JACOBS UNIVERSITY

School of Engineering and Science

Dissertation
for a Ph.D. in Computer Science

Improving Point Cloud Quality for Mobile Laser Scanning

Jan Elseberg

November 2013

Supervisor:	Prof. Dr. Andreas Nüchter
Second Reviewer:	Prof. Dr. Andreas Birk
Third Reviewer:	Prof. Dr. Joachim Hertzberg
Fourth Reviewer:	Prof. Dr. Rolf Lakämper
Date of Defense:	13th of September 2013

Abstract

This thesis deals with mobile laser scanning and the complex challenges that it poses to data processing, calibration and registration. New approaches to storing, searching and displaying point cloud data as well as algorithms for calibrating mobile laser scanners and registering laser scans are presented and discussed. Novel methods are tested on state of the art mobile laser scanning systems and are examined in detail. Irma3D, an autonomous mobile laser scanning platform has been developed for the purpose of experimentation. This work is the result of several years of research in robotics and laser scanning. It is the accumulation of many journal articles and conference papers that have been reviewed by peers in the field of computer science, robotics, artificial intelligence and surveying.

Danksagung

The work that goes into finishing a work like this is long and arduous, yet it can at times be pleasing, exciting and even fun. I would like to thank my advisor Prof. Dr. Andreas Nüchter for providing such joy during my experiences as a Ph.D. student. His infectious eagerness for the subject and his ability to teach are without equal.

Prof. Dr. Joachim Hertzberg deserves many thanks for introducing me to robotics and for being such an insightful teacher of interesting subjects.

Ich danke zudem
meinen Eltern, deren Unterstützung unschätzbaren Wert hat,
meinen Geschwister, Niels, Tim und Tobias,
meiner brillianten Kollegin und besten Freundin Dorit Borrmann, die mir immer mit Rat und Tat zur Seite stand,
und allen meinen anderen Freunden, die es ebenfalls verdient haben, erwähnt zu werden.
Aus tiefstem Herzen vielen Dank!

Contents

Chapter 1

Introduction

We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time.

T.S. Eliot, Little Gidding

The science of surveying, the art of measuring space, the quantification of distances and angles are all ancient dating back to at least the old kingdom of Egypt. As early as 2540 BC the Egyptians were capable of constructing almost perfectly aligned and proportioned Pyramids using only simple tools and basic geometry [43]. Since then many advances have been made in these fields. From the classical Dioptra over the Azimuthal Quadrant to the modern Theodolite, surveyors developed ever more accurate and powerful tools (Fig. 1.1). Concurrent with the increasing complexity of the equipment and the increasing number of applications, advanced mathematical methods were also developed simultaneously. For example, the method of triangulation, which is still used to this day, was first described by Gemma Frisius in 1533 [32]. More often than not mathematical advances preceded or encouraged the development of technologies. The solution to the problem of longitude, that is, the task of determining the longitudinal coordinate of a point at sea, was well known in theory many decades before it could be solved in practice, by John Harrison when he invented the marine chronometer [68].

In 1997 one of the most versatile and powerful tools for surveying, the laser scanner, appeared on the market. A laser scanner, essentially a combination of a theodolite and a laser range finder, is capable of taking several thousands of range and angle measurements a second. The ability to rapidly acquire spatial data has led to a surge of new developments not only in the field of surveying but also in related fields such as robotics and computer science. As is frequently the case in computer science, this new technology has revealed a multitude of computational problems that are in need of algorithmic solutions.

This thesis is the result of several years of research and of investigating solutions to the challenges that have newly emerged. During this time these new measurement devices were used for a variety of different applications. The most straight-forward use of a laser scanner is certainly as a surveying tool. First, one acquires several views on whatever the object, building or environment to be surveyed. Then one assembles a map, using either classical surveying

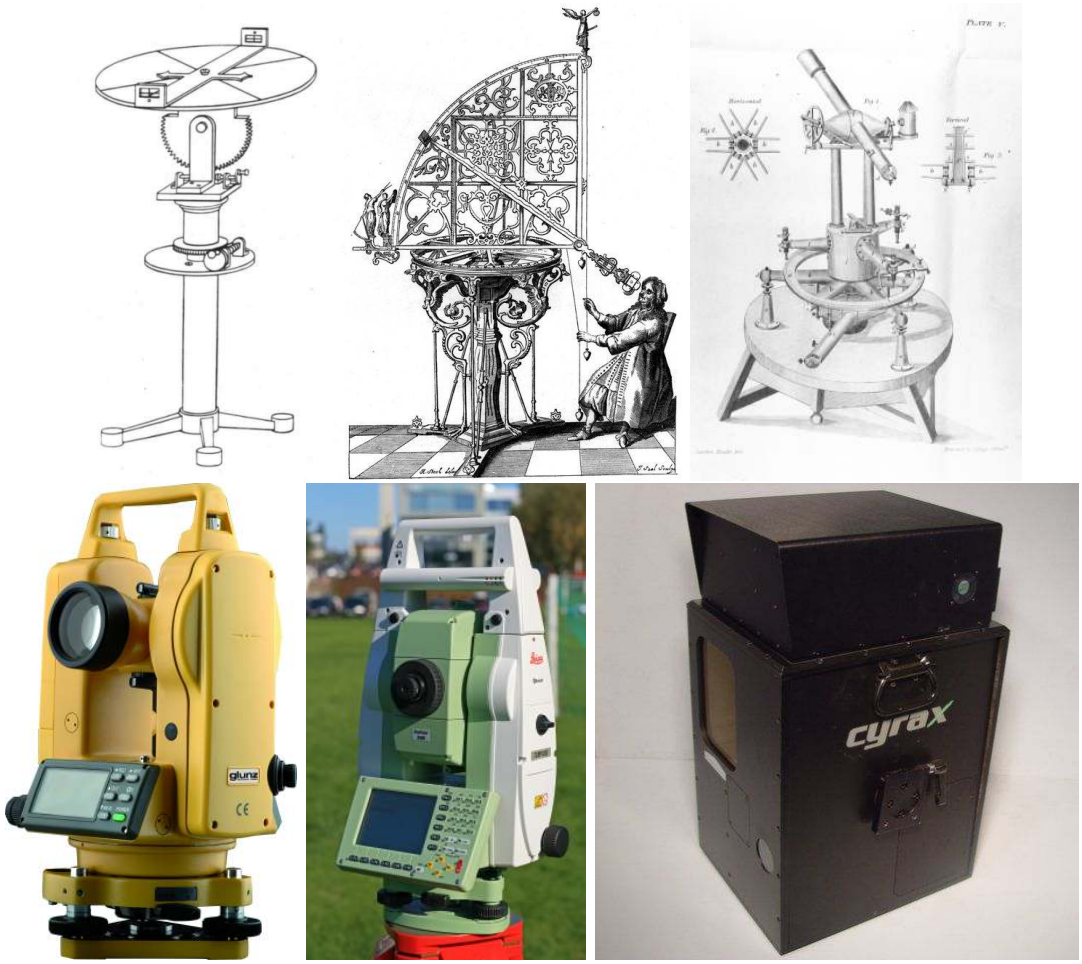


Fig. 1.1: Top from left to right: A Dioptra, an Azimuthal Quadrant and a theodolite. All these devices are used for angle measurements of points relative to the observer. The Dioptra is an ancient tool that was invented at least 2000 years ago [1]. In the 13th century the Persian scholar Nasir al-Din al-Tusi invented the Azimuthal Quadrant for observing the movement of celestial bodies [41]. The theodolite was invented in the 16th century by Martin Waldseemüller who called it the polimetrum [2]. Since then many improvements to the original design have been made, resulting in modern theodolites such as the Topcon DT-205 (bottom left) [16]. The combination of a range finder with these devices is embodied by the tachymeter (bottom middle). The Cyrax 2400 (bottom right) is one of the first laser scanners to appear on the market in 1998 [27]. Laser scanners can be considered an advanced development of the tachymeter, in the sense that they acquire the same measurements, only at a much higher rate and do not require special targets. Images taken from the manufacturers.

techniques or automatic algorithmic solutions. Automating this process further, by fixing the laser scanner on a mobile platform, creates yet again new opportunities and new challenges. Recently, this has culminated in a new process called mobile laser scanning, where the laser scanner is in use while the mobile platform is in motion. The opportunities inherent in this mode

of operation are obvious for robotics. The laser scanner is transformed from a surveying tool into a sensor valuable to the robot itself. The laser scanner enables a new mode of experiencing the environment that other sensors either cannot provide to such precision or only provide indirectly. Although a laser scanner gives a purely metrical impression of the world, machine learning techniques can be used to extract additional semantic information out of the data as well.

The main focus has been to develop methods for creating accurate digital models of the world by using spatial information that is captured by laser scanners. These algorithms ought to require the least amount of human intervention as possible, ideally none at all, to the main benefit of automating and therefore speeding up these processes. The resulting model of the world is supposed to bea spatial likeness of the object or scene that has been captured. For this, one strives to create accurate, precise, yet compact representations of the data. Combining these should not only be possible, but easy, so as to create more accurate and more detailed representations of the world.

This thesis is an accumulation of my work done on the topic of laser scanning in the format of a collection of journal articles. The papers herein have been peer-reviewed by an international community of scientists and experts in the field of computer science, robotics, artificial intelligence or surveying or a combination thereof. All of the following documents have therefore appeared in other publications before.

Manuscripts that appear in this work:

D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally Consistent 3D Mapping with Scan Matching. *Journal of Robotics and Autonomous Systems (JRAS)*, 56(2):130–142, February 2008. `http://www.journals.elsevier.com/robotics-and-autonomous-systems`

J. Elseberg, D. Borrmann, and A. Nüchter. Algorithmic solutions for computing precise maximum likelihood 3D point clouds from mobile laser scanning platforms. *Remote Sensing (accepted)* , 2013. `http://www.mdpi.com/journal/remotesensing`

J. Elseberg, D. Borrmann, and A. Nüchter. One Billion Points in the Cloud - An Octree for Efficient Processing of 3D Laser Scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76(0):76–88, 2013. `http://www.journals.elsevier.com/isprs-journal-of-photogrammetry-and-remote-sensing`

J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter. Comparison of Nearest-Neighbor-Search Strategies and Implementations for Efficient Shape Registration. *Journal of Software Engineering for Robotics (JOSER)*, 3(1):2 – 12, 2012. `http://www.joser.org`

A. Nüchter, J. Elseberg, P. Schneider, and D. Paulus. Study of Parameterizations for the Rigid Body Transformations of The Scan Registration Problem. *Journal Computer Vision and Image Understanding (CVIU)*, 114(8):963–980, August 2010. `http://www.journals.elsevier.com/computer-vision-and-image-understanding`

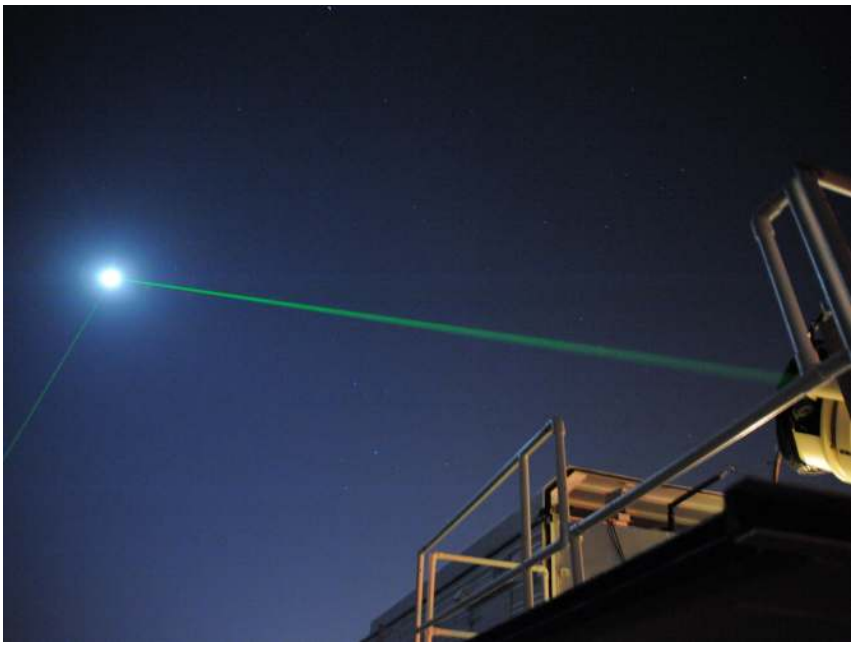


Fig. 1.2: Two green laser beams are pointed at the moon by the Laser Ranging Facility at the Geophysical and Astronomical Observatory at NASA's Goddard Space Flight Center in Greenbelt, Md. Image courtesy of NASA.

After a short introduction on the technology of laser scanning and its many possible applications in this chapter, I will describe my own research and the contributions made. Chapter 2 describes algorithms for creating consistent maps out of range measurements for static laser scanning, also referred to as terrestrial laser scanning. It also deals with the issues of processing and storing the output of range sensors. This is a crucial problem as these devices usually produce data at an astonishing rate. The next chapter, Chapter 3, deals with the more challenging problem of scanning while moving. The registration problem in this area is similar to the one for terrestrial laser scanning and parts of the solutions of the latter can be applied to the former. The chapter also describes a novel procedure for the complete and accurate calibration of mobile laser scanners in arbitrary configurations. Calibration is a step that is crucial for obtaining accurate measurements by freeing them of systematical errors. Chapter 4 then gives an outlook on the many venues of research that are still open and concludes this work.

1.1 Applications of Laser Scanning

The technology of Light Detection And Ranging (LiDAR) has been in use for more than 40 years. In the 1970s the National Aeronautics and Space Administration (NASA) experimented with LiDAR as a sensor for satellites and observatories that was intended to measure properties of the atmosphere and surface of a planet. Retroreflective targets were left on the surface of the moon during the Apollo 11 mission and are still used for observing and tracking the distance of

Fig. 1.3: The Laser Scanner RIEGL LMS-Z420i was used for measuring the wreckage of the half sunken Costa Concordia on the coast of Isola des Giglio in Italy. This kind of data can be used for monitoring the position of the ship, as well as for observing the deformation of the hull over time. Image courtesy of RIEGL Measurement GmbH.

the moon to the earth (see Fig. 1.2) [3]. The working principle of LiDAR is similar to that of Radio Detection and Ranging (Radar). A LiDAR has two main components, a light emitter and a light sensor. A laser beam or pulse is emitted from the device which may or may not hit a surface in the environment. If the light hits the surface it is influenced according to the properties of the surface, it could be absorbed, refracted or reflected diffusely, specularly or retroactively in any combination. If enough light is reflected for the light sensor to be able to register it, electronic components in the LiDAR use the reflected signal to compute the range to the surface.

Since the advent of accurate global positioning systems in the middle of the 1990s, LiDAR devices began to be utilized for topographical mapping via airborne systems. Soon after, the first terrestrial LiDAR devices appeared on the market. These laser scanners were portable and were usually mounted on a tripod when used. At first, they had a data acquisition rate of about 10.000 measurements per second. This offered a huge advantage over former state of the art surveying techniques, which usually required a few seconds for a single measurement. In addition, each target, i.e., a point that is supposed to be measured, had to be prepared in advance. Laser scanners are capable of measuring surfaces without preparation, provided that the surface offers a minimum of diffuse reflective or retroreflective properties.

However, laser scanners did not merely change the field of surveying. As laser scanners acquire a large amount of dense data, maps of unprecedented accuracy can be generated. These can be used to accurately model flood or landslides in high-risk regions. Similarly, accurate

Fig. 1.4: The VZ-400 laser scanner by RIEGL Laser Measurement GmbH is used for documenting a traffic collision [53]. 3D models of car accidents can be used for simulating the collision, for confirming eyewitness reports by checking line of sight and for documenting evidence for use in court [10].

city models are generated for better urban planning. The distribution of air pollution can be simulated [49], the amount of sunlight certain buildings receive can be accurately computed [48], the consequences a proposed building has on the view of the city can be estimated [55] and many others. The capability for fast and contactless distance measurements meant that laser scanners were quickly used in a large number of other applications as well. Laser scanners can be used for monitoring, i.e., the observation of geological faults for shifts, the observation of glaciers to measure their mass over time and the like. A related application is found in construction. While digging tunnels, laser scanners are used to ensure that there are no shifts in the earth or other reasons for deformation in the structure of the tunnel that may lead to cave-ins. For open pit mines, laser scanners are used to detect landslides early enough for evacuation. A prominent example of scanning used to evaluate impending environmental or other large scale events that may compromise workers and bystanders is shown in Fig. 1.3. In 2012 the cruise-ship Costa Concordia sank off the coast of Isola des Giglio in Italy. The vessel grounded near to the shore in an unsteady position only half submerged in danger of falling off a rocky ledge [21]. To prevent this from happening the ship was secured. During the salvage operation the ship is observed with a laser scanner, to check for any movements of the wreck and analyze possible deformation in the hull.

Laser scanners are also used at the scenes of less spectacular accidents and crimes to document the location and the position of all evidence. Such documentation is then exploited by investigators to reconstruct the progression of events or details of the crime or accident. They enable 3D simulation of the accident or crime without access to the scene with no danger of deteriorating it (see Fig. 1.4). For example, a 3D model of a murder scene can be used by ballistics experts to determine the position from which a weapon has been fired. The size and stance of the shooter is easily extrapolated with the help of the spatial information. Laser scanning is ideally suited for forensics, since the acquisition of data is done quickly and does not disturb the site.

