm for TLS vectorization
FTLS	Fast total least squares algorithm for vectorization (see Section 6.3)
AFTLS	Augmented fast total least squares algorithm (see Section 6.4)
NM	Nelder-Mead method [140]
ICP	Iterative closest point registration method

LIST OF APPENDICES

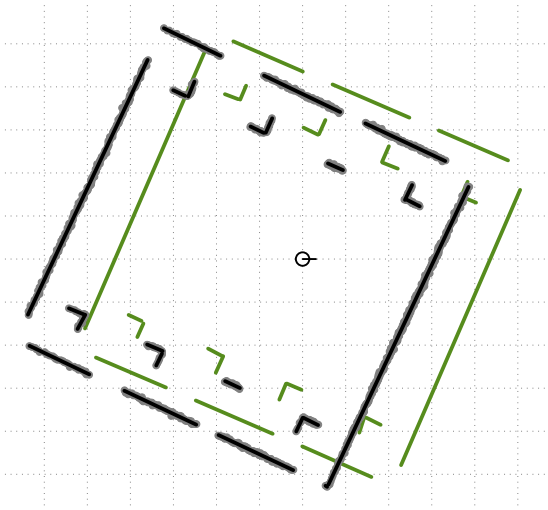
A CORRESPONDENCE SEARCH CASE REPORT - PILLARS DATASET

Dataset:

Environment:	pillars
Noise σ:	1.0 cm
Grid:	50 x 50 cm

Observation:

Scan points :	#0 → #1
True Δx :	67 cm
True Δy :	0 cm
True $\Delta\alpha$:	0.03 rad



Algorithm settings:

Maximal $\|\Delta x\|$:	120 cm
Maximal $\|\Delta y\|$:	120 cm
Maximal $\|\Delta\alpha\|$:	1.26 rad
Vectorization max. error :	3.0 cm
Reliability threshold :	0.10
Ambiguity threshold :	25
Ambiguity k_α:	1000
Ambiguity k_{xy}:	1
Discrepancy threshold:	2000

Description:

The *pillars* environment was designed to challenge the registration algorithm, while the robot has significantly obstructed view. The four walls around form a reliable reference and the eight pillars act as the obstacles to be mapped.

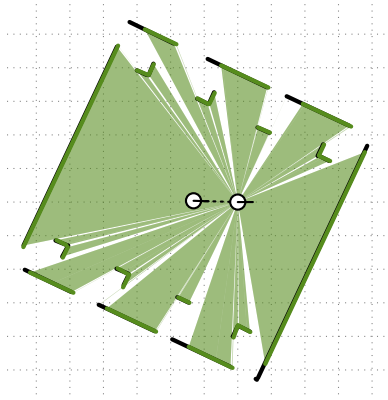
The low level of noise makes the pillars well distinguishable and the vectorization provides an adequate line segment approximations, which can take place in the registration process.

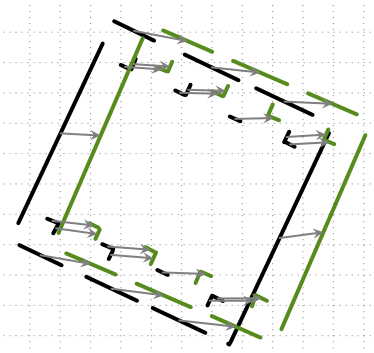
Legend:

Static scan(#0) | Registered scan(#1) | Scanned points

Correspondence | Common valid area | Discrepancy area

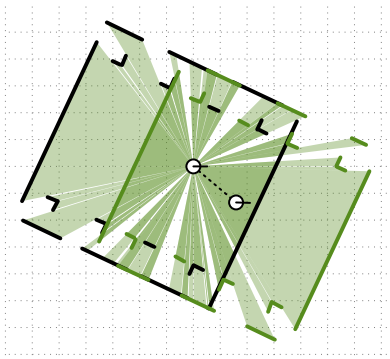
Solutions:

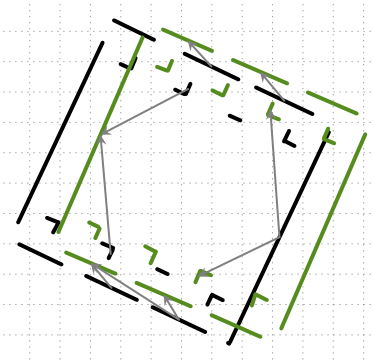




Solution: **# 0**
Reliability: 0.962
Ambiguity: 14.2
Discrepancy: 1128
⇒ **Accepted**

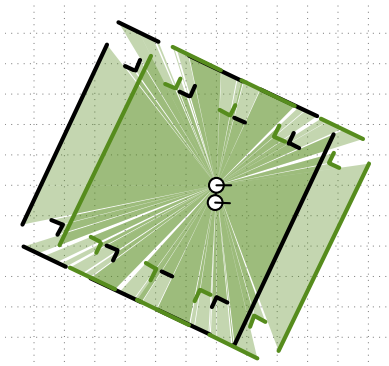
Statistics:
True positive: 22
True negative: 65
False positive: 0
False negative: 0