Another kind of documentation that laser scanning is helpful for is in the field of forestry [75]. Aeriel based laser scanning is frequently used to gauge and record the amount of foliage that is

present in forests. The height, straightness and volume of trees in a wide area can be measured in a short period of time by terrestrial laser scanners. The resulting map is then used for choosing which trees to cut down and to schedule an optimal logging schedule. This not only reduces the time otherwise spent on manually measuring every single tree with calipers, but also increases the amount of lumber harvested while cutting fewer trees.

The same data acquired by aerial laser scanning in the context of forestry can reveal structures beneath the forest cover and even beneath layers of soil that are still recognizable due to the shape of the surface [61,67]. The science of geology benefits from the ability of LiDAR technology to "penetrate" vegetation in a similar way. The dense data of surface elevation enables geologists to detect geological features beneath canopy. In 1996, an aerial laser scanning mapping project of Bainbridge Island in the Puget Sound revealed fault strands within the Seattle fault zone [4]. Geologists can also study changes over time more easily as repeated surveys are simplified by laser scanning. This is often done to observe the loss of land in coastal regions due to landslides or the sea slowly eroding the coast [33].

In time, the use of laser scanners was also established in the field of archeology to document and preserve archaeological sites. Not only is the process of laser scanning faster than conventional approaches, it also produces data that can be analyzed more objectively than manual sketches or the like. Furthermore, the digital 3D models are made accessible everywhere. This is exploited to great effect for cultural heritage preservation, especially for 3D sculptures and historical artifacts like stone slabs, clay shards, tools and more. In the year 2000 a triangulating laser scanner was constructed to create highly detailed 3D models of ten statues by Michelangelo Buonarroti and all fragments of the Forma Urbis Romae (see Fig.1.5). Such models are used to study artifacts without actually having access to them. Visualizing the models of statues is also done in museums to give visitors a more interactive and engaging experience when looking at the art work [64].

Of course, 3D models of objects are also created for industrial purposes. Many types of components are frequently reverse engineered. With the help of laser scanners proper CAD (computer-aided design) models of a sample of a component are created so that the component can be manufactured without the original plans. Another industry using laser scanners is the construction industry. During construction, laser scanners allow for the rapid assessment of whether the current construction status conforms to the specifications. This is especially important for large scale projects such as tunnels. In this scenario, it must be ensured that the excavation proceeds exactly as planned, less the tunnel surfaces at a position other than intended.

Finally, robotics is a field that has made great strides with the use of laser scanning technology. 2D laser scanners can be found on almost any modern autonomous mobile robot that is actuated by wheels or tracks (see Fig.1.6). Nowadays, they are also frequently used in aerial robotics. Laser scanners allow robots to sense their environment at a speed and precision unlike sensors previously used for these tasks like sonar devices. Autonomous robots evaluate the data recorded by laser scanners to create metrical maps of their environments. More interestingly, an already established metrical map in combination with new data acquired by laser scanners can be used to locate the robot with respect to the map. This knowledge of where a robot is at any point in time is invaluable to achieve higher autonomy for mobile robots. For example, only if a robot knows its own location, it can plan routes to certain locations it intends to move towards. This task is usually referred to as navigation, or path planning. When executing a planned

Fig. 1.5: Top left: A triangulating laser scanner that projects a laser line onto the surface of one of Michelangelo's statues. The position of where the returning light of the laser line hits an array of light sensitive sensors is used to triangulate the position of that point on the object. Middle: The resulting 3D Model, after several sweeps of the statue. Right: A rendering of the model of the David statue that was acquired by the same process. Bottom left: A model from one of the shards of the Forma Urbis Romae. All 1163 fragments of the map of ancient Rome were documented with the intention of virtually reassembling the map. Images taken from the Digital Michelangelo Project [44].

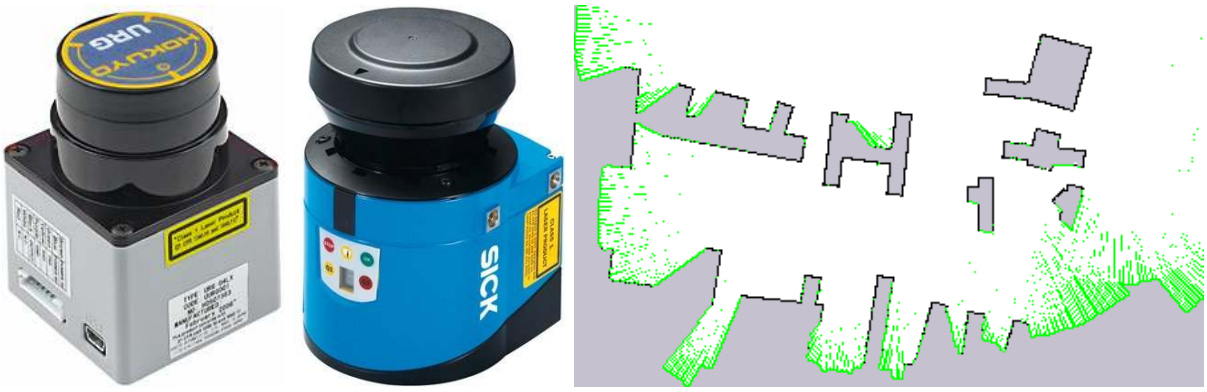


Fig. 1.6: Left: 2D laser scanners commonly used on mobile robots, the URG-04LX by HOKUYO AUTOMATIC CO.,LTD. [35] and the LMS 100 by SICK Vertriebs-GmbH [66]. Currently, these kind of sensors offer an angular resolution between 1° and 0.25°, an opening angle between 180° and 270°, a scan rate of up to 100 Hz and a maximum range of approx. 30 m. Right: A 2D map representation of a partially explored environment that was created by a mobile robot using a 2D laser scanner.

Fig. 1.7: For about a decade now, mobile robots have been using 3D laser scanners for sensing their environment. At first, conventional 2D laser scanners were actuated with servo motors to create 3D scans. The robot Kurt3D (left) employs a SICK LMS 200 2D laser scanner which is rotated around its pitching axis to acquire 3D scans. Recently, professional 3D laser scanners have begun to appear on mobile platforms as well. The robot Irma3D (middle) is equipped with a VZ-400 laser scanner by RIEGL Measurement GmbH. 3D laser scanners allow for high resolution 3D maps of the environment of the robot. These maps can be interpreted by the robot to detect objects and classify structures in the world (right). Figure taken from [57].

path trough a known but dynamic environment, the robot may encounter moving or previously unknown obstacles. A laser scanners high rate of data acquisition means that these obstacles can be detected early enough to avoid collisions. Similarly, dynamic objects of interest may be tracked and followed by the robot.

For a while now, mobile robots have started to be equipped with 3D laser scanners instead of or in addition to 2D laser scanners (see Fig.1.7). Laser scanners that are strictly 2D restrict mobile robots to environments that can be assumed to be fully represented as a flat world. First and foremost, the addition of the third dimension enables robots to map and traverse a full 3D environment. As 3D scanners naturally see more of the world as well, object recognition and detection in the range data has become more feasible as well.

The most common way for a mobile robot to be equipped with 3D laser scanning capabilities is by using a 2D laser scanner and rotating it around some axis using servo motors. The motion of the 2D scanner lets the laser beam sweep a larger field of view. This is a relatively cheap way of acquiring 3D point cloud data in comparison to professional 3D laser scanners. These too, have recently appeared on mobile platforms as they acquire measurements at a much higher rate, have a higher range and are also more accurate. One significant drawback in comparison to 2D laser scanners, that even dedicated 3D laser scanners exhibit, is the low scan rate, i.e., the time required for a single 3D sweep of the environment. Even the scan rate of 3D laser scanners designed specifically for mobile applications has so far not surpassed ≈ 10 Hz. As a consequence 3D laser scanners are frequently used in addition to 2D laser scanners which are responsible for obstacle avoidance and localization.

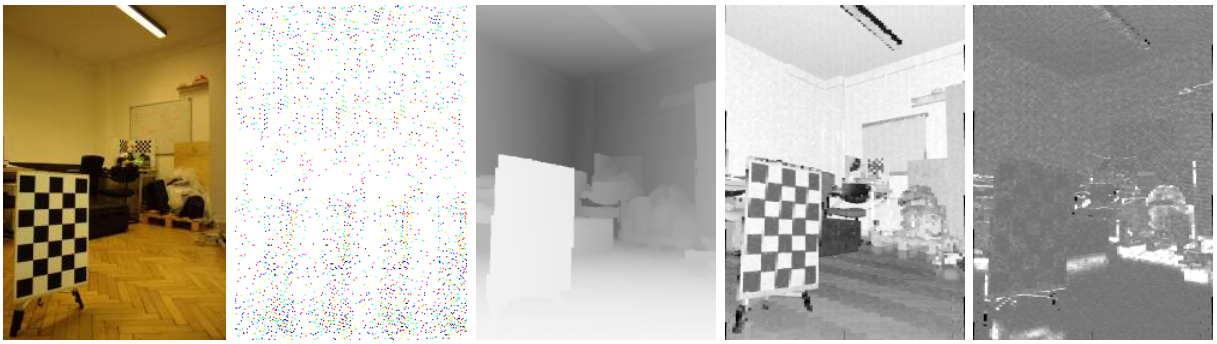


Fig. 1.8: From left to right. 1. A color image of the scene that was scanned. 2. A point cloud of the scene rendered from a different perspective. 3. A range image of the scan. Darker shades indicate higher distances to the scanner. Modern scanners also determine other modalities of the scanned surface. 4. Reflectance image. Darker shades indicate lower reflectance and brighter shades indicate higher reflectance. 5. Deviation image. Brighter shades indicate higher deviation, i.e., a reading was more noisy for a particular measurement.

1.2 Laser Scanners and Other Range Sensors

Laser scanners come in a surprising number of varieties, with several profoundly different operating principles. They all have in common that they use light, both in and outside of the visible spectrum, as a primary means to sense distances to objects. All laser scanners emit a pulse or beam of coherent light, i.e., laser. If the laser hits a surface that reflects a sufficient amount of light back to the scanner, a light sensor in the device will register it. The position, timing or other properties of the reflected light are then used to determine the distance of the surface to the laser scanner. By stepwise alteration of the direction of the emitted laser, either via mirrors or movement of the scanner itself or both, the laser sweeps a larger area of the environment so that more than a single range measurement is acquired and a full 3D view is assembled. The result of this process is usually in the form of a set of points with at least the coordinates x_i, y_i, z_i. This is commonly referred to as a point cloud. An example of this is seen in Fig. 1.8. If the sweeps of the laser scanner are in a regular pattern, the output can also be a range image as depicted in Fig. 1.8. In addition to the distance, modern laser scanners sometimes give additional information per point measurement like reflectivity or a measure of certainty.

We can separate laser scanners into three main distinct types, based on the principle which they use to measure the distance to a surface. There are triangulating laser scanners, used primarily for short range object modelling. One of these types of scanners is depicted in Fig. 1.5. Triangulating laser scanners project a laser beam, often shaped to a line, onto an object. A light sensitive sensor array, i.e., a digital camera is used to record the reflecting light. The distance of the laser scanner to the point of reflection can be determined by the position on the sensor array where the reflection was detected. The position on the sensor array directly correlates to the angle of the reflection to the laser scanner. As the distance between the laser emitter and the sensor array as well as the angle of the emitted laser beam is known, the distance to the point of reflection can be computed via triangulation. A 3D scan is generated by rotating the laser

emitter as well as the sensor. The laser beam should be as tightly focussed as possible while still being detectable at the ranges the scanner is intended for. Likewise, the resolution of the digital camera should be as high as possible to increase the measurement accuracy.

Long range 3D laser scanners work by one of two principles. One is commonly known as the phase shift method, also referred to as phase-based laser scanning. Phase-based laser scanners emit a continuous laser beam. The optical intensity of the laser beam is modulated sinusoidally. The reflected laser beam is measured and its phase is compared to the reference signal that was emitted. The change in phase between the reference and the received signal is directly correlated to the time the original signal needed to return to the scanner, and this time is directly correlated to the distance to the object. Since the laser emitter is always active this technique can determine distances to targets at a very high rate.

The phase difference ψ is the sum of an integral multiple of 2π and a part $\Delta\varphi$ which represents the additional fraction of the wave cycle that has elapsed since the last full cycle:

$$\psi = a \cdot 2\pi + \Delta\varphi \tag{1.1}$$

The distance to the measured object is then given by multiplication with $\frac{\lambda}{2\pi}$, where λ is the wavelength of the modulation:

$$2D = a\lambda + \frac{\Delta\varphi}{2\pi}\lambda. \tag{1.2}$$

Since the laser beam must travel to the object and back this results in twice the distance. Detecting the phase shift between two sinusoids is ambiguous, i.e., the value for a is inherently unobservable in this context. As a consequence the range measurement is also ambiguous. This is referred to as the ambiguity range. The apparent solution of using a modulation with a particular long wavelength is not entirely feasible. The accuracy of the measurement of the phase difference is strictly correlated with the wavelength itself. The larger the wavelength, the less accurate is the measurement. As a consequence, the trade-off for larger measurement ranges is a decrease in measurement accuracy. Consequently, the larger the range of the laser scanner, the less accurate each range measurement [39]. Modern laser scanners mitigate this problem by using multiple modulations of the laser beam at different wavelengths. In effect the scanner is measuring an object multiple times. Short wavelengths are used for precise but highly ambiguous measurements, while the longer wavelengths are used for coarse measurements and to resolve the ambiguity of smaller wavelengths. This is illustrated in Fig. 1.9. Nonetheless, phase-based laser scanners suffer from a reduced maximal measurement range when compared to other types of terrestrial laser scanners.

Another method of laser scanning is the time-of-flight principle. A laser scanner of this type will send out a very short laser pulse. The pulse is reflected towards the scanner where a receiver detects the light and records the time of arrival. The time-of-flight, i.e., the time that elapsed between emitting the pulse and receiving it can easily be used to compute the distance to the target via the speed of light. The main challenge in this approach is the requirement for extremely high precision clocks. In a single nanosecond light travels approximately 30 cm, which means a laser scanner outfitted with a clock of nanosecond accuracy would only have a ranging accuracy of 15 cm. Compared to the phase-based principle this type of laser scanning is typically slower,

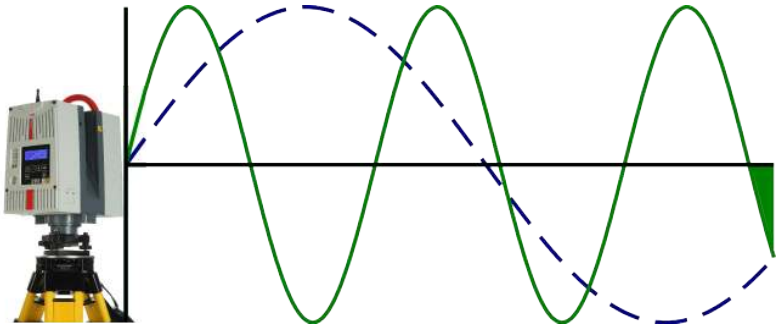


Fig. 1.9: The working principle of modern phase based laser scanner. Laser beams that are intensity modulated with different wavelengths are emitted continuously. Once reflected back to the device, the phase difference of the signal is determined. This is equivalent to measuring the phase of the last incomplete wave period. In this image, the dashed wave is used to coarsely resolve where in the cycle the solid and more precise wave encountered an object.

since discrete pulses have to be emitted and received for each measurement. However, these types of scanners offer some benefits over pulse-based scanners. First, they usually allow for a higher maximal measurement range. In fact, since there is no ambiguity distance, the maximal range of a time-of-flight, or pulse-based, scanner is only restricted by the strength of the laser emitter and the sensitivity of the light sensor. Another advantage that only pulse-based laser scanners offer is the so-called Full-Wave-Analysis (FWA). A scanner employing this technique does not merely detect the first reflection received, but digitizes the entire response as illustrated in Fig. 1.10. The response can then be analyzed to extract multiple echoes. Multiple echoes occur when a laser pulse hits more than a single surface. This usually happens when vegetation is present, i.e., a pulse only partially hits one or several branches or leaves and possibly some obstacle behind that as well. In terrestrial laser scanning multiple echoes also frequently occur on the edge of other objects, such as sign posts, buildings, fences and many others. Thus, FWA enables the scanner to determine the range of multiple objects using only a single laser pulse. In addition to this, FWA also allows for a closer examination of each of the echoes. For example, the standard version of the VZ-400 laser scanner by the RIEGL Measurement GmbH does on-board FWA by fitting Gaussians into the sensor response. It then returns the amplitude and the deviation thereof in addition to the calibrated reflectance for each echo. Other laser scanners return the entire digitized response, such that FWA can be done after the fact. Kirchhof et al. do FWA to generate local surface primitives such as planes to further refine the range measurements and even detect new echoes that would be missed due to a weak response [42].