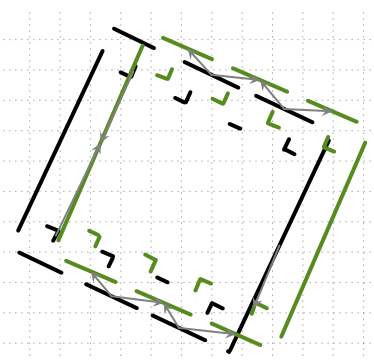




Solution: **# 1**
Reliability: 0.954
Ambiguity: 5.5
Discrepancy: 116222
⇒ **Rejected**

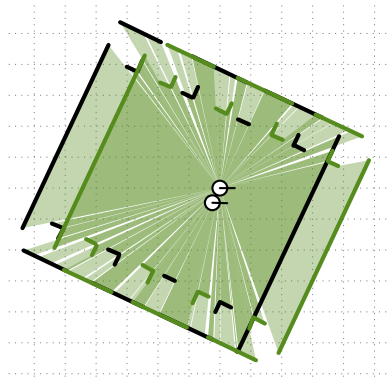
Statistics:
True positive: 0
True negative: 56
False positive: 9
False negative: 22

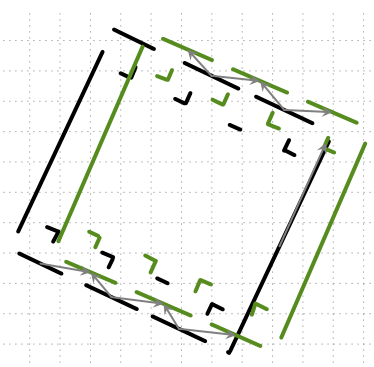




Solution: **# 2**
Reliability: 0.902
Ambiguity: 15.5
Discrepancy: 65803
⇒ **Rejected**

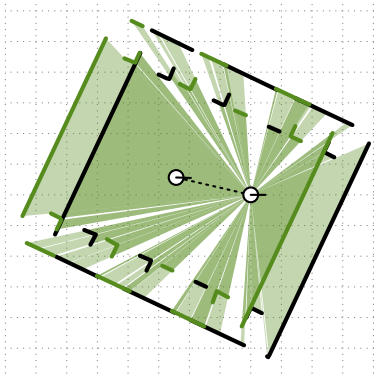
Statistics:
True positive: 4
True negative: 58
False positive: 7
False negative: 18

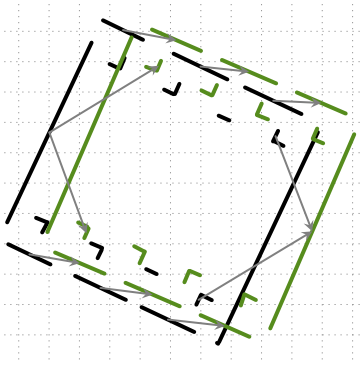




Solution: **# 3**
Reliability: 0.658
Ambiguity: 9.3
Discrepancy: 59411
⇒ **Rejected**

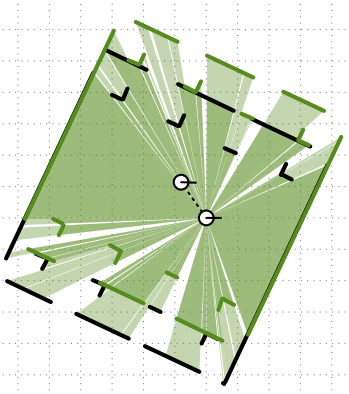
Statistics:
True positive: 5
True negative: 60
False positive: 5
False negative: 17

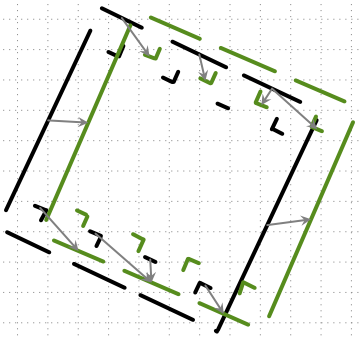




Solution: **# 4**
Reliability: 0.942
Ambiguity: 18.3
Discrepancy: 56556
⇒ **Rejected**

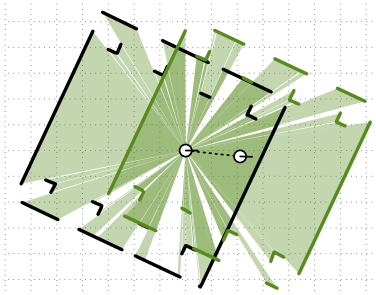
Statistics:
True positive: 6
True negative: 61
False positive: 4
False negative: 16

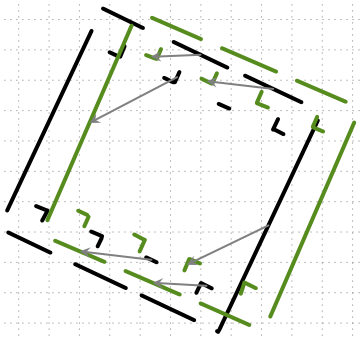




Solution: **# 5**
Reliability: 0.997
Ambiguity: 17.2
Discrepancy: 40565
⇒ **Rejected**

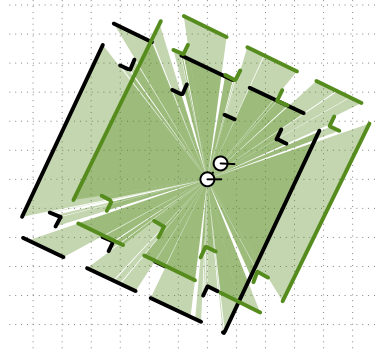
Statistics:
True positive: 2
True negative: 57
False positive: 8
False negative: 20

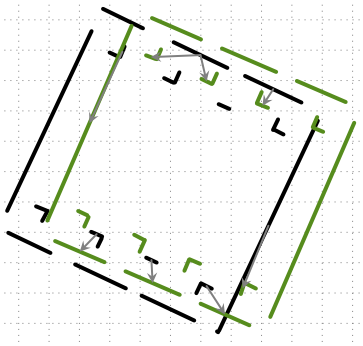




Solution: **# 6**
Reliability: 0.995
Ambiguity: 9.5
Discrepancy: 123021
⇒ **Rejected**

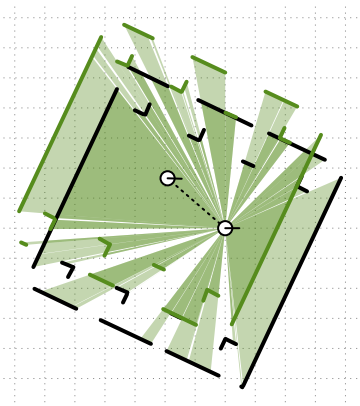
Statistics:
True positive: 0
True negative: 59
False positive: 6
False negative: 22

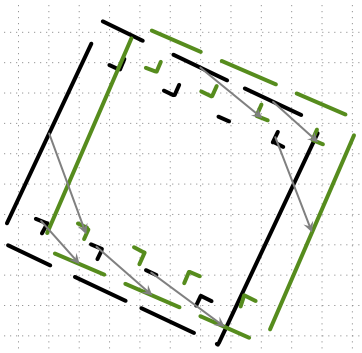




Solution: **# 7**
Reliability: 0.968
Ambiguity: 15.3
Discrepancy: 82684
⇒ **Rejected**

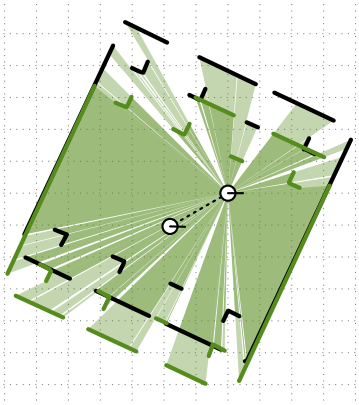
Statistics:
True positive: 0
True negative: 57
False positive: 8
False negative: 22

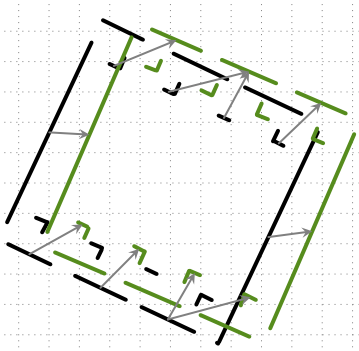




Solution: # 8
Reliability: 0.990
Ambiguity: 23.4
Discrepancy: 65553
⇒ **Rejected**

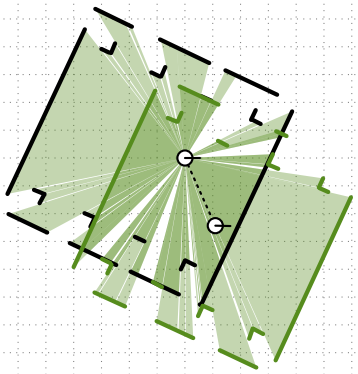
Statistics:
True positive: 0
True negative: 58
False positive: 7
False negative: 22

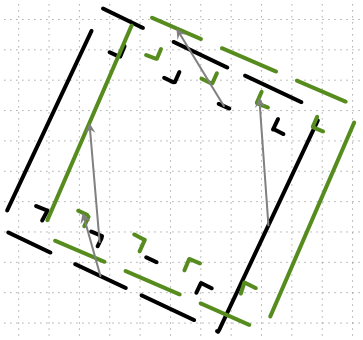




Solution: # 9
Reliability: 0.999
Ambiguity: 12.4
Discrepancy: 37888
⇒ **Rejected**