This is by no means a complete overview on all methods used for laser range finding. There are several other techniques, like the frequency modulated continuous wave (FMCW) method, which is similar in principle to the phase-based method. FMCW laser range finders detect the difference in frequency between the emitted and received laser signal. These devices either modulate the frequency of the light itself or they frequency modulate the intensity with which the laser beam is emitted. The correlation method sends out a modulated laser signal which is correlated with the received reflection, i.e., a resemblance between the output and input is detected to estimate the

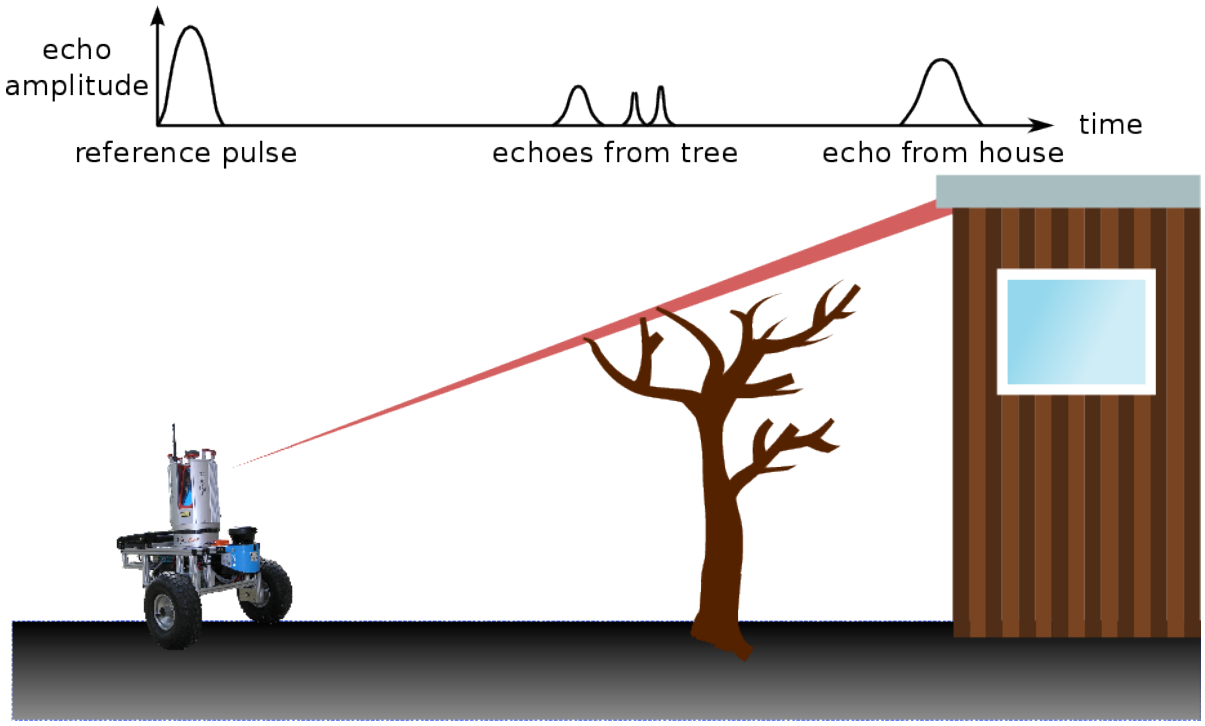


Fig. 1.10: The working principle of a pulse based laser scanner. A short pulse of light is sent out. The pulse partially hits and is reflected from the branches in the tree. The remaining light is reflected from the building. The laser scanner itself receives the reflections at different times, which is correlated to the distance of the surface that reflected the initial beam. The entire response of the sensor can be digitized, so that the amplitude, the deviation and timing of multiple echoes can be analyzed. This process is referred to as Full-Wave-Analysis.

time needed for the signal to return. This is similar to pulse-based systems in that time-of-flight is directly measured. There is also interferometry which relies on the effect of interference to measure range based on the power or shape of the resulting beam. However, unlike the pulse-based and the phase-based methods none of these techniques are used for long and intermediate range laser scanning. Table 1.1 lists the current top-of-the-line terrestrial laser scanners from the four largest producers. All of these devices have a horizontal field-of-view of 360° and are therefore also called panoramic laser scanners. There are also so-called camera-scanners, with a field of view similar to that of a camera. However, these have become much less common in recent years. Another recent development on the market is the addition of digital cameras that are integrated into the device to directly give color information for each range measurement. Pulse-based systems sacrifice speed for range, whereas phase-based systems do the opposite. However, in the last ten years, pulse-based laser scanners have continued to become faster and phase-based laser scanners have increased their maximal measurement range. Note, that Table 1.1 only lists the precision of each device instead of their accuracy, the latter of which is usually somewhat worse than the precision. In this context, accuracy refers to how close a range measurement typically

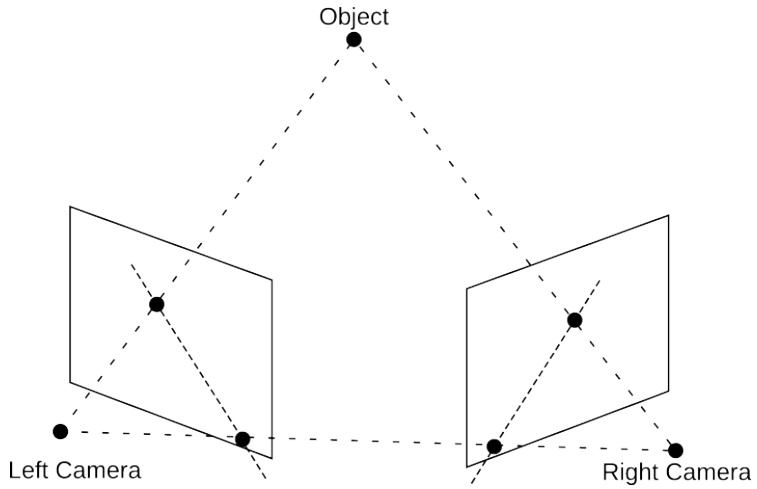


Fig. 1.11: The geometric relationship between two cameras in a stereo vision system. Both cameras are assumed to be modelled by the pinhole camera model [24]. The cameras are calibrated to minimize deviations from this model. An object is projected on the image plane of each camera. When the relation between the two cameras is known (via external calibration) a projection on one image plane defines an epipolar line on the other image plane. If the object is not occluded for the second camera, it must be located somewhere on this line. This simplifies the establishment of correspondences to a large degree. Once the location of an object in both images is established the 3D coordinate of that object can be computed via triangulation.

is to the actual ground truth distance to the target. However, precision refers to how repeatable a measurement is, i.e., how large the deviation of repeated range measurements towards the same target is. The precision is given as one standard deviation of the range that is repeatedly measured at the same target for a specified distance and a specified reflectivity.

In addition to these terrestial laser scanners, a number of other sensors exist that also acquire range information. Table 1.2 lists range sensors that are frequently used in the field of robotics. Range sensors other than laser scanners are usually much less accurate and usually offer a much more narrow field of view as well. However, since they provide a much higher frame rate than the laser scanners in Table 1.1, they are extremely useful in applications such as obstacle avoidance, motion detection, and object manipulation. Although not low-cost these devices are also more affordable than terrestial laser scanners.

The Bumblebee2 by Point Grey belongs to the class of stereo cameras. Stereo cameras are two calibrated cameras with an established baseline between them. Both cameras capture RGB images of the same scene at the same time. Computer Vision algorithms are applied to such an image pair to establish correspondences, i.e., to establish which part of one image correspond to the other image. An object seen by both cameras will appear in different positions in the two images. Since both cameras are calibrated these positions can directly be transformed into the pairs of angles that the object is seen at from each camera position. All necessary values are then known to perform triangulation to determine the distance of the object to the stereo camera. This is illustrated in Fig. 1.11.

Table 1.1: Comparison of modern terrestrial laser scanners. Data provided by the manufacturers of the devices [22, 27, 30, 69]. Precision data is given for targets with reflectivity > 80%.

	RIEGL VZ-400	Faro Focus 3D 120	Zoller+Fröhlich Imager 5010C	Leica HDS6200
				
Type	pulse-based	phase-based	phase-based	phase-based
Max. Range	600 m	120 m	187 m	79 m
Vertical field of view	200°	320°	320°	310°
Acquisition Rate (pts./s)	122.000	976.000	1.016.027	1.016.727
Laser Class	1	3R	1	3R
Precision at 25 m	NA	0.95 mm	0.5 mm	1 mm
Precision at 50 m	1.5 mm	NA	0.8 mm	2 mm
Additional Notes	Full Wave Transform allows multiple targets to be acquired	Integrated Camera for colored point clouds	Integrated Camera for colored point clouds	

Table 1.2: Overview on range sensors.

	Point Grey Bumblebee2	Microsoft Kinect	SwissRanger SR4000	Velodyne HDL-64E S2	Raytrix R5
Category	Stereo Camera	Structured Light	ToF Camera	Laser Scanner	Light field Camera
field of view	97° × 72°	57° × 43°	69° × 56°	360° × 26.8°	variable
Resolution	1024 × 768	640 × 480	176 × 144	1024 × 64	up to 1 MP
Acquisition Rate (Mpts./s)	15.7	9.2	1.2	1.3	up to 180
Scanning Rate	20 Hz	30 Hz	50 Hz	20 Hz	180 Hz

The Microsoft Kinect is a game controller that was designed or the Xbox 360 console [15]. Shortly after its release on the market, its potential use as depth sensor for robotics and automation tasks was quickly discovered. Within weeks the first open source drivers were developed, to enable the use of the Kinect on platforms other than the Xbox 360. As the Kinect is intended for the consumer market, its low price led to a surge of development in the robotics community, with many autonomous robots using the Kinect as a primary or main vision sensor. The Kinect uses the technology of structured light. Structured light can be understood as active stereo vision. Instead of two cameras, a structured light system employs only a single camera and a structured light pattern projector. The projector projects one or multiple patterns onto the environment. The pattern will be deformed by the shape and distance of the objects in the environment and reflected towards the camera. The camera image is analyzed to establish how the pattern has been deformed, i.e., to find where the features of the known pattern are projected onto the image plane. The same geometry as in Fig. 1.11 applies. Again, using the principle of triangulation the position of an object is determined by correlating a part of the projected pattern, which is at a known angle, to that part of the image onto which the pattern has been reprojected. The Kinect uses a single infrared pattern that is continuously projected by an infrared laser diode. This is combined with a monochrome CMOS sensor that captures the scene for depth sensing and an RGB camera that can be used to color the depth image. A significant downside to this approach is the low robustness against external light sources. The Kinect is designed for indoor use and will fail to acquire depth information when exposed to sunlight or other strong sources of light that will overpower the CMOS sensor.

The technology employed in the SwissRanger SR4000 is very similar to phase-based laser scanning [36]. The camera continuously sends out intensity-modulated light. The reflected light is gathered by the lens of the camera and is focussed onto a light sensitive array. The phase difference between the emitted and received signal is measured for each pixel in the array. Consequently, the distance to the target can be computed for each pixel. This is not referred to as laser scanning because instead of the light being generated by a laser the SwissRanger SR4000 employs simple LEDs to shine infrared light. These cameras are affected by the same problem as the Kinect. Bright external lighting that enters the sensor will obfuscate the emitted light pulse. This leads to noisy or no range measurements in sunlight or in other similar conditions.

It should be noted that one of the devices listed in Table 1.2, the Velodyne HDL-64E S2 is in fact a pulse-based laser scanner [73]. It achieves its high acquisition rate by 64 discrete laser emitters and receivers that rotate continuously around the main axis of the scanner. Unlike the previous two sensors, the Velodyne has much less problems with sunlight. As a consequence, it is frequently used for autonomous driving of cars [51, 72].

Finally, the Raytrix R5 is one in a line of cameras that are commonly known as light field cameras [29]. Light field cameras capture the 4D light field of a scene instead of a mere projection of the scene onto an image plane as conventional cameras do. A 4D light field is defined as the amount of light travelling through each point in space in every direction. A common design of a light field camera is shown in Fig. 1.12. The design of a conventional digital camera is enhanced with a number of smaller lenses aligned in a regular grid, i.e., an array of microlenses. The array is inserted between the main lens and the photosensor. The focus of the main lens is then modified to be on the array of lenses instead of on the sensor. Each of the microlenses separates the rays of light that converge on it so that they will be measured by several pixels of the sensors.

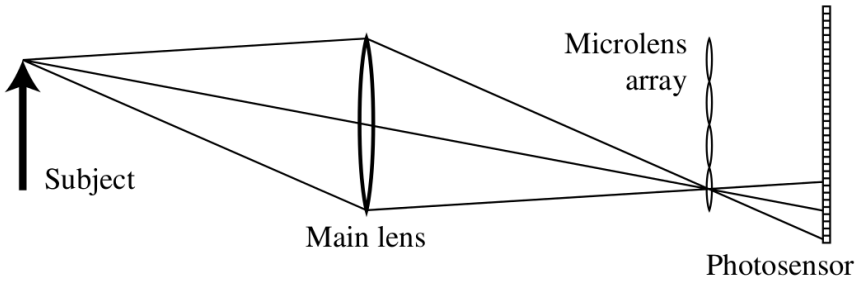


Fig. 1.12: Common design for light field cameras. A conventional digital camera design is enhanced by adding an array of small lenses between the main lens and the sensor. Light rays coming from the subject are focussed on the microlens array. There, the light rays are separated again, such that rays going into different directions can be sampled from individual pixels on the sensor. In this way the 4D light field of a scene can be captured, and depth information extracted. Diagram taken from [54].

This drastically reduces the resolution of the resulting image, but it also enables the camera to measure the incidence angle of the light rays. The captured data can be processed to create some extraordinary effects. One interesting facet is that an RGB image acquired by a light field camera can be refocussed after the fact. Once a light field image has been taken, a user can manually select a plane of focus and a virtual aperture, such that an RGB image focussed on the plane of focus is rendered. It is even possible to render an RGB image with perfect focus, such that all objects seen by the light field camera are in focus.

A more relevant aspect is that any light field camera is in effect a range sensor as well. Assuming the camera is calibrated, finding the perfect focus plane for each pixel is equivalent to finding the distance of that pixel to the camera. The range can also directly be computed by considering the image on the sensor as a composite of a continuum of stereo images. This technology allows for very high frame rates when acquiring high resolution RGB/range images both in indoor and outdoor scenarios. Unfortunately, the resolution of the range measurements themselves is poor in current light field cameras in comparison to the other range sensors in Tab. 1.2. Modern light field cameras resolve each range measurement at only hundreds of discrete steps, instead of the thousands of discrete steps even cheap hardware like the Kinect allows. Furthermore, light field cameras have only just emerged on the consumer market with the Lytro [26], so that they are still somewhat expensive. Consequently, these cameras are not yet used in robotics in a wide spread manner.

Chapter 2

Rigid registration for terrestrial laser scanning

> There are too many of us and we are all too far apart.
>
> Kurt Vonnegut,
> Welcome to the Monkey House

Whatever measurement device is used and whatever application is considered, there is one crucial problem that deserves discussion. It is referred to as the problem of rigid registration. In abstract terms, registration is the task of putting together a complex 3D jigsaw puzzle. Each piece of the puzzle is a single, or a set of multiple range measurements that have all been acquired from the same position. This chapter is concerned only with terrestrial type laser scanning. We assume that there is a set of point clouds, each of which was acquired from a discrete position using a range sensor that is not in motion during acquisition. Like a piece of the puzzle, one such 3D scan may be freely moved and rotated in space, but should not be stretched, separated, bent or deformed in any way. Translations, rotations and combinations thereof are rigid transformations. Scans are only allowed to act as rigid bodies, i.e., the distance between any pair of points of the scan must remain constant. This holds true only for rigid transformations.

Like in a puzzle, the pieces are at first not neatly arranged, but are instead disconnected and unordered. A point cloud returned by a 3D laser scanner may be highly accurate, dense and representative of the scanned objects. However, all range sensors measure the environment only relative to themselves, i.e., each measurements is but a local impression of the world. A priori, the location of the range sensor itself is unknown, so that no global measurement is available. This also means that, it is unknown how two or more point clouds taken from different locations relate to each other.

The pieces of the puzzle must therefore be assembled to create the whole. Instead of a rewarding picture of a horse or the like, we aim towards a complete and global representation of the part of the world that is locally represented in each point cloud. “Complete” of course, because all of the acquired data should be used for a representation that is as exhaustive as possible. The property “global” refers to the crux of the matter, which is finding a non-contradictory

relationship between all local systems. But "global" can mean many things. In the field of geodesy a global representation is one that is anchored to the coordinate system of the earth. Indeed, "global" is only meaningful if defined with respect to some coordinate system. In the context of this chapter we are satisfied with any arbitrary coordinate system to which we can relate our point clouds. In practice, we choose the local coordinate systems of any one of the point clouds. This choice is truly arbitrary: Once assembled, a jigsaw puzzle acts as a whole. Using any piece as an anchoring point, it can be moved and rotated at will. As long as the pieces stay together the picture will remain the same. The same holds true for our global representation of the world. No matter which local system defines the global coordinate system, the relations between the local point clouds remain the same. Picking and choosing the anchoring point is equivalent to applying a translation and a rotation to each local point cloud. Doing so cannot modify the relations between the local point clouds.

Solving the rigid registration problem means finding the location and orientation of each scan position in the global coordinate system. A position and orientation is combined into the so-called pose. A pose is a six dimensional vector $(x, y, z, \theta_x, \theta_y, \theta_z)$ that fully describes the 6 degrees of freedom (DOF) position and orientation of a rigid object. The position is described by x, y and z, which define the translation of the local coordinate system in the global Cartesian coordinate system. The orientation is defined by the values for θ_x, θ_y and θ_z. These are called Euler angles, where each dimension represents a rotation around one axis in the coordinate system [20]. Using these three values we can construct the rotation matrix

$$\boldsymbol{R}_{\theta_x,\theta_y,\theta_z} = \begin{pmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z & -\cos\theta_y \sin\theta_x \\ \sin\theta_x \sin\theta_z - \cos\theta_x \cos\theta_z \sin\theta_y & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z & \cos\theta_x \cos\theta_y \end{pmatrix}. \tag{2.1}$$

Any point $\boldsymbol{p} = (p_x, p_y, p_z)^T$ in the local coordinate system of a scan position can be transformed into the global coordinate system by:

$$\boldsymbol{p'} = \boldsymbol{R}_{\theta_x,\theta_y,\theta_z} \cdot p + \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{2.2}$$

It should be noted that there are many possible ways to define Euler angles and the rotation matrix $\boldsymbol{R}_{\theta_x,\theta_y,\theta_z}$. These depend on the order and direction of rotation around the coordinate axes, as well as the coordinate system as well. The formula given in Eq. 2.1 is given for a left handed coordinate system. The order of rotation is x-axis first, then around the y-axis and finally around the z-axis. For a more detailed description on Euler angles see [19]. The definition given in Eq. 2.1 and Eq.2.2 is the definition used throughout this thesis. Furthermore, Euler angles are not the only way to parametrize rotations in 3D. Other possibilities are quaternions and the axis-angle representation. For more information on the many different ways to represent rotations in 3D see [19].