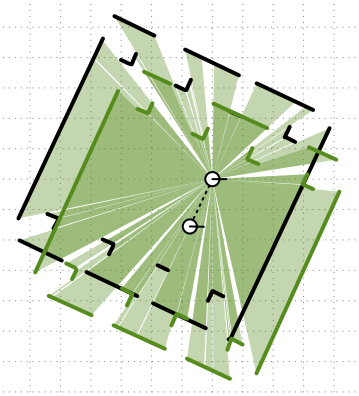
Statistics:
True positive: 2
True negative: 57
False positive: 8
False negative: 20

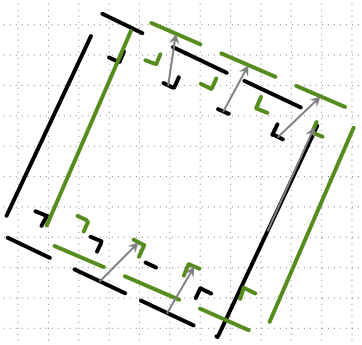




Solution: # 10
Reliability: 1.000
Ambiguity: 8.7
Discrepancy: 124077
⇒ **Rejected**

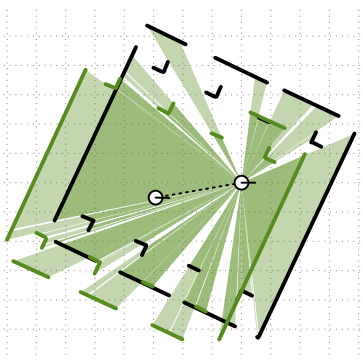
Statistics:
True positive: 0
True negative: 61
False positive: 4
False negative: 22

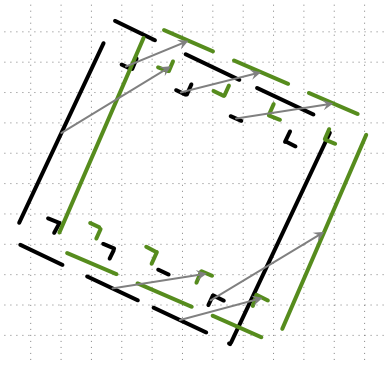




Solution: # 11
Reliability: 0.812
Ambiguity: 6.7
Discrepancy: 72935
⇒ **Rejected**

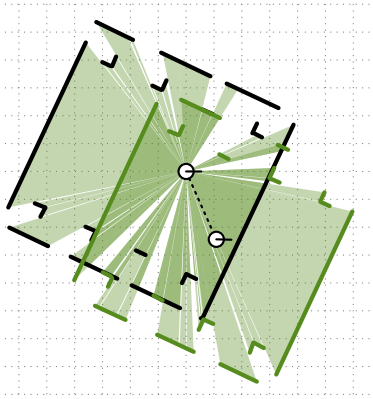
Statistics:
True positive: 0
True negative: 59
False positive: 6
False negative: 22

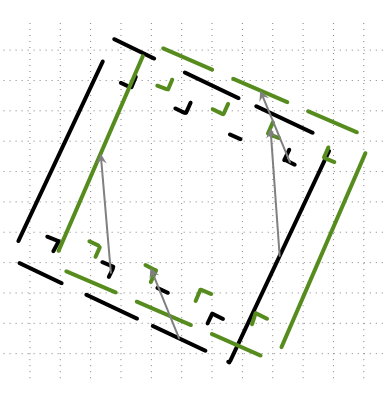




Solution: **# 12**
Reliability: 0.999
Ambiguity: 9.1
Discrepancy: 66354
⇒ **Rejected**

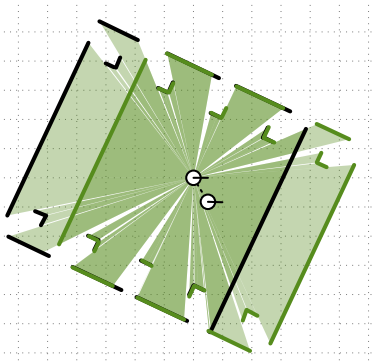
Statistics:
True positive: 0
True negative: 58
False positive: 7
False negative: 22

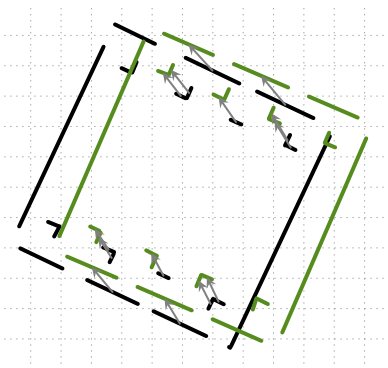




Solution: **# 13**
Reliability: 0.671
Ambiguity: 3.1
Discrepancy: 123892
⇒ **Rejected**

Statistics:
True positive: 0
True negative: 61
False positive: 4
False negative: 22





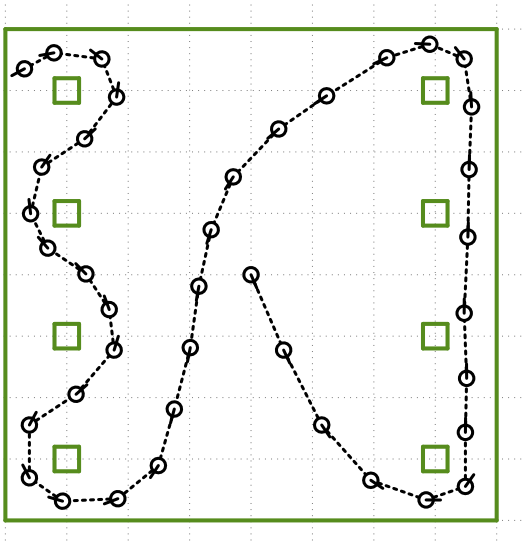
Solution: **# 14**
Reliability: 0.431
Ambiguity: 8.0
Discrepancy: 69782
⇒ **Rejected**

Statistics:
True positive: 0
True negative: 51
False positive: 14
False negative: 22

B CORRESPONDENCE SEARCH OVERALL REPORT - PILLARS DATASET

Dataset:

Environment:	pillars
Noise σ:	1.0 cm
Grid:	50 x 50 cm



Algorithm settings:

Maximal $\|\Delta x\|$:	120 cm
Maximal $\|\Delta y\|$:	120 cm
Maximal $\|\Delta \alpha\|$:	1.26 rad
Vectorization max. error :	3.0 cm
Reliability threshold :	0.10
Ambiguity threshold :	25
Ambiguity k_α:	1000
Ambiguity k_{xy}:	1
Discrepancy threshold:	2000

Description:

The *pillars* environment was designed to challenge the registration algorithm, while the robot has significantly obstructed view. The four walls around form a reliable reference and the eight pillars act as the obstacles to be mapped.

The low level of noise makes the pillars well distinguishable and the vectorization provides an adequate line segment approximations, which can take place in the registration process.

Legend:

Map edges Robot pose Robot path

Accepted results:

Scan points	Solution number	Rel.	Amb.	Disc.	True positive	True negative	False positive	False negative
#0 ⇒ #1	0	0.962	14.2	1128	22	65	0	0
#1 ⇒ #2	0	0.961	14.2	1021	21	68	0	0
#2 ⇒ #3	0	0.974	15.4	829	21	109	0	0
#3 ⇒ #4	-	-	-	-	-	-	-	-
#4 ⇒ #5	0	0.993	2.9	854	8	18	0	0

Scan points	Solution number	Rel.	Amb.	Disc.	True positive	True negative	False positive	False negative
#5 ⇒ #6	0	0.992	2.3	175	11	38	0	0
#6 ⇒ #7	0	0.974	6.5	744	17	42	0	0
#7 ⇒ #8	0	0.983	6.2	358	14	46	0	0
#8 ⇒ #9	0	0.934	3.1	255	15	37	0	0
#9 ⇒ #10	0	0.984	14.0	646	15	47	0	0
#10 ⇒ #11	0	0.906	15.4	1511	16	36	0	0
#11 ⇒ #12	0	0.968	11.2	914	10	44	0	0
#12 ⇒ #13	-	-	-	-	-	-	-	-
#13 ⇒ #14	0	0.982	10.5	799	14	53	0	1
#14 ⇒ #15	-	-	-	-	-	-	-	-
#15 ⇒ #16	-	-	-	-	-	-	-	-
#16 ⇒ #17	1	0.950	11.1	716	24	93	0	0
#17 ⇒ #18	0	0.951	6.4	799	22	70	0	0
#18 ⇒ #19	0	0.955	3.6	1099	22	64	0	0
#19 ⇒ #20	0	0.958	9.0	739	22	63	0	0
#20 ⇒ #21	0	0.964	10.7	650	19	61	0	1
#21 ⇒ #22	0	0.966	7.9	695	20	74	0	1
#22 ⇒ #23	0	0.974	22.7	1113	19	92	0	1
	14	0.797	3.3	0	0	90	2	20
#23 ⇒ #24	-	-	-	-	-	-	-	-
#24 ⇒ #25	0	1.000	0.5	589	7	15	0	0
#25 ⇒ #26	0	1.000	2.0	255	12	43	0	0
	4	0.926	0.7	0	0	41	2	12
#26 ⇒ #27	0	0.998	15.7	290	15	65	0	1
	8	0.958	10.9	0	0	63	2	16
	11	0.265	18.8	0	1	63	2	15
	12	0.259	18.6	0	2	63	2	14
#27 ⇒ #28	0	0.993	14.4	797	15	64	0	1
#28 ⇒ #29	4	0.989	13.6	1679	20	98	0	0
#29 ⇒ #30	2	0.997	6.7	1605	16	69	0	0
#30 ⇒ #31	0	0.996	9.6	874	15	46	0	0