The registration problem can be solved by direct observation of these values for each scan. This can involve global positioning systems (GPS), levels, compasses and inertial measurement units (IMUs) on the range sensors themselves or total stations external to the scanner. Measuring the pose of each scan has several disadvantages. Each additional sensor comes with a price. The

Fig. 2.1: A selection of different types of targets used in terrestrial laser scanning [25, 31, 45, 60]. The top three images show targets whose location in the point cloud is determined by their shape. The sphere (top left) is a popular target, as it is usable from all viewing angles. The prism (top right) is a retroreflector, the position of which can be measured to a very high degree using a total station. The bottom images show planar targets, where the reference point is determined via the reflectance values of the range measurements. The rightmost picture depicts a type of retroreflective target used by RIEGL sensors.

most obvious price is of course the monetary expense. To sufficiently solve the registration problem purely by sensing requires expensive high quality hardware. Beyond that, additional sensors increase the complexity of the scanning task. For example, if a total station is the tool of choice for measuring the scanner position the personnel undertaking the scanning must be trained in the use of a total station in addition to the laser scanner. As a matter of fact, this way of solving registration is often more time-consuming than the scanning itself because more data needs to be collected, stored and evaluated. Consequently, measuring the pose for each scan position directly is very rarely done for static laser scanning. There are other manual techniques for registering laser scans to a global system. These will be reviewed shortly in the next section. Afterwards, automatic techniques for registration are presented.

2.1 Registration based on Artificial Landmarks

The usual method of registering point clouds in the surveying community is via so-called landmarks. Landmarks are artificial control points that are easily recognized and easy to locate in a point cloud. For this purpose, specially designed markers are placed in the environment that is to be measured. The markers must be fixed to static parts of the scene to ensure they do not move in-between the scans. They must also be visible in at least two scans. Each marker corresponds to some real world datum called reference. The reference for most markers is a point coordinate.

There are several types of markers, each with different shape, material and design. See Fig. 2.1 for an overview on the different marker types. Every type comes with its own set of advantages and disadvantages [40]. A sphere with a known radius is easily identified in point clouds that may be acquired from any vantage point. The reference, i.e., the centre of the sphere, is easily computed by fitting a sphere into the point cloud. This reduces location errors due to sensor noise, since the point location is computed using many range measurements. Producing standardized spheres with constant radius is challenging, which makes these targets expensive. A cylindrically shaped target also reduces the effect of sensor noise on the estimate of the reference by having multiple range measurements contributing to its computation. Another advantage is that the reference is a line, which eliminates one more degree of freedom in the registration process. However, they are not usable from all vantage points and are also expensive to produce. A target used frequently in surveying is the retroreflector prism, also called corner cube. Any surface or device that always reflects light towards the source, regardless of its angle of incidence is referred to as a retroreflector. The prism achieves this effect by three mirrors that are arranged perpendicular to each other. Any beam of light entering the prism is reflected in the same direction with a small translational offset. The reference for these targets is the point where all three mirrors meet. A total station can align itself to the exact center of the prism and thus measures the target to a very high precision. Laser scanners, by their very design, cannot do this. Thus, they cannot achieve as high a precision as total stations on these targets. For this reason, prisms now play mostly a supporting role in terrestrial laser scanning.

There are also types of targets whose reflectance properties play an important role in identifying them in a point cloud and in computing their reference values. These targets are usually planar and have some pattern on them, like a checkerboard or concentric circles. Their main advantage is their price, as they can be produced cheaply, as well as their ease of use. Shape targets require some kind of mounting mechanism to keep them in place. They must be placed on tripods, or drilled into structures. Cheap planar targets can be attached to many planar surfaces. The clear disadvantage of these targets is that their outline will change depending on the viewing angle. This may effect the estimation of the reference point, especially when less measurements lie on the target. This can be overcome by mounting them on a device that allows for rotations around the targets reference point. A second issue with planar targets is that laser scanners often have systematic errors in their range measurements that depend on the reflectivity of the surface. This is especially pronounced on retroreflective targets such as the RIEGL disc in Fig. 2.1. The range measurements for this target will tend to be behind the true targets position. However, these systematical errors can be accounted for when the target and scanner are calibrated correctly.

After placing the markers in the scene the scanning can begin. During scanning it must be taken care that a few of the markers are visible in each scan. Usually, at least three markers are necessary for this. This number can be lower if the type of marker restricts the degrees of freedom to a sufficient extent like the cylindrical marker. The position of some of the markers is often measured to a high precision in some global coordinate system. These points are referred to as fixed points. However, laser scanner manufacturers have begun to provide landmark based solutions to the registration problem that do not require the location of the fixed points to be known in the global coordinate system. The laser scans are then simply registered in the project coordinate system, which is equivalent to the local coordinate systems of one of the laser scans.

First the location of the targets are established in each point cloud. This is done either manually or semi-automatically by thresholding the reflection values for retroreflective targets, simple shape recognition in 2D for checkerboard targets or 3D for non planar targets. The latter process is often only semi-automatic due to false positives or negatives in the recognition processes that need to be corrected manually. The reference point of each target is established in the local coordinate system of each scan. The targets are then identified, i.e., the many manifestations of a target in different point clouds are assigned to each other. This can happen manually or automatically, the latter usually using the RANSAC algorithm [23]. By knowing which landmark is located where in each scan, the registration proceeds by bringing the reference points into consistent positions in the global coordinate system [56].

2.2 Automatic Registration

The 3D jigsaw puzzle analogy fails in one regard. Unlike pieces of a jigsaw puzzles, 3D scans overlap and do not neatly align on their edges. When a range sensor is used to map an environment it is likely that many surfaces will be sampled more than once. Structures and features of the environment will therefore appear in multiple point clouds. With landmark based registration this is a forced effect, i.e., features are manually placed in the scene and are deliberately scanned from several positions. However, the property of overlapping point clouds can be exploited for automated solutions to the registration problem without the need to manipulate the environment.

Let us at first consider a simplified registration problem with only two point clouds that need to be aligned to each other. The basics behind many automatic registration algorithms is to find points or features in each scan that both refer to the same real world object. These common points must not necessarily be part of the point clouds as such. Their position must only be clearly defined in the local coordinate system of each scan. An example of this is the corner of a room which has not been directly measured. The position of such a corner can instead be inferred from the three intersecting surfaces that have been measured. Once a minimum number of corresponding points has been established, the two scans are matched together. Fundamentally, this constitutes a chicken and egg problem in the sense that it is trivial to establish corresponding features when the correct orientation and translation between the two scans is known. Similarly, it is trivial to compute the pose difference between a pair of scans once corresponding features are known. Assuming the features are points, this is achieved by minimizing the summed distance

这是对点对应的解释

between the paired points m_i and d_i:

$$E(\boldsymbol{R}_{\theta_x,\theta_y,\theta_z}, \boldsymbol{T}) = \sum_{i=1}^{N} \left|\left|\boldsymbol{m}_i - (\boldsymbol{R}_{\theta_x,\theta_y,\theta_z}\boldsymbol{d}_i + \boldsymbol{T})\right|\right|^2 . \tag{2.3}$$

Here, $\boldsymbol{R}_{\theta_x,\theta_y,\theta_z}, \boldsymbol{T}$ defines the difference between the pose of the two scans, where $\boldsymbol{T} = (x, y, z)^T$ is a translation vector. There are several efficient solutions for finding the parameters that minimize $E(\boldsymbol{R}_{\theta_x,\theta_y,\theta_z}, \boldsymbol{T})$.

To overcome the chicken and egg nature of the problem, several researchers have proposed the Iterative Closest Point (ICP) algorithm [5, 12, 76]. The algorithm assumes that there is some a priori estimate of the correct parameters. In practice a rough pose estimate is often available. This is especially true in robotics where cheap sensors on the robotic platform, like wheel encoders, GPS devices or IMUs, are used to estimate poses between scan positions. The key idea is to use the pose estimate to estimate point correspondences based on the nearest neighbour principle. In other words, the two scans are aligned according to the pose estimate. Then, for each point in one scan its corresponding point is determined via selecting the closest point in the other scan based on the euclidean distance measure. The so established pairs of points are used to re-estimate the pose difference between the scans by minimizing Eq. 2.3. This process is iterated until there is no more overall improvement in the error measure or there is only insufficient change in the pose estimate. It can be proven that under some conditions the ICP algorithm always converges [5]. One of these conditions is that there is no distance too large for a point pair to be accepted. Especially when aiming for a high precision alignment of two point clouds this condition is not applicable. However, even when relaxing these conditions in practice the algorithm usually converges. Of course, the ICP algorithm is dependant on the initial pose estimate as it will only converge to a local minima of the overall error function.

Aside from the ICP algorithm and variants thereof, there are several other registration algorithms. Some are similar in that they establish corresponding features like lines [82] planes [7, 59] or other shape features [65]. Some algorithms first transform the point clouds into some other space where matching is easier such as the Hough space [11]. Bülow and Birk use the Fast Fourier Transform as well the Mellin Transform to [8] determine the scale and pose difference between a pair of range images. The Normal Distributions Transform (NDT) algorithm transforms one point cloud into a set of normal distributions [6]. The algorithm proceeds similar to the ICP algorithm. A pose estimate is used to transform the second point cloud onto the normal distributions. Since the normal distributions are arranged in a regular grid pattern, correspondences between points and normal distribution can be established in constant time. Using these correspondences an error metric is minimized leading to a new pose estimate.

A common problem with all these algorithms follows from the assumption made at the beginning of this section, i.e., that the registration problem is restricted to only two point clouds. Clearly, registration problems with a larger number of point clouds can be attacked with repeated application of pairwise solutions as well, by matching the first and second, then the second and third scan and so forth. However, no alignment made by any of these algorithms will ever be perfect. Some error, no matter how small, will always remain. This error will add up if we chain our solutions.

Other, more advanced algorithms are needed to tackle larger registration problems. Although

scan matching with more than two scans is more challenging, it also offers some opportunities. A point cloud overlapping with more than a single scan will generally have more evidence towards its position in the global system than otherwise. The problem can be viewed as a graph optimization problem. Each scan position is a node in the graph. An overlap between two point clouds is represented as an edge between the corresponding nodes. A series of edges, with each edge occuring only once, that connects a node to itself creates a loop. This is called loop-closure. In robotics this usually occurs when some agent returns to a location that it has previously visited. Such an event adds information, which is used to correct the errors that have added up in the previous chain of pairwise registrations.

The simultaneous localization and mapping (SLAM) problem is closely related to the registration problem. SLAM refers to the difficulty an autonomous mobile agent faces in an unknown environment. The agent senses its surroundings and must determine two things at the same time. It must create a map of the environment and find its own position and orientation therein. If the robot is equipped with a 3D range sensor, then being able to correctly register a point cloud to one or a set of other point clouds is equivalent to solving the SLAM problem.

2.2.1 Globally Consistent 3D Mapping with Scan Matching

The following paper is the result of early work on solving the SLAM problem. The paper describes one of the first algorithms that solves the problem for applications with 3D range sensors computing full 6 DOF poses. Before this, there were approaches intended for planar environments with 2D scans and 3 DOF poses. The described algorithm was developed as an extension to one of these algorithms presented by Lu and Milios [47]. It features the mathematical foundation needed for the 6DOF computations which are considerably more challenging than the original 3DOF computations due to the additional two rotational degrees of freedom. The presented method also modifies the original algorithm by iterating the estimation. By continually re-estimating the poses and searching for improved correspondences the algorithm finds a much improved estimate of the global poses. It can thus also be considered a generalization of the ICP algorithm to three or more scans.

The methods presented in this paper are the foundation for some of the later work done on mobile mapping and will be expanded upon in the next section.

Dorit Borrmann, Jan Elseberg, Kai Lingemann,
Andreas Nüchter, Joachim Hertzberg

University of Osnabrück
Institut of Computer Science, Knowledge-Based Systems Research Group
Albrechtstr. 28, D-49069 Osnabrück, Germany

Globally Consistent 3D Mapping with Scan Matching

Abstract

A globally consistent solution to the simultaneous localization and mapping (SLAM) problem in 2D with three degrees of freedom (DoF) poses was presented by Lu and Milios [20]. To create maps suitable for natural environments it is however necessary to consider the 6 DoF poses case, namely the three Cartesian coordinates and the roll, pitch and yaw angles. This article describes the extension of the proposed algorithm to deal with these additional DoF and the resulting non-linearities. Simplifications using Taylor expansion and Cholesky decomposition yield a fast application that handles the massive amount of 3D data and the computational requirements due to the 6 DoF. Our experiments demonstrate the functionality of estimating the exact poses and their covariances in all 6 DoF, leading to a globally consistent map. The correspondences between scans are found automatically by use of a simple distance heuristic.

Key words: Simultaneous Localization and Mapping (SLAM), 6D SLAM, GraphSLAM, scan matching.

1 Introduction

A globally consistent representation of a robot's environment is crucial for basic robot tasks such as localization and navigation. Equipped with a laser scanner, many mobile systems gather information about their local environments. These local representations have to be matched to build a global map. Iterative application of pairwise matching algorithms leads to inconsistencies

Email address: nuechter@informatik.uni-osnabrueck.de (Andreas Nüchter, Joachim Hertzberg).

due to errors in laser scans and the matching procedures itself. To avoid these problems, global matching algorithms are needed, taking correspondences between all scans into account. The common methods for merging all scans are based on probabilistic information. Lu and Milios presented a solution using a network of relations between laser scan poses. A single linear equation system, built from all error measurements, yields optimal estimations for all poses. Limited to 2D laser scans, this approach does not fulfill the requirements for building a correct map of outdoor environments. In this article we extend the linear estimation algorithm to work with 3D scans and 6 DoF. To run the algorithm automatically, the network is built using a distance heuristic. Furthermore, we integrate the global optimization into a system consisting of odometry extrapolation and the well-known iterative closest point algorithm (ICP). After describing the problem formulation and the algorithm we document its functionality by experimental results.

2 Related Work

2.1 Categorization of Robotic Mapping Algorithms

One way to categorize mapping algorithms is by the map type. The map can be topological or metrical. Metrical maps represent quantitive distances of the environment. These maps can either be 2D, usually an upright projection, or 3D, i.e., a volumetric environment map. Furthermore, SLAM approaches can be classified by the number of DoF of the robot pose. A 3D pose estimate contains the (x, y)-coordinate and a rotation θ, whereas a 6D pose estimate consideres all DoF a rigid mobile robot can have, i.e., the (x, y, z)-coordinate and the roll, yaw and pitch angles.

In the literature, three different techniques are used for generating 3D maps: First, a planar localization method combined with a 3D sensor; second, a precise 6D pose esimate combined with the data from a 2D sensor; and third, a 3D sensor with a 6D localization method. Tab. 1 summarizes these mapping techniques (black) in comparison with planar 2D mapping (grey). In this paper, we focus on 3D data and 6D localization, hence on 6D SLAM.

Table 1
Overview of the dimensionality of SLAM approaches. Grey: 2D maps. Black: 3D maps.

Sensor data	Dimensionality of pose representation: 3D	6D
2D	**Planar 2D mapping (2.2)** 2D mapping of planar sonar and laser scans.	**Slice-wise 6D SLAM (2.4)** 3D mapping using a precise localization, considering the x,y,z-position and the roll yaw and pitch angle.
3D	**Planar 3D mapping (2.3)** 3D mapping using a planar localization method and, e.g., an upward looking laser scanner or 3D scanner.	**Full 6D SLAM (2.5)** 3D mapping using 3D laser scanners or (stereo) cameras with pose estimates calculated from the sensor data.

2.2 Planar 2D Mapping

State of the art for planar 2D metric maps are probabilistic methods, where the robot has probabilistic motion and uncertain perception models. By integrating these two distributions with a Bayes filter, e.g., Kalman or particle filter, it is possible to localize the robot. Mapping is often an extension to this estimation problem. Beside the robot pose, positions of landmarks are estimated. Closed loops, i.e., a second encounter of a previously visited area of the environment, play a special role here. Once detected, they enable the algorithms to bound the error by deforming the already mapped area such that a topologically consistent model is created. However, there is no guarantee for a correct model. Several strategies exist for solving SLAM. Thrun reviews in [29] existing techniques, i.e., maximum likelihood estimation [12], expectation maximization [10, 30], extended Kalman filter [9] or (sparse extended) information filter [33]. In addition to these methods, FastSLAM [32], which approximates the posterior probabilities, i.e., robot poses, by particles, and the method of Lu/Milios on the basis of IDC scan matching [20], play an important role in 2D.

2.3 Planar 3D Mapping.

Instead of using 3D scanners, which yield consistent 3D scans in the first place, some groups have attempted to build 3D volumetric representations of environments by translating 2D laser range finders. Thrun et al. [32], Früh et al. [13] and Zhao et al. [41] use two 2D laser scanners for acquiring 3D data. One scanner is mounted horizontally, the other vertically. The latter one grabs a vertical scan line which is transformed into 3D points based on the current

3D robot pose. Since the vertical scanner is not able to scan sides of objects, Zhao et al. use two additional, vertically mounted 2D scanners, shifted by 45° to reduce occlusions [41]. The horizontal scanner is used to compute the 3D robot pose. The precision of 3D data points depends, besides on the precision of the scanner, critically on that pose estimation.

Recently, different groups employ rotating SICK scanners for acquiring 3D data [18, 28, 39]. Wulf et al. let the scanner rotate around the vertical axis. They acquire 3D data while moving, thus the quality of the resulting map crucially depends on the pose estimate that is given by inertial sensors, i.e., gyros [39]. In addition, their SLAM algorithms do not consider all 6 DoF.

2.4 Slice-wise 6D SLAM.

Local 3D maps built by translated 2D laser scanners and 6D pose estimates are often used for mobile robot navigation. A well-known example is the grand challenge, where the Stanford racing team used this technique for high speed terrain classification [34].

2.5 Full 6D SLAM.

A few other groups use highly accurate, yet somewhat immobile 3D laser scanners [1, 14, 27]. The RESOLV project aimed at modeling interiors for virtual reality and tele-presence [27]. They used a RIEGL laser range finder on robots and the ICP algorithm for scan matching [5]. The AVENUE project develops a robot for modeling urban environments [1], using a CYRAX scanner and a feature-based scan matching approach for registering the 3D scans. However, in their recent work they do not use data of the laser scanner in the robot control architecture for localization [14]. The group of M. Hebert has reconstructed environments using the Zoller+Fröhlich laser scanner and aims to build 3D models without initial position estimates, i.e., without odometry information [15]. Magnusson and Duckett proposed a 3D scan alignment method that – in contrast to the previously mentioned research groups – does not use ICP algorithm, but the normal distribution transform instead [21].

In principle, the probabilistic methods from planar 2D mapping are extendable to 3D mapping with 6D pose estimates. However, no reliable feature extraction nor a strategy for reducing the computational costs of multi hypothesis tracking, e.g., FastSLAM, that grows exponentially with the DoF, has been published to our knowledge.

2.6 Recent Trends

A recent trend in SLAM research is to apply probabilistic methods to 3D mapping. Katz et al. use a probabilistic notion of ICP scan matching [17]. Weingarten et al. [37] and Cole et al. [8] use an extended Kalman filter on the mapping problem. In the present paper, we extend this state of the art by a GraphSLAM method. A similar approach was used in [35]. However, their algorithm is not practical due to the reported computational requirements. Without considering any sensor data such as laser scans, Olson et al. create globally consistent maps by minimizing the global non-linear constraint network on a set of poses [24]. Triebel et al. apply this approach to their multi-level surface maps [36]. Furthermore, Frese presented an extension of his treemap SLAM algorithm to 6 DoF [11].

2.7 Globally Consistent Range Image Alignment

Besides the robotic community, computer vision researchers are interested in consistent alignment methods. Chen and Medoni [7] aimed at globally consistent range image alignment when introducing an incremental matching method, i.e., all new scans are registered against the so-called metascan, which is the union of the previously acquired and registered scans. This method does not spread out the error and is order-dependent.

Bergevin et al. [4], Benjemaa and Schmitt [2,3], and Pulli [25] present iterative approaches. Based on networks representing overlapping parts of images, they use the ICP algorithm for computing transformations that are applied after all correspondences between all views have been found. However, the focus of reseach is mainly 3D modelling of small objects using a stationary 3D scanner and a turn table; therefore, the used networks consist mainly of one loop [25]. A probabilistic approach was proposed by Williams et al. [38], where each scan point is assigned a Gaussian distribution in order to model the statistical errors made by laser scanners. This causes high computation time due to the large amount of data in practice. Krishnan et al. [19] presented a global registration algorithm that minimizes the global error function by optimization on the manifold of 3D rotation matrices.

3 Algorithm Overview

In the following the extension to 6 DoF data of the global consistent scan matching approach by Lu and Milios is described. The scan matching process

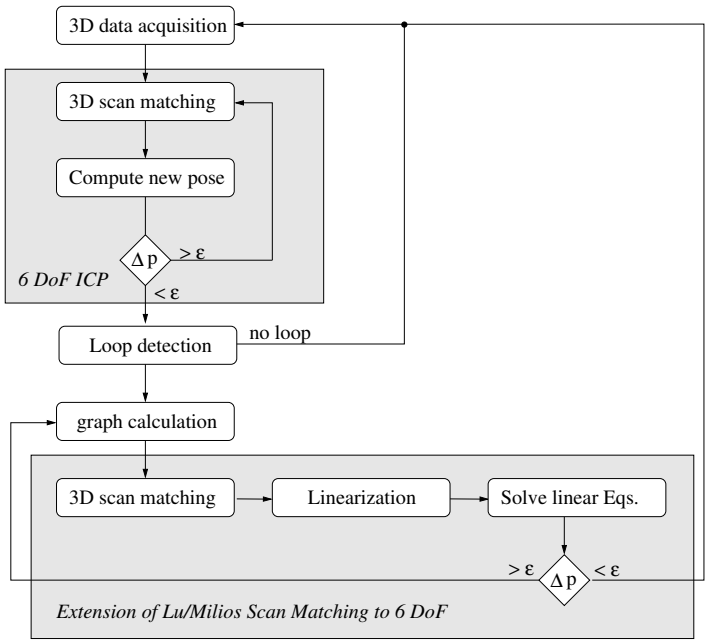


Fig. 1. The interaction of ICP and LUM in the overall algorithm. The threshold ϵ used to determine if a scan changed its pose is the same in both algorithm parts.

is outlined in Fig. 1.

All scans are registered sequentially using the ICP algorithm until convergence. To this end, the odometry of newly acquired scans is extrapolated to 6 DoF using registration matrices of previously registered scans. We are using a left-handed coordinate system, i.e., the y coordinate represents elevation. Then, the change of the robot pose $\Delta\mathbf{P}$, given the odometry information $(x_n^{\text{odo}}, z_n^{\text{odo}}, \theta_{y,n}^{\text{odo}})$, $(x_{n+1}^{\text{odo}}, z_{n+1}^{\text{odo}}, \theta_{y,n+1}^{\text{odo}})$ and the registration matrix $\mathbf{R}(\theta_{x,n}, \theta_{y,n}, \theta_{z,n})$, is calculated by solving:

$$\begin{pmatrix} x_{n+1}^{\text{odo}} \\ 0 \\ z_{n+1}^{\text{odo}} \\ 0 \\ \theta_{y,n+1}^{\text{odo}} \\ 0 \end{pmatrix} = \begin{pmatrix} x_n^{\text{odo}} \\ 0 \\ z_n^{\text{odo}} \\ 0 \\ \theta_{y,n}^{\text{odo}} \\ 0 \end{pmatrix} + \left(\begin{array}{c|c} \mathbf{R}(\theta_{x,n}, \theta_{y,n}, \theta_{z,n}) & \mathbf{0} \\ \hline \mathbf{0} & I_3 \end{array}\right) \cdot \underbrace{\begin{pmatrix} \Delta x_{n+1} \\ \Delta y_{n+1} \\ \Delta z_{n+1} \\ \Delta\theta_{x,n+1} \\ \Delta\theta_{y,n+1} \\ \Delta\theta_{z,n+1} \end{pmatrix}}_{\Delta\mathbf{P}}.$$

Therefore, calculating $\Delta\mathbf{P}$ requires a matrix inversion. Finally, the 6D pose $\mathbf{P}_{n+1}$ is calculated by

$$\mathbf{P}_{n+1} = \Delta\mathbf{P} \cdot \mathbf{P}_n$$

using the poses' matrix representations.

Once a loop is detected, i.e., the distance between the poses of two scans falls below a treshold, the adaption of Lu/Milios style scan matching described in section 6 is applied. For each iteration, a network of pose relations is built automatically. From the corresponding scans, a linear equation system representing distance measurements is built and solved, resulting in optimized pose estimations. When the sum of all distances is below a treshold, new data is acquired and registered using ICP.

4 3D Scan Matching

We use the well-known Iterative Closest Points (ICP) algorithm [5] to calculate the transformation while the robot is acquiring a sequence of 3D scans. ICP calculates the point correspondences iteratively. In each iteration step, the algorithm selects the closest points as correspondences and calculates the transformation $(\mathbf{R}, \mathbf{t})$ for minimizing the equation

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t})\right|\right|^2,$$

where N_m and N_d, are the numbers of points in the model set M or data set D, respectively, and $w_{j,i}$ are the weights for a point match. The weights are assigned as follows: $w_{i,j} = 1$, if $\mathbf{m}_i$ is the closest point to $\mathbf{d}_j$ within a close limit, $w_{i,j} = 0$ otherwise. The assumption is that in the last iteration the point correspondences are correct. In each iteration, the transformation is calculated by the quaternion based method of Horn [16].

To digitalize environments without occlusions, multiple 3D scans have to be registered. Consider a robot travelling along a path, and traversing $n+1$ poses $V_0, \ldots, V_n$. A straightforward method for aligning several 3D scans taken from $V_0, \ldots, V_n$ is *pairwise ICP*, i.e., matching the scan taken from pose V_1 against the scan from pose V_0, matching the scan taken from V_2 against the scan from pose V_1, and so on.

5 Loop Closing

Pairwise ICP improves the robot pose estimates, but registration errors sum up. SLAM algorithms use loop closing to bound this error. If two estimated robot poses V_i and V_j are close enough, i.e., their distance falls below a threshold (here: 5 meters), we assume these scans overlap and are matchable. To a

graph, initially containing the sequence of all poses $(V_0, V_1), (V_1, V_2), \ldots,$ (V_{n-1}, V_n), the edge (V_i,V_j) is added.

While processing the scans with *pairwise ICP*, we detect closed loops using this simple distance criterion. Once detected, a 6D graph optimization algorithm for global relaxation based on the method of Lu and Milios [20] is employed, namely, Lu and Milios style SLAM (LUM). We extended this variant of GraphSLAM to 6 DoF, as described in the next section.

If this simple loop closing strategy fails, the resulting map is incorrect. We accept this drawback, since a multi hypothesis approach with 6 DoF is currently not tractable. In [40] we give a quantitative performance evaluation and an in depth analyzation of the loop closing method.

6 Global Relaxation

6.1 Problem Formulation

Consider a robot traveling along a path, and traversing the $n + 1$ poses $V_0, \ldots, V_n$. At each pose V_i, the robot stops to take a laser scan of its environment. By matching two scans made at two different poses, we acquire a set of neighbor relations between these poses. In the resulting network, nodes represent poses, and edges neighbor relations between them. Given such a network with $n+1$ graph nodes $X_0, \ldots, X_n$ and the directed edges $D_{i,j}$, we want to estimate optimally all poses to build a consistent map of the environment. For simplification, the measurement equation is assumed to be linear:

$$D_{i,j} = X_i - X_j.$$

The observation $\bar{D}_{i,j}$ of the true underlying difference is modeled as $\bar{D}_{i,j} = D_{i,j} + \Delta D_{i,j}$ where $\Delta D_{i,j}$ is a Gaussian distributed error with zero mean and a covariance matrix $C_{i,j}$, that is assumed to be known.

Maximum likelihood estimation is used to approximate the optimal poses X_i. Under the assumption that all errors in the observations are Gaussian and independently distributed, maximizing the probability of all $D_{i,j}$, given their observations $\bar{D}_{i,j}$, is equivalent to minimizing the following Mahalanobis distance:

$$\mathbf{W} = \sum_{(i,j)} (D_{i,j} - \bar{D}_{i,j})^T C_{i,j}^{-1} (D_{i,j} - \bar{D}_{i,j}). \quad (1)$$

6.2 Solution as given by Lu and Milios

We consider the simple linear case of the estimation problem. Without loss of generality we assume that the network is fully connected, i,e, each pair of nodes X_i, X_j is connected by a link $D_{i,j}$. In the case of a missing link $D_{i,j}$ we set the corresponding $C_{i,j}^{-1}$ to 0. Eq. (1) unfolds:

$$\mathbf{W} = \sum_{(0 \le i < j \le n)} (X_i - X_j - \bar{D}_{i,j})^T C_{i,j}^{-1} (X_i - X_j - \bar{D}_{i,j}). \tag{2}$$

To minimize equation (2), a coordinate system is defined by setting one node as a reference point. Setting $X_0 = (0, 0, 0)$, the n free nodes $X_1, \ldots, X_n$ denote the poses relative to X_0.

Using the signed incidence matrix $\mathbf{H}$, the concatenated measurement equation $\mathbf{D}$ is written as

$$\mathbf{D} = \mathbf{H}\mathbf{X},$$

with the concatenation $\mathbf{X}$ of X_1 to X_n. The Mahalanobis distance equation (1) can be written as:

$$\mathbf{W} = (\bar{\mathbf{D}} - \mathbf{H}\mathbf{X})^T \mathbf{C}^{-1} (\bar{\mathbf{D}} - \mathbf{H}\mathbf{X}).$$

The concatenation of all observations $\bar{D}_{i,j}$ forms the vector $\bar{\mathbf{D}}$, while $\mathbf{C}$ is a block-diagonal matrix comprised of $C_{i,j}$ as submatrices. The solution $\mathbf{X}$ that minimizes the equation (2), and its covariance $\mathbf{C}_\mathbf{X}$ are given by

$$\begin{aligned} \mathbf{X} &= (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{D}} \\ \mathbf{C}_\mathbf{X} &= (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1}. \end{aligned} \tag{3}$$

The matrix $\mathbf{G} = \mathbf{H}^T \mathbf{C}^{-1} \mathbf{H}$ and the vector $\mathbf{B} = \mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{D}}$ simplify the notation of the solution. $\mathbf{G}$ consists of submatrices

$$G_{i,j} = \begin{cases} \sum_{j=0}^{n} C_{i,j}^{-1} & (i = j) \\ C_{i,j}^{-1} & (i \neq j). \end{cases} \tag{4}$$

The entries of $\mathbf{B}$ are obtained by:

$$B_i = \sum_{\substack{j=0 \\ j \neq i}}^{n} C_{i,j}^{-1} \bar{D}_{i,j}. \tag{5}$$

Solving the linear optimal estimation problem (3) is equivalent to solving the following linear equation system:

$$\mathbf{G}\mathbf{X} = \mathbf{B}. \tag{6}$$

6.3 The Extension to Six Degrees of Freedom

To apply the above solution to mobile robots, it is necessary to linearize the pose difference equation (2). The three DoF case, i.e., $(x, z, \theta)^T$ poses, was solved by Lu and Milios [20]. In addition to pose relations from scan matching, Lu and Milios considered pose relations from odometry. Our algorithm derives relations for 6 DoF poses, i.e., $(x, y, z, \theta_x, \theta_y, \theta_z)^T$, by matching data obtained by a 3D laser range finder. The challenges of the extension were:

The amount of data. A 3D laser range finder scans the environment with a large number of samples. Typical resolutions vary between 23168 and 360500 range points per 3D scan.

Linearization. Linearization of the rotation must regard the 3 DoF. The rotation consists of the three Euler angles $(\theta_x, \theta_y, \theta_z)$, and the multiplication of the corresponding three rotation matrices results in the desired overall rotation. By linearizing the Euler angles, we enforce valid rotation matrices.

Solution space. The additional three DoF result in an exponentially larger solution space. The solution is computationally more complex.

We define a 6D pose relation as follows: Assume that a robot starts at the pose $V_b = (x_b, y_b, z_b, \theta_{x_b}, \theta_{y_b}, \theta_{z_b})^T$ and changes its pose by $D = (x, y, z, \theta_x, \theta_y, \theta_z)^T$ relative to V_b, ending up at $V_a = (x_a, y_a, z_a, \theta_{x_a}, \theta_{y_a}, \theta_{z_a})^T$. The poses V_a and V_b are related by the compounding operation $V_a = V_b \oplus D$. Similarly, a 3D position vector $u = (x_u, y_u, z_u)$ is compounded with the pose V_b by $u' = V_b \oplus u$:

$$
\begin{aligned}
x'_u &= x_b - z_u \sin\theta_{y_b} + \cos\theta_{y_b}(x_u \cos\theta_{z_b} - y_u \sin\theta_{z_b}) \\
y'_u &= y_b + z_u \cos\theta_{y_b} \sin\theta_{x_b} + \cos\theta_{x_b}(y_u \cos\theta_{z_b} + x_u \sin\theta_{z_b}) \\
&\quad + \sin\theta_{x_b} \sin\theta_{y_b}(x_u \cos\theta_{z_b} - y_u \sin\theta_{z_b}) \\
z'_u &= z_b - \sin\theta_{x_b}(y_u \cos\theta_{z_b} + x_u \sin\theta_{z_b}) \\
&\quad + \cos\theta_{x_b}(z_u \cos\theta_{y_b} + \sin\theta_{y_b}(x_u \cos\theta_{z_b} - y_u \sin\theta_{z_b}))
\end{aligned}
$$

This operation is used to transform a non-oriented point (from the scanner data) from its local to the global coordinate system.