Scan points	Solution number	Rel.	Amb.	Disc.	True positive	True negative	False positive	False negative
#31 ⇒ #32	0	0.976	8.8	1857	12	47	0	0
#32 ⇒ #33	0	0.987	6.9	786	13	34	0	0
#33 ⇒ #34	0	0.996	18.3	947	15	44	0	0
	2	0.870	20.3	0	4	38	6	11
#34 ⇒ #35	5	0.756	6.7	0	0	74	3	16
#35 ⇒ #36	0	0.979	15.1	1671	18	78	0	1
#36 ⇒ #37	0	0.921	23.9	1318	12	49	0	1
#37 ⇒ #38	0	0.973	3.3	686	8	14	0	0

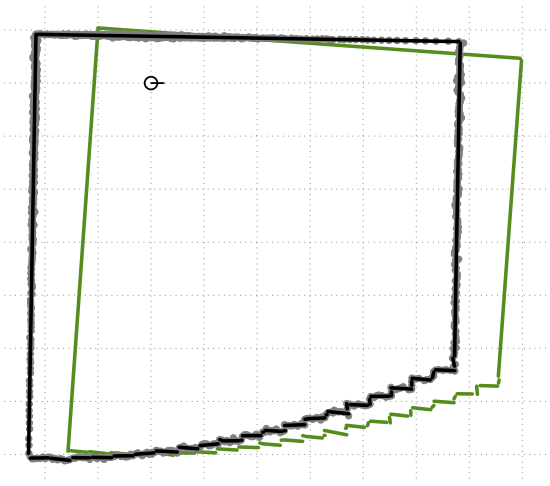
C CORRESPONDENCE SEARCH CASE REPORT - QUADRATIC DATASET

Dataset:

Environment:	quadratic
Noise σ:	1.0 cm
Grid:	50 x 50 cm

Observation:

Scan points :	#0 → #1
True Δx :	56 cm
True Δy :	0 cm
True $\Delta\alpha$:	-0.05 rad



Algorithm settings:

Maximal $\|\Delta x\|$:	120 cm
Maximal $\|\Delta y\|$:	120 cm
Maximal $\|\Delta\alpha\|$:	1.26 rad
Vectorization max. error :	3.0 cm
Reliability threshold :	0.10
Ambiguity threshold :	25
Ambiguity k_α:	1000
Ambiguity k_{xy}:	1
Discrepancy threshold:	5000

Description:

The *quadratic* environment was designed to find out, how much detailed pattern is the registration algorithm able to deal with. The quadratic slope of the stairs, makes the steps gradually better and better distinguishable.

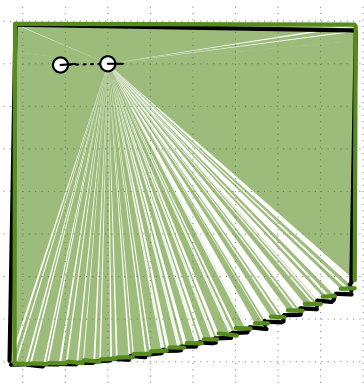
The low level of noise allows the stair steps to enter the registration process, but the lowest differences are clearly under the level of noise. Registration success ratio reflects this in the falsely set correspondences.

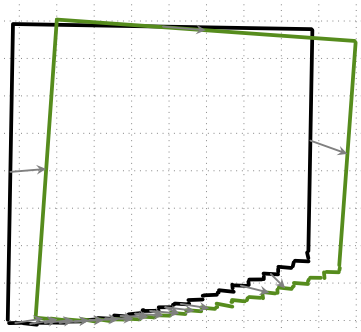
Legend:

Static scan(#0) | Registered scan(#1) | Scanned points

Correspondence | Common valid area | Discrepancy area

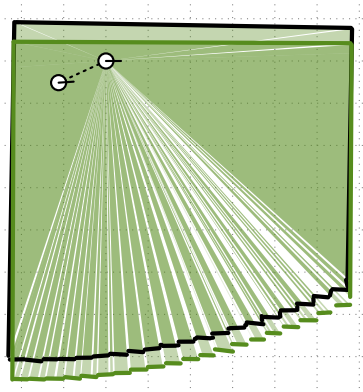
Solutions:

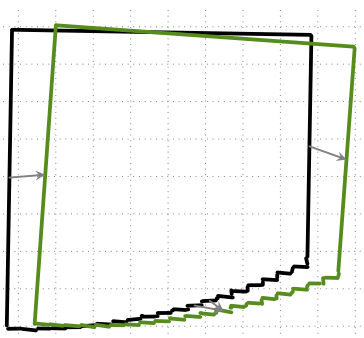




Solution: **# 0**
Reliability: 0.917
Ambiguity: 22.7
Discrepancy: 3929
⇒ **Accepted**