Scan matching computes a set of m corresponding point pairs u_k^a, u_k^b between two scans, each representing a single physical point. The positional error made

by identifying these two points in different scans, is described by:

$$F_{ab}(V_a, V_b) = \sum_{k=1}^{m} \left\| V_a \oplus u_k^a - V_b \oplus u_k^b \right\|^2 \tag{7}$$

$$= \sum_{k=1}^{m} \left\| (V_a \ominus V_b) \oplus u_k^a - u_k^b \right\|^2. \tag{8}$$

Based on these m point pairs, the algorithm computes the matrices $\bar{D}_{i,j}$ and $C_{i,j}$ for solving Eq. (1). $\bar{D}_{i,j}$ is derived as follows:

Let $\bar{V}_a = (\bar{x}_a, \bar{y}_a, \bar{z}_a, \bar{\theta}_{x_a}, \bar{\theta}_{y_a}, \bar{\theta}_{z_a})$ and $\bar{V}_b = (\bar{x}_b, \bar{y}_b, \bar{z}_b, \bar{\theta}_{x_b}, \bar{\theta}_{y_b}, \bar{\theta}_{z_b})$ be close estimates of V_a and V_b. If the global coordinates of a pair of matching points $u_k = (x_k, y_k, z_k)$, then (u_k^a, u_k^b) fulfill the equation

$$u_k \approx V_a \oplus u_k^a \approx V_b \oplus u_k^b.$$

For small errors $\Delta V_a = \bar{V}_a - V_a$ and $\Delta V_b = \bar{V}_b - V_b$, a Taylor expansion leads to:

$$\begin{aligned}
\Delta Z_k &= V_a \oplus u_k^a - V_b \oplus u_k^b := F_k(V_a, V_b) \\
&\approx F_k(\bar{V}_a, \bar{V}_b) - [\nabla_{\bar{V}_a}(F_k(\bar{V}_a, \bar{V}_b))\Delta V_a \\
&\quad - \nabla_{\bar{V}_b}(F_k(\bar{V}_a, \bar{V}_b))\Delta V_b] \\
&= \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b - [\nabla_{\bar{V}_a}(\bar{V}_a \oplus u_k^a)\Delta V_a \\
&\quad - \nabla_{\bar{V}_b}(\bar{V}_b \oplus u_k^b)\Delta V_b]
\end{aligned} \tag{9}$$

where $\nabla_{\bar{V}_a}(F_k(\bar{V}_a, \bar{V}_b))$ is the gradient of the pose compounding operation. By matrix decomposition

$$\begin{aligned}
M_k H_a &= \nabla_{\bar{V}_a}(F_k(\bar{V}_a, \bar{V}_b)) \\
M_k H_b &= \nabla_{\bar{V}_b}(F_k(\bar{V}_a, \bar{V}_b)),
\end{aligned}$$

Eq. (9) simplifies to:

$$\begin{aligned}
\Delta Z_k &\approx \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b - M_k[H_a \Delta V_a - H_b \Delta V_b] \\
&= \bar{Z}_k - M_k D
\end{aligned}$$

with

$$\begin{aligned}
\bar{Z}_k &= \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b \\
D &= (H_a \Delta V_a - H_b \Delta V_b) \\
M_k &= \begin{pmatrix} 1 & 0 & 0 & 0 & -y_k & -z_k \\ 0 & 1 & 0 & z_k & x_k & 0 \\ 0 & 0 & 1 & -y_k & 0 & x_k \end{pmatrix}
\end{aligned} \tag{10}$$

$$H_a = \begin{pmatrix} *6c1 & 0 & 0 & 0 & \bar{z}_a\cos(\bar{\theta}_{x_a})+\bar{y}_a\sin(\bar{\theta}_{x_a}) & \bar{y}_a\cos(\bar{\theta}_{x_a})\cos(\bar{\theta}_{y_a})-\bar{z}_a\cos(\bar{\theta}_{y_a})\sin(\bar{\theta}_{x_a}) \\ 0 & 1 & 0 & -\bar{z}_a & -\bar{x}_a\sin(\bar{\theta}_{x_a}) & -\bar{x}_a\cos(\bar{\theta}_{x_a})\cos(\bar{\theta}_{y_a})-\bar{z}_a\sin(\bar{\theta}_{y_a}) \\ 0 & 0 & 1 & \bar{y}_a & -\bar{x}_a\cos(\bar{\theta}_{x_a}) & \bar{x}_a\cos(\bar{\theta}_{y_a})\sin(\bar{\theta}_{x_a})+\bar{y}_a\sin(\bar{\theta}_{y_a}) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_{y_a}) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_{x_a}) & \cos(\bar{\theta}_{x_a})\cos(\bar{\theta}_{y_a}) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_{x_a}) & -\cos(\bar{\theta}_{y_a})\sin(\bar{\theta}_{x_a}) \end{pmatrix}.$$

H_b is given analogously. This matrix decomposition and the derivation of H_a, H_b is the crucial step in extending Lu and Milios style SLAM to 6 DoF.

D as defined by Eq. (10) is the new linearized measurement equation. To calculate both $\bar{D}$ and C_D, the equation (8) is rewritten in matrix form

$$F_{ab}(D) \approx (\mathbf{Z} - \mathbf{M}D)^T(\mathbf{Z} - \mathbf{M}D).$$

$\mathbf{M}$ is the concatenated matrix consisting of all M_k's, and $\mathbf{Z}$ the concatenated vector consisting of all Z_k's. The vector $\bar{D}$ that minimizes F_{ab} is given by

$$\bar{D} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}. \tag{11}$$

Since minimizing F_{ab} constitutes a least squares linear regression, we model the Gaussian distribution of the solution with mean $\bar{D}$ and standard covariance estimation

$$C_D = s^2(\mathbf{M}^T\mathbf{M}). \tag{12}$$

s^2 is the unbiased estimate of the covariance of the identically, independently distributed errors of Z_k, given by:

$$s^2 = (\mathbf{Z} - \mathbf{M}\bar{D})^T(\mathbf{Z} - \mathbf{M}\bar{D})/(2m-3) = \frac{F_{ab}(\bar{D})}{2m-3}.$$

The error term W_{ab} corresponding to our pose relation is defined by:

$$W_{ab} = (\bar{D} - D)^T C_D^{-1}(\bar{D} - D).$$

6.4 Transforming the Solution

Solving the linear equation (6) leads to an optimal estimate of the new measurement equation of D (Eq. (10)). To yield an optimal estimation of the robot poses, it is necessary to transform D. By this optimal estimation, a set of solutions $X_i = H_i\Delta V_i$ is computed, each corresponding to a node in the network. Assuming that the reference pose $V_0 = 0$, the pose V_i and its covariance C_i

Algorithm 1 Optimal estimation algorithm

(1) Compute the point correspondences u_k^a, u_k^b.
(2) For any link (i, j) in the given graph compute the measurement vector $\bar{D}_{ij}$ by Eq. (11) and its covariance C_{ij} by Eq. (12).
(3) From all $\bar{D}_{ij}$ and C_{ij} form the linear system $\mathbf{GX} = \mathbf{B}$, with $\mathbf{G}$ and $\mathbf{B}$ as given in Eq. (4) and (5) respectively.
(4) Solve for $\mathbf{X}$
(5) Update the poses and their covariances, as explained in Section 6.4.

are updated by:

$$V_i = \bar{V}_i - H_i^{-1} X_i,$$
$$C_i = (H_i^{-1}) C_i^X (H_i^{-1})^T.$$

If V_0 is nonzero, the solutions have to be transformed by:

$$V_i' = V_0 \oplus V_i$$
$$C_i' = K_0 C_i K_0^T$$

where

$$K_0 = \begin{pmatrix} R_{\theta_{x_0}, \theta_{y_0}, \theta_{z_0}} & 0 \\ 0 & I_3 \end{pmatrix}$$

with a rotation matrix $R_{\theta_{x_0}, \theta_{y_0}, \theta_{z_0}}$.

6.5 The Algorithm

The optimal estimation algorithm is given as Algorithm 1. Iterative execution of Algorithm 1 yields a successive improvement of the global pose estimation. Step 3 is sped up by component-wise computation of $\mathbf{G}$ and $\mathbf{B}$. The components $C_{i,j}^{-1} = (\mathbf{M}^T\mathbf{M})/s^2$ and $C_{i,j}^{-1}\bar{D}_{i,j} = (\mathbf{M}^T\mathbf{Z})/s^2$ are expanded into simple summations, as shown in the appendix B. The most expensive operation is solving the linear equation system $\mathbf{GX} = \mathbf{B}$. Since $\mathbf{G}$ is a positive definite, symmetric $6n \times 6n$ matrix, this is done by Cholesky decomposition in $\mathcal{O}(n^3)$.

6.6 Performace Issues

The large amount of 3D data to be processed makes computing time an issue in globally consistent range scan matching. Our algorithm again benefits from the network structure. Each scan has to be aligned only to few neighbors in the

graph. Compared to ICP metascan matching, LUM becomes more advantageous with increasing number of scans n. Our SLAM algorithm spends $\mathcal{O}(n^3)$ time on matrix computation. The matrices $\mathbf{B}$ and $\mathbf{G}$ are filled efficiently using simple additions (cf. appendix B). However, calculating the corresponding points for a link needs $\mathcal{O}(N \log N)$, using standard nearest neighbor search methods, namely k-d trees. N denotes the number of points per 3D scan, $n \ll N$. In all experiments the most computing time was spent in step 2 of algorithm 1, e.g., $n < 13$, $N < 300000$ or $n < 468$, $N < 18000$, respectively (cf. Sec. 7). Due to these performace issues, we presented several speed-ups for closest point computation in the scan matching context, i.e., approximate k-d tree search and cached k-d tree search [22, 23].

6.7 Invertibility of $\mathbf{G}$

The proposed algorithm depends on the invertibility of the matrix $\mathbf{G}$, which is the case if:

(1) All covariances are positive or negative definite, and:
(2) The pose graph is connected, i.e., there exist no two separate subgraphs.

The second condition is trivially met in practice, since all consecutive poses are linked. The inductive proof of the first condition over the number of nodes in the graph is given in appendix A.

7 Experiments and Results

The proposed algorithm has been tested in various experiments. In [6] we show by evaluating 2D laser range scans that the extension to three dimensions did not decrease the functionality of the algorithm in areas where it did work sufficiently well before. Furthermore 3D data was succesfully aligned to ground truth when matching data from a planar indoor environment. Additionally, full functionality in all 6 DoF was shown by use of data obtained from non-planar outdoor environments.

In this article, we first present an evaluation using high resolution 3D scans. The resulting poses are compared with ground truth. Second, highly connected data from a 3D environment is matched with an active loop closing algorithm. This leads to a network with many links that helps to diminish inconsistencies in the resulting map.

Table 2
Position and orientation errors in the Horn data set. The remaining error after the application of our algorithm and the initial error are given. The initial error was choosen randomly, i.e., we added noise to the horizontal axis in postion and to the vertical axis in orientation.

3D scan No.	initial error [m] (Eucl. distance)	position error [m] after registration	initial error [deg] $(\theta_x, \theta_y, \theta_z)$	rotation error [deg] after registration
1	0	0	(0.0, 0.0, 0.0)	(0.0, 0.0, 0.0)
2	0.780	0.017	(0.0, 4.0, 0.0)	(-0.0221, 0.0177, -0.0116)
3	1.965	0.024	(0.0, 4.0, 0.0)	(-0.0242, 0.0257, -0.0236)
4	1.764	0.029	(0.0, 2.0, 0.0)	(-0.0242, 0.0228, -0.0200)
5	1.222	0.037	(0.0, 5.0, 0.0)	(-0.0266, 0.0298, -0.0339)
6	0.859	0.059	(0.0, 1.0, 0.0)	(-0.0297, 0.0183, 0.0039)
7	2.151	0.052	(0.0, 2.0, 0.0)	(0.0061, 0.0195, -0.0269)
8	0.451	0.053	(0.0, 1.0, 0.0)	(-0.0416, 0.0173, 0.0076)
9	1.023	0.069	(0.0, 3.0, 0.0)	(-0.0690, 0.0279, 0.0170)
10	2.039	0.082	(0.0, 1.0, 0.0)	(-0.0690, 0.0165, 0.0190)
11	1.654	0.059	(0.0, 5.0, 0.0)	(-0.0988, 0.0936, 0.0040)
12	1.340	0.033	(0.0, 2.0, 0.0)	(0.0030, 0.0196, 0.0410)
13	1.195	0.017	(0.0, 5.0, 0.0)	(0.0022, 0.0593, 0.0510)

7.1 Registration of High Resolution Outdoor 3D Scans

For this experiment we used a data set from the main square in Horn (Austria), consisting of 13 laser scans. Each scan is composed of 240000 to 300000 points. Scan matching with reduced points took 19 minutes (6 minutes for ICP and 13 minutes for LUM) until convergence, e.g. no scan was moved more than 0.5 cm in one iteration.

Parts of the resulting map can be seen in Fig. 2 (see `http://kos.informatik.uni-osnabrueck.de/download/Riegl/index.html` for the whole map). From top to bottom, the number of iteration increases, and analogously the precision of the map. Qualitatively, no inconsistencies remain in the bottommost images.

For quantitative results, Tab. 2 shows the drastic reduction of errors compared to ground truth, acquired by manual alignment using retro-reflective targets of known coordinates [26], during scan matching. Pose errors are equally reduced for each laser scan, without any accumulation of errors. This holds not only for position errors but also for rotation errors, which pose the most significant challenge in 6D SLAM.

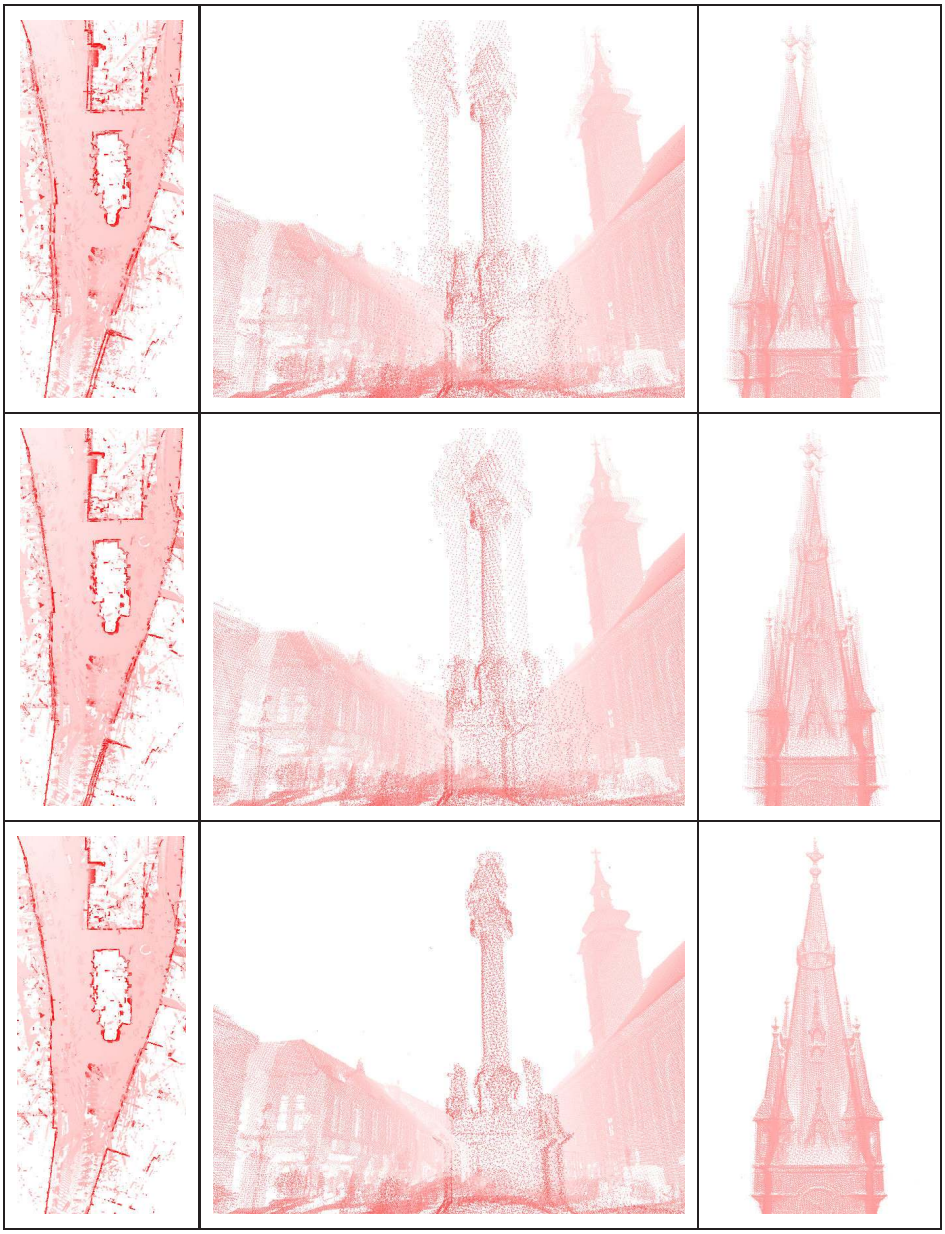


Fig. 2. Map improvement during LUM. Main square in Horn (Austria). Data provided by courtesy of RIEGL LMS GmbH [26]. Progress after 1 (top), 30 (middle), 200 (bottom) iterations. Left: Top view, Middle: Monument in the center of the main square, Right: Church spire.

7.2 Registration with Dynamic Network Construction

Detecting loops in sets of data helps build globally consistent maps, because it facilitates distributing larger errors over all scans. Each pose relation gives

Fig. 3. Photos showing the scene presented in Fig. 2. Data provided by courtesy of RIEGL LMS GmbH [26]. Left: The right part of the middle detailed view. Right: The white-steepled St. Georg church.

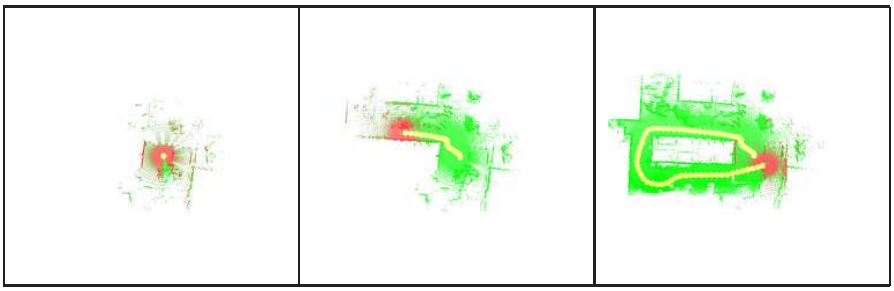


Fig. 4. Sequential ICP registration, from start to first loop detection, in the Hannover data set.

additional information for improving the calculation of optimal pose estimations. After loop detection, we automatically determine the pose network, using simple distance heuristics. By calculating the pose relations dynamically after each iteration of LUM, we obtain optimized poses, leading iteratively to more accurate networks. Scans that converge towards each other result in new pose relations, while connections between diverging scans disappear.

This is demonstrated on a set of 468 laser scans from a roboter run at the Leibniz Universität Hannover while driving a distance od ca. 750 m. Each scan consists of 14000 to 18000 scan points. The convergence limit was set to 0.1 cm movement per pose. Neighbor relations are established between all scans within a distance of less than 7.5 m.

Fig. 4 to 8 show the maps after particular steps of scan matching. First, scans are registered iteratively using the ICP algorithm, generating the maps shown in Fig. 4. Once a loop is detected, LUM is used to achieve global consistency (cf. Fig. 5). Fig. 6 shows the map after LUM at the second, third and forth closed loop and Fig. 8 the final map. While matching the forth loop, the robot

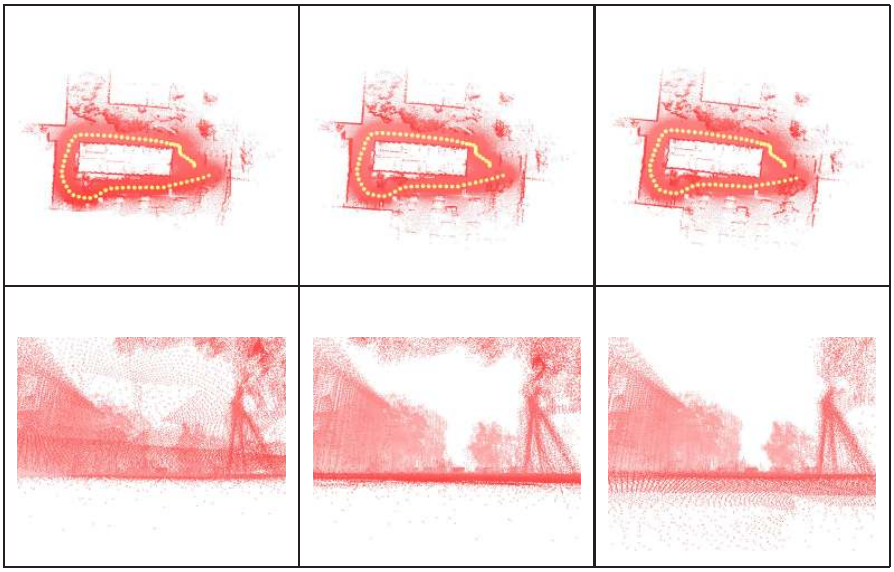


Fig. 5. LUM iterations at first closed loop. Close ups made from the same virtual camera position. After few iterations the leaves of the tree are moved behind the camera position (bottom). Left: 1, Middle: 10, Right: 77 iterations.

path merges through recomputation of the graph, as shown in Fig. 7.

Ground truth for this data set is not available, therefore no comparison of our final 3D map to a reference 3D model is possible. Wulf et al. developed a method to benchmark our mapping algorithm using Monte Carlo Localization in 2D reference maps from the German Land Registry office [40]. Using this novel benchmarking method on a similar data set they demonstrated that our algorithm maps the areas where the loops are closed with higher precision than the remaining parts.

8 Summary

One very active research area in robotics is mapping environments by matching point clouds collected by laser scanners. Popular techniques for 3D scan matching are based on minimizing the distances between point pairs detected in two corresponding range scans. Errors in laser scan data and imprecise matching methods lead to accumulated errors in the progress of building large maps, causing inconsistencies in regions where loops are closed.

This paper has presented a technique of matching laser scans globally consistently. Since a global error function is minimized in our approach, it avoids the common problems of sequential matching strategies.

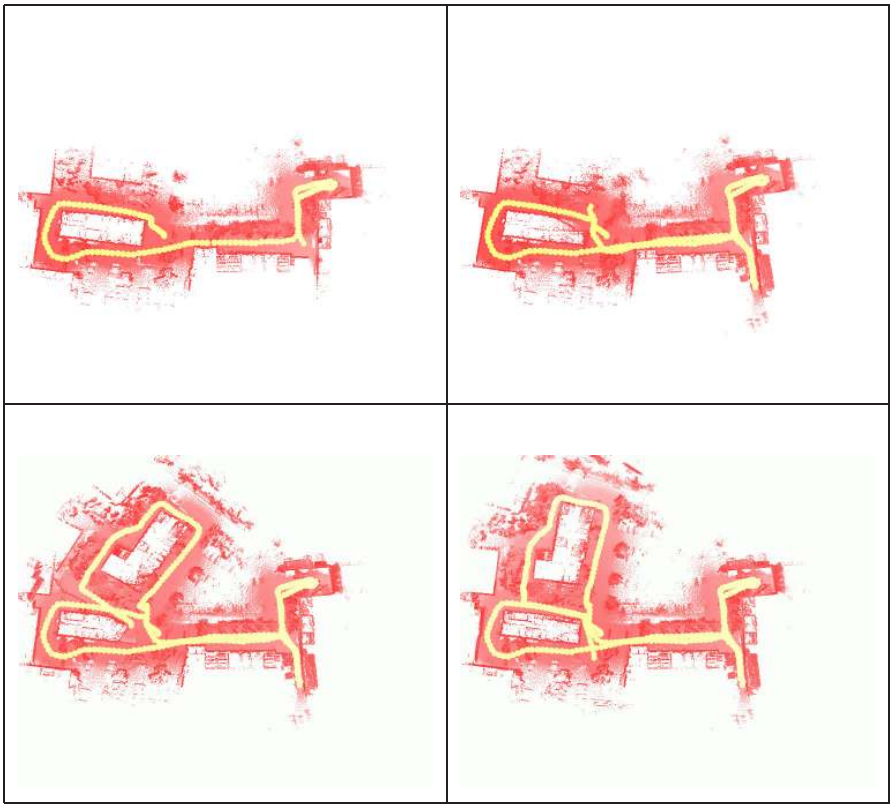


Fig. 6. LUM iterations at second, third and fourth closed loop. Top: 2nd and 3rd loop closing. Bottom: 4th loop closing after 1 iteration and after 20 iterations. Notice the merging of the robot path through graph recomputation in the 4th loop closing! (see Fig. 7 for the corresponding graphs, Fig. 8 for the resulting map)

The algorithm copes with the difficulties posed by the 6 DoF and the large amount of data in a fast and robust manner. The instabilities reported in [31] for 2D scans and 3D poses did not occur in the 3D scan/6D pose case.

9 Acknowledgments

The authors would like to acknowledge Nikolaus Studnicka (RIEGL Laser measurement Systems GmbH, Horn) and Oliver Wulf, Bernardo Wagner (Leibniz Universität Hannover) for providing the data sets. Further thanks to Evangelos E. Milios (Dalhousie University), Sebastian Thrun (Stanford University) and Szymon Rusinkiewicz (Princeton University) for answering questions about GraphSLAM methods.

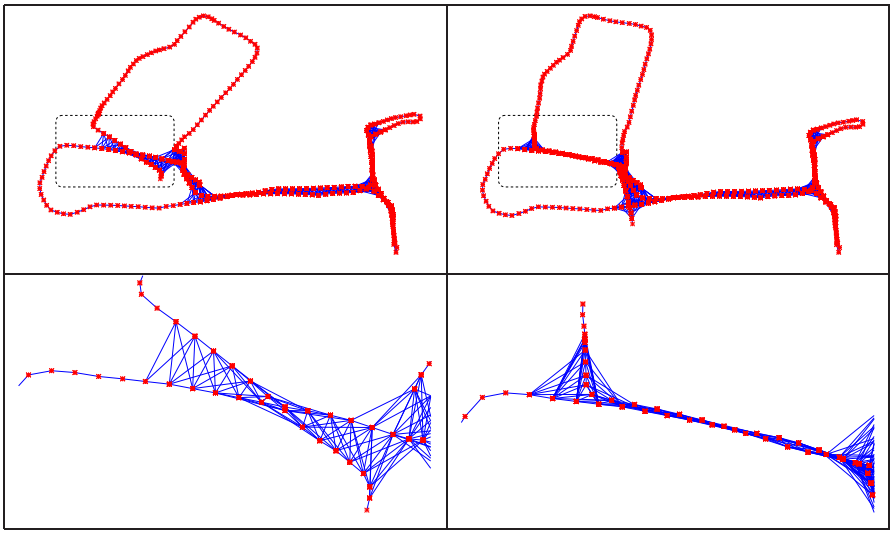


Fig. 7. Top: Graphs corresponding to the two allignments of Fig. 6 (Bottom). Bottom: Zoom into the boxed areas.

References

[1] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. AVENUE: Automated Site Modelling in Urban Environments. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM '01)*, Quebec City, Canada, May 2001.

[2] R. Benjemaa and F. Schmitt. Fast Global Registration of 3D Sampled Surfaces Using a Multi-Z-Buffer Technique. In *Proceedings IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '97)*, Ottawa, Canada, May 1997.

[3] R. Benjemaa and F. Schmitt. A Solution For The Registration Of Multiple 3D Point Sets Using Unit Quaternions. *Computer Vision – ECCV '98*, 2:34 – 50, 1998.

[4] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multi-view registration technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(5):540 – 547, May 1996.

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

[6] D. Borrmann, J. Elseberg, K.Lingemann, A. Nüchter, and J. Hertzberg. Efficient Extension of Lu and Milios SLAM to Six Degrees of Freedom, (in preparation).

[7] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Image Vision Comput.*, 10(3):145–155, 1992.

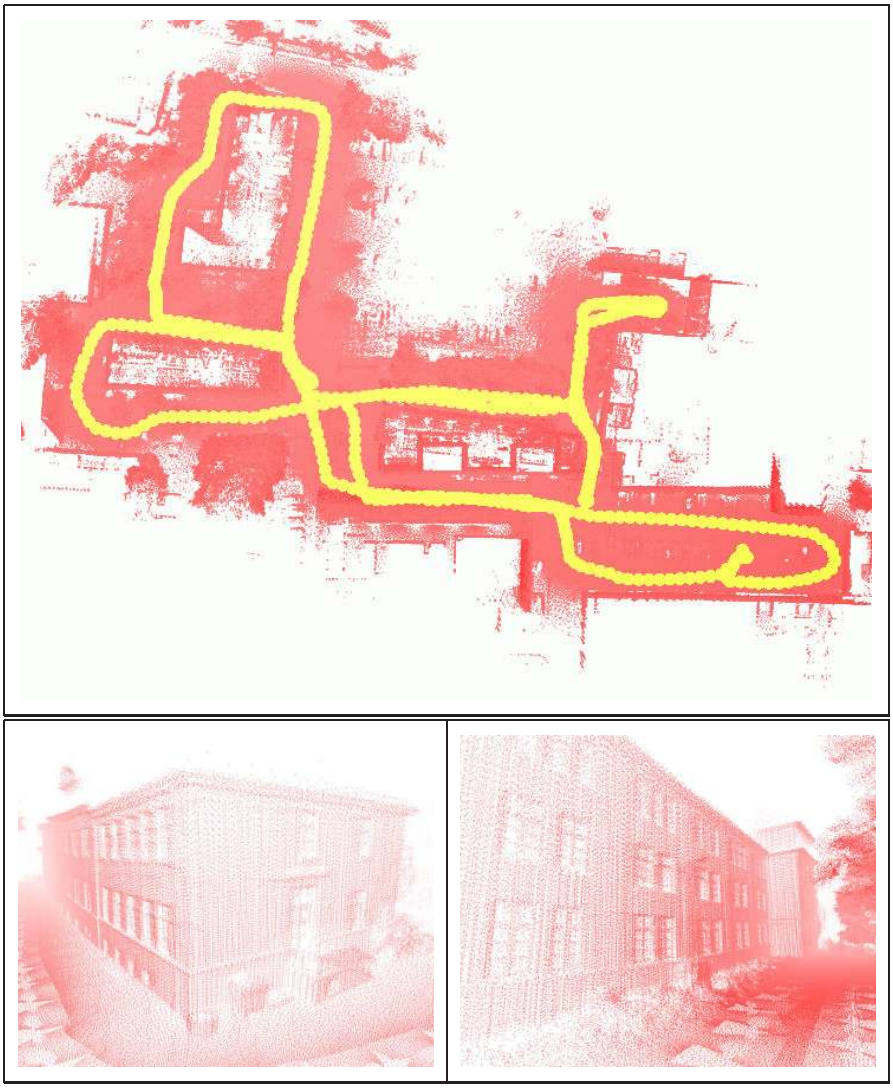


Fig. 8. Final map. Top: Top view including the final trajectory. Bottom: Two details rendered from the Hannover scene.

[8] D. M. Cole and P. M. Newman. Using Laser Range Data for 3D SLAM in Outdoor Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)*, Florida, U.S.A., 2006.

[9] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229 – 241, June 2001.

[10] J. Folkesson and H. Christensen. Outdoor Exploration and SLAM using a Compressed Filter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)*, pages 419–426, Taipei, Taiwan,

September 2003.

[11] U. Frese. Efficient 6-DOF SLAM with Treemap as a Generic Backend. In *Proceedings of the IEEE Internation Conference on Robotics and Automation (ICRA '07)*, Rome, Italy, April 2007.

[12] U. Frese and G. Hirzinger. Simultaneous Localization and Mapping – A Discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17 – 26, Seattle, USA, August 2001.

[13] C. Früh and A. Zakhor. 3D Model Generation for Cities Using Aerial Photographs and Ground Level Laser Scans. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR '01)*, Kauai, Hawaii, USA, December 2001.

[14] A. Georgiev and P. K. Allen. Localization Methods for a Mobile Robot in Urban Environments. *IEEE Transaction on Robotics and Automation (TRO)*, 20(5):851 – 864, October 2004.

[15] M. Hebert, M. Deans, D. Huber, B. Nabbe, and N. Vandapel. Progress in 3–D Mapping and Localization. In *Proceedings of the 9th International Symposium on Intelligent Robotic Systems, (SIRS '01)*, Toulouse, France, July 2001.

[16] B. K. P. Horn. Closed–form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629 – 642, April 1987.

[17] R. Katz, N. Melkumyan, J. Guivant, T. Bailey, J. Nieto, and E. Nebot. Integrated Sensing Framework for 3D Mapping in Outdoor Navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, Beijing, China, October 2006.

[18] P. Kohlhepp, M. Walther, and P. Steinhaus. Schritthaltende 3D-Kartierung und Lokalisierung für mobile Inspektionsroboter. In R. Dillmann, H. Wörn, and T. Gockel, editors, *Proceedings of the Autonome Mobile Systeme 2003, 18. Fachgespraeche*, December 2003.

[19] S. Krishnan, P. Y. Lee, J. B. Moore, and S. Venkatasubramanian. Global Registration of Multiple 3D Point Sets via Optimization on a Manifold. In *Eurographics Symposium on Geometry Processing*, 2000.

[20] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4:333 – 349, April 1997.

[21] M. Magnusson and T. Duckett. A Comparison of 3D Registration Algorithms for Autonomous Underground Mining Vehicles. In *Proceedings of the Second European Conference on Mobile Robotics (ECMR '05)*, pages 86 – 91, Ancona, Italy, September 2005.

[22] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM with Approximate Data Associoation. In *Proceedings of the 12th IEEE International Conference on Advanced Robotics (ICAR '05)*, pages 242 – 249, Seattle, U.S.A., July 2005.

[23] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM for 3D Mapping Outdoor Environments. *Journal of Field Robotics*, (accepted).

[24] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006.

[25] K. Pulli. Multiview Registration for Large Data Sets. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling (3DIM '99)*, pages 160 – 168, Ottawa, Canada, October 1999.

[26] RIEGL Laser Measurement Systems GmbH. `www.riegl.com`.

[27] V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, and D. Hogg. Automated 3D reconstruction of interiors with multiple scan–views. In *Proceedings of SPIE, Electronic Imaging '99, The Society for Imaging Science and Technology/SPIE's 11th Annual Symposium*, San Jose, CA, USA, January 1999.

[28] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. A 3D Laser Range Finder for Autonomous Mobile Robots. In *Proceedings of the 32nd International Symposium on Robotics (ISR '01)*, Seoul, Korea, April 2001.

[29] S. Thrun. Robotic Mapping: A Survey. *CMU-CS-02-111*, February 2002. School of Computer Science - Carnegie Mellon University. Pittsburgh, PA 15213.

[30] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots*, 31(5):1 – 25, October 1997.

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

[32] S. Thrun, D. Fox, and W. Burgard. A Real-Time Algorithm for Mobile Robot Mapping with Application to multi robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 00)*, San Francisco, CA, USA, April 2000.