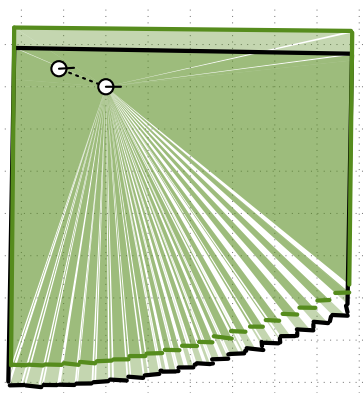
Statistics:
True positive: 16
True negative: 169
False positive: 10
False negative: 7

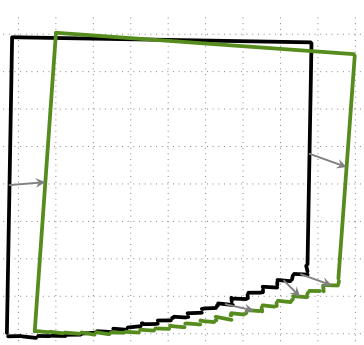




Solution: **# 1**
Reliability: 0.594
Ambiguity: 2.6
Discrepancy: 15085
⇒ **Rejected**

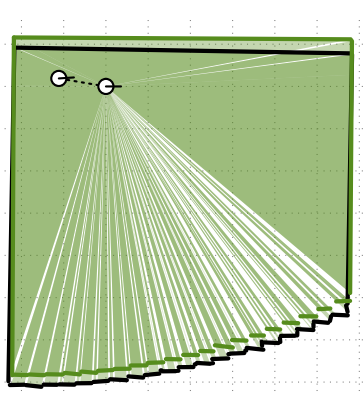
Statistics:
True positive: 3
True negative: 178
False positive: 1
False negative: 20

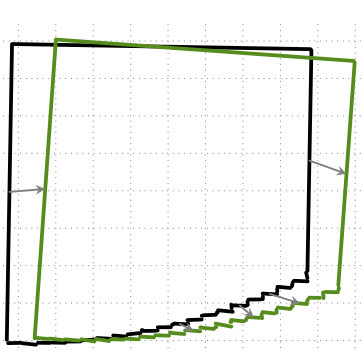




Solution: **# 2**
Reliability: 0.999
Ambiguity: 15.5
Discrepancy: 17136
⇒ **Rejected**

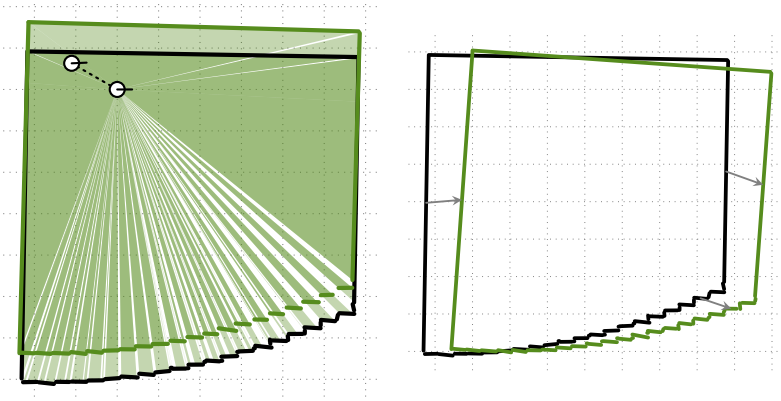
Statistics:
True positive: 4
True negative: 178
False positive: 1
False negative: 19





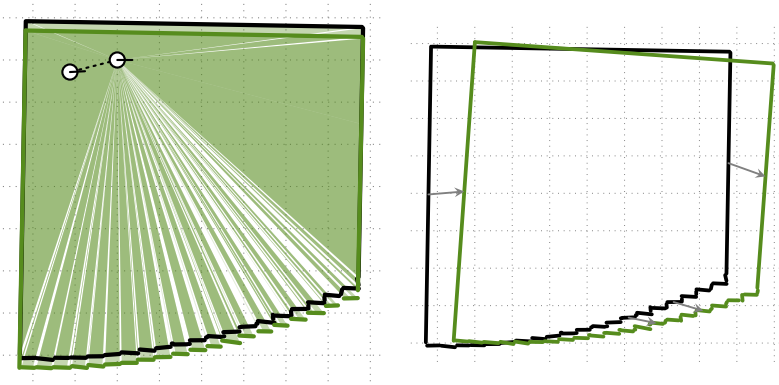
Solution: **# 3**
Reliability: 0.987
Ambiguity: 8.8
Discrepancy: 10027
⇒ **Rejected**

Statistics:
True positive: 3
True negative: 177
False positive: 2
False negative: 20



Solution: # 4
Reliability: 0.773
Ambiguity: 2.4
Discrepancy: 22595
⇒ **Rejected**

Statistics:
True positive: 3
True negative: 179
False positive: 0
False negative: 20



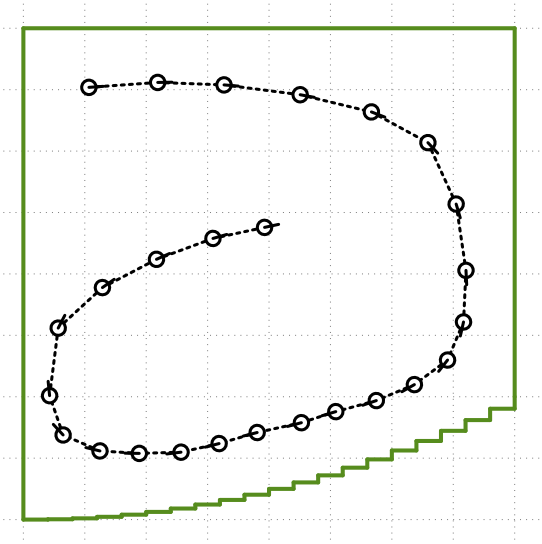
Solution: # 5
Reliability: 0.976
Ambiguity: 9.2
Discrepancy: 7639
⇒ **Rejected**

Statistics:
True positive: 4
True negative: 179
False positive: 0
False negative: 19

D CORRESPONDENCE SEARCH OVERALL REPORT - QUADRATIC DATASET

Dataset:

Environment:	quadratic
Noise σ:	1.0 cm
Grid:	50 x 50 cm



Algorithm settings:

Maximal $\|\Delta x\|$:	120 cm
Maximal $\|\Delta y\|$:	120 cm
Maximal $\|\Delta \alpha\|$:	1.26 rad
Vectorization max. error :	3.0 cm
Reliability threshold :	0.10
Ambiguity threshold :	25
Ambiguity k_α:	1000
Ambiguity k_{xy}:	1
Discrepancy threshold:	5000

Description:

The *quadratic* environment was designed to find out, how much detailed pattern is the registration algorithm able to deal with. The quadratic slope of the stairs, makes the steps gradually better and better distinguishable.

The low level of noise allows the stair steps to enter the registration process, but the lowest differences are clearly under the level of noise. Registration success ratio reflects this in the falsely set correspondences.

Legend:

Map edges Robot pose Robot path

Accepted results:

Scan points	Solution number	Rel.	Amb.	Disc.	True positive	True negative	False positive	False negative
#0 ⇒ #1	0	0.917	22.7	3929	16	169	10	7
#1 ⇒ #2	0	0.937	21.1	1930	14	175	6	9
#2 ⇒ #3	0	0.943	19.9	1106	15	180	4	8
	3	0.507	1.1	3526	3	184	0	20
	4	0.937	20.3	747	16	182	2	7

Scan points	Solution number	Rel.	Amb.	Disc.	True positive	True negative	False positive	False negative
#3 ⇒ #4	0	0.964	14.9	977	15	189	5	8
	6	0.574	0.7	3746	3	194	0	20
#4 ⇒ #5	0	0.968	17.7	1489	15	209	6	8
#5 ⇒ #6	0	0.973	12.9	724	14	214	5	9
#6 ⇒ #7	0	0.974	13.5	1321	14	194	0	9
	6	0.492	0.9	2671	3	194	0	20
#7 ⇒ #8	0	0.966	13.9	764	13	190	0	9
#8 ⇒ #9	0	0.976	13.7	1213	12	208	5	10
	7	0.299	0.5	3726	3	213	0	19
#9 ⇒ #10	0	0.979	18.9	3404	16	227	6	7
#10 ⇒ #11	0	0.979	16.4	1174	16	178	1	6
	6	0.376	0.9	4798	4	179	0	18
#11 ⇒ #12	0	0.990	14.8	1041	21	169	1	1
#12 ⇒ #13	0	0.951	15.1	454	18	176	1	3
#13 ⇒ #14	0	0.914	17.7	1504	17	153	0	2
#14 ⇒ #15	0	0.892	16.6	2492	17	134	2	0
#15 ⇒ #16	0	0.899	17.8	1014	16	120	0	0
#16 ⇒ #17	0	0.893	19.5	1251	15	112	1	0
#17 ⇒ #18	0	0.889	18.8	639	15	111	2	0
#18 ⇒ #19	0	0.897	18.4	2286	15	123	8	1
#19 ⇒ #20	0	0.912	17.4	2572	13	159	2	3
#20 ⇒ #21	0	0.906	16.5	991	12	181	4	7
	3	0.288	0.1	4470	3	185	0	16
#21 ⇒ #22	0	0.873	18.1	948	17	228	11	6
#22 ⇒ #23	0	0.875	20.3	956	19	192	0	4
#23 ⇒ #24	0	0.897	20.6	1384	19	189	0	4
	5	0.439	0.4	3760	3	189	0	20
#24 ⇒ #25	0	0.923	16.7	1361	15	179	2	8