[33] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. F. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Machine Learning and Autonomous Robots*, 23(7 – 8):693 – 716, July/August 2004.

[34] S. Thrun, M. Montemerlo, and A. Aron. Probabilistic Terrain Analysis For High-Speed Desert Driving. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2006.

[35] R. Triebel and W. Burgard. Improving Simultaneous Localization and Mapping in 3D Using Global Constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI '05)*, 2005.

[36] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[37] J. Weingarten and R. Siegwart. EKF-based 3D SLAM for structured environment reconstruction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pages 2089 – 2094, Edmonton, Alberta Canada, August 2005.

[38] J.A. Williams and M. Bennamoun. Multiple View 3D Registration using Statistical Error Models. In *Vision Modeling and Visualization*, 1999.

[39] O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 4204 – 4209, New Orleans, USA, April 2004.

[40] O. Wulf, A. Nüchter, J. Hertzberg, and B. A. Wagner. Ground Truth Evaluation of Large Urban 6D SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, San Diego, U.S.A., October 2007, (accepted).

[41] H. Zhao and R. Shibasaki. Reconstructing Textured CAD Model of Urban Environment Using Vehicle-Borne Laser Range Scanners and Line Cameras. In *Second International Workshop on Computer Vision System (ICVS '01)*, pages 284 – 295, Vancouver, Canada, July 2001.

A Proof of Invertibility

The algorithm described in this paper is based on the inversion of matrix $\mathbf{G}$. We prove that $\mathbf{G}$ is positive definite, and therefore invertible, using induction. In order to simplify the proof, we show that changing the reference pose does not change the positive definiteness of $\mathbf{G}$. Without loss of generality, $\mathbf{G}$ is a positive definite matrix of the form (4), with the reference node X_0. Switching the reference node to X_i results in the matrix $\mathbf{G}'$. These two matrices are related by

$$\mathbf{G}' = \mathbf{I}_i \mathbf{G}$$

where $\mathbf{I}_i$ is an identity matrix of size $(dn \times dn)$ with a row of negative $(d \times d)$ identity matrices of the form:

$$\mathbf{I}_i = \begin{pmatrix} I_{d(i-1)} & 0 & 0 \\ -I_d & \dots & -I_d \\ 0 & 0 & I_{d(n-i+1)} \end{pmatrix}.$$

Multiplication with $\mathbf{I}_i$ corresponds to replacing the submatrices at (i, j) with the negative sum of all submatrices at row j. Since $\mathbf{I}_i$ is invertible, $\mathbf{G}'$ remains positive definite.

Induction base $k = n$: Assuming a graph with $n + 1$ nodes and n links. The matrix $\mathbf{G}$ is transformed into the block diagonal matrix $\mathbf{G}'$, composed of covariance matrices by

$$\mathbf{G}' = \mathbf{I_D} \mathbf{G} \mathbf{I}_\mathbf{D}^T,$$

with an upper-right triangular matrix $\mathbf{I_D}$ of d-dimensional identity matrices

$$\mathbf{I_D} = \begin{pmatrix} I_d & \dots & I_d \\ & \ddots & \vdots \\ 0 & & I_d \end{pmatrix}.$$

Since $\mathbf{G}'$ is given by

$$\begin{aligned} G'_{i,i} &= C^{-1}_{i-1,i} \\ G'_{i,j} &= 0 \qquad (i \neq j) \end{aligned}$$

and all covariances are positive definite, $\mathbf{G}'$ itself is positive definite. The same holds for $\mathbf{G}$, as $\mathbf{I_D}$ is invertible.

Inductive step $k \to k + 1$: Let $\mathbf{G}$ be a positive definite matrix that corresponds to a graph with $n+1$ nodes and k links. An additional link between the nodes X_i and X_j is inserted, with positive definite covariance $C_{i,j}$. Without restriction, X_i is the reference node of the given graph, since the reference pose is arbitrary. Thus, the resulting matrix $\mathbf{G}'$ is changed only at the $d \times d$ submatrix $\mathbf{G'_{j,j}}$:

$$G'_{j,j} = G_{j,j} + C^{-1}_{i,j}.$$

In case $C_{i,j}$ is positive definite, $\mathbf{G}'^*$ is positive definite, too, iff

$$\mathbf{X}^T \mathbf{G}' \mathbf{X} > 0 \qquad \mathbf{X} \in \mathbb{R}^{d \cdot n} \qquad \mathbf{X} \neq 0,$$

which is equivalent to

$$\sum_{k,l=1}^{n} X_k^T G'_{k,l} X_l > 0, \tag{A.1}$$

where X_k are the d-dimensional subvectors of $\mathbf{X}$. Expanding Eq. (A.1) to

$$\begin{aligned}\sum_{k,l=1}^{n} X_k^T G'_{k,l} X_l &= X_j^T G'_{j,j} X_j + \sum_{\substack{k,l=1 \\ k\neq j\neq l}}^{n} X_k^T G_{k,l} X_l \\ &= X_j^T C_{i,j}^{-1} X_j + \sum_{k,l=1}^{n} X_k^T G_{k,l} X_l \\ &= X_j^T C_{i,j}^{-1} X_j + \mathbf{X}^T \mathbf{G} \mathbf{X} > 0.\end{aligned}$$

$\mathbf{G}'$ is a positive definite matrix. □

B Fast Construction of the Linear Equation System

To solve the linear equation system $\mathbf{GX} = \mathbf{B}$,

$$\begin{aligned}C_D^{-1} &= (\mathbf{M}^T\mathbf{M})/s^2 \\ C_D^{-1}\bar{D} &= (\mathbf{M}^T\mathbf{Z})/s^2.\end{aligned}$$

are needed. To calculate these efficiently, summations are substituted for matrix multiplication by using the regularities in the matrix $\mathbf{M}$. $\mathbf{M}^T\mathbf{M}$ is represented as a sum over all corresponding point pairs:

$$\mathbf{M}^T\mathbf{M} = \sum_{k=1}^{m} \begin{pmatrix} 1 & 0 & 0 & 0 & -y_k & -z_k \\ 0 & 1 & 0 & z_k & x_k & 0 \\ 0 & 0 & 1 & -y_k & 0 & x_k \\ 0 & z_k & -y_k & y_k^2+z_k^2 & x_k z_k & -x_k y_k \\ -y_k & x_k & 0 & x_k z_k & y_k^2+x_k^2 & y_k z_k \\ -z_k & 0 & x_k & -x_k y_k & y_k z_k & x_k^2+z_k^2 \end{pmatrix}.$$

Similarly, $\mathbf{M}^T\mathbf{Z}$ is calculated as follows:

$$\mathbf{M}^T\mathbf{Z} = \sum_{k=0}^{m} \begin{pmatrix} \Delta x_k \\ \Delta y_k \\ \Delta z_k \\ -z_k \cdot \Delta y_k + y_k \cdot \Delta z_k \\ -y_k \cdot \Delta x_k + x_k \cdot \Delta y_k \\ z_k \cdot \Delta x_k - x_k \cdot \Delta z_k \end{pmatrix}$$

with

$$\begin{pmatrix} \Delta x_k \\ \Delta y_k \\ \Delta z_k \end{pmatrix} = \bar{Z}_k = \bar{V}_a \oplus u_k^a - \bar{V}_b \oplus u_k^b$$

and an approximation for each point:

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = u_k \approx (\bar{V}_a \oplus u_k^a + \bar{V}_b \oplus u_k^b)/2.$$

Finally, s^2 is a simple summation using the observation of the linearized measurement equation $\bar{D} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}$:

$$\begin{aligned} s^2 = \sum_{k=0}^{m} \Big[& (\Delta x_k - (\bar{D}_0 - y_k \cdot \bar{D}_4 + z_k \cdot \bar{D}_5))^2 \\ & + (\Delta y_k - (\bar{D}_1 - z_k \cdot \bar{D}_3 + x_k \cdot \bar{D}_4))^2 \\ & + (\Delta z_k - (\bar{D}_2 + y_k \cdot \bar{D}_3 - x_k \cdot \bar{D}_5))^2 \Big]. \end{aligned}$$

$\bar{D}_i$ denotes the i-th entry of the vector $\bar{D}$. Summation of C_D^{-1} and $C_D^{-1}\bar{D}$ yields $\mathbf{B}$ and $\mathbf{G}$.

2.2.2 Study of Parameterizations for the Rigid Body Transformations of the Scan Registration Problem

This next paper investigates many different routes for minimizing Eq. 2.3 and its equivalent for any number of point clouds. In particular, several different parametrizations of rotations in 3D were used which results in multiple linearizations of Eq. 2.3. The uncertainty based linearization of Eq. 2.3 using Euler angles as presented in Section 2.2.1 is consequently extended to the quaternion representation. Another formulation, also based on Euler angles is used for linearizing Eq. 2.3 as well. In this case, the explicit probabilistic formulations are dropped for simplification. Furthermore, the *sin* and *cos* terms are replaced with simple linear terms that are most accurate for small angles. This results in a very simple set of optimization equations. This linearization is referred to as small angle approximation. Finally, the helix-transform is used to linearize Eq. 2.3. The helix transform is a parametrization of rigid transformations in 3D. Each helix transform defines an axis. A point that is rigidly transformed using the transform is rotated around that axis in a helical fashion. In this way all rigid transformations can be represented by the helix transform. A linearization of the helix transform was given by Hofer and Pottman [34, 62]. In addition to presenting novel optimizations of Eq. 2.3, the paper also presents additional solutions to the problem of registration for more than two scans. The alternative linearizations are compared on the basis of their performance in scan registration. It turns out that it is not particularly important how Eq. 2.3 is minimized for the performance of the matching algorithm. Sophisticated linearizations based on probability theory perform on par with much less sophisticated ad-hoc linearizations. Because of the iterative nature of the studied matching algorithm, small differences in the minimization result will be made inconsequential during later iterations.

Study of Parameterizations for the Rigid Body Transformations of The Scan Registration Problem

Andreas Nüchter*, Jan Elseberg*, Peter Schneider†, Dietrich Paulus†

*Jacobs University Bremen
School of Engineering and Science
Campus Ring 1, 28759 Bremen, Germany
andreas@nuechti.de

†University of Koblenz-Landau
Institute for Computational Visualistics, Active Vision Group
Universitätsstr. 1, 56070 Koblenz, Germany

Abstract

The ICP (Iterative Closest Point) algorithm is the de facto standard for geometric alignment of three-dimensional models when an initial relative pose estimate is available. The basis of the algorithm is the minimization of an error function that takes point correspondences into account. Four closed-form solution methods are known for minimizing this function. This paper presents novel linear solutions to the scan registration problem, i.e., to the problem of putting and aligning 3D scans in a common coordinate system. We extend the methods for registering n-scans in a global and simultaneous fashion, such that the registration of the n-th scan influences all previous registrations in *one* step.

Key words: 3D scan matching, 3D point cloud registration, ICP algorithm

1 Introduction

Registering 3D models, i.e., putting two or more 3D scans in a common coordinate system, is a crucial step in 3D model construction. Many applications use the iterative closest points (ICP) algorithm. Nowadays precise 3D scanners are available that are used in architecture, industrial automation, agriculture, cultural heritage conservation, and facility management. Other applications of point cloud registration algorithms include medical data processing, art history, archaeology, and rescue and inspection robotics. The recent advent of

time-of-flight (TOF) 3D cameras is likely to generate another burst of ICP-like applications in the near future [52, 56, 57].

The ICP algorithm registers two independently acquired 3D scans or 3D point clouds into a common coordinate system. Here the algorithm relies on minimizing an error function over closest point correspondences. The following analogy is often used to depict the minimization issue: The correspondences represent a system of springs that forces the scan to be aligned to the correct position. Four closed form solution methods are known for minimizing the ICP error function [28]. The difficulty in minimizing the ICP error function is to ensure the orthonormality constraint of the included rotation matrix. This paper presents linearized solution methods, where the optimal rotation is approximated by solving a system of linear equations. The linearization methods are the helix transform, the small angle approximation, and the uncertainty based registrations. For the latter one we model the scan pose, i.e., position and orientation, as Gaussian distributions and use the Euler and the quaternion representation. Our objective is to compare different gradient-descent algorithms based upon different rotation parameterizations to solve the bundle adjustment problem that arises with 3D point clouds produced by range finders. In [46] the recommendation concerning the parameterization choice is stated as:

> "*Similarly, experience suggests that quasi-global 3 parameter rotation parameterizations such as Euler angles cause numerical problems unless one can be certain to avoid their singularities and regions of uneven coverage. Rotations should be parameterized using either quaternions subject to* $|\dot{q}|^2 = 1$*, or local perturbations* $\mathbf{R}\delta\mathbf{R}$ *or* $\delta\mathbf{R}\mathbf{R}$ *of an existing rotation* $\mathbf{R}$*, where* $\delta\mathbf{R}$ *can be any well-behaved 3 parameter small rotation approximation, e.g.* $\delta\mathbf{R} = (\mathbf{I}_{3\times3} + [\delta\mathbf{r}]_\times)$*, the Rodriguez formula, local Euler angles, etc.*".

Recently, Grisetti et al. [21] used a gradient descent technique for 3D robotic mapping incorporating incremental spherical linear interpolation [2]. Also Mitra et al. [31] and C. Hertzberg [22] make use of some gradient descent method for registration. This paper focuses on the issue of linearization of the rotation manifold and studies alternative methods.

If n-scans have to be registered, any sequential application of ICP will accumulate errors, and therefore the ICP algorithm has to be extended. Fig. 1 presents an urban scene digitalized with roughly 1000 3D scans. The accompanying video demonstrates the ICP scan matching and the loop closing, when the robot that acquires the 3D scans returns to a location where it has been before. Every frame corresponds to an ICP scan matching (we skip the animation of the actual ICP scan matching), or to one step of the global scan matching (red frames)). Locally consistent registration algorithms retain the analogy of the spring system [13] and the resulting algorithms need many

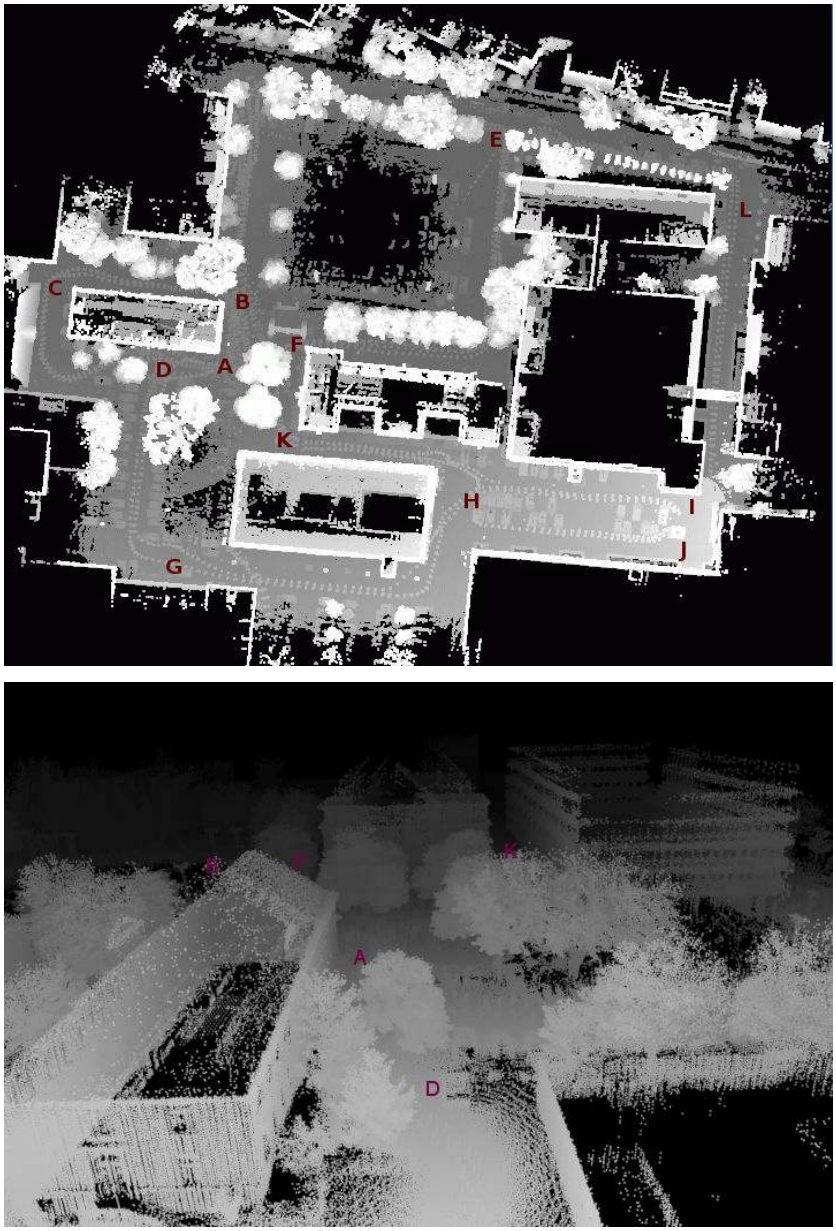


Fig. 1. Registration of n-scans. Top: scene HANNOVER2 of an urban environment in a bird's eye view. The scans have been taken according to the trajectory: A-B-C-D-A-B-E-F-A-D-G-H-I-J-H-K-F-E-L-I-K-A. A video of the scan matching process can be found under http://plum.eecs.jacobs-university.de/submissions/large_slam.mpg. Bottom: 3D view of the scene viewed from the left side.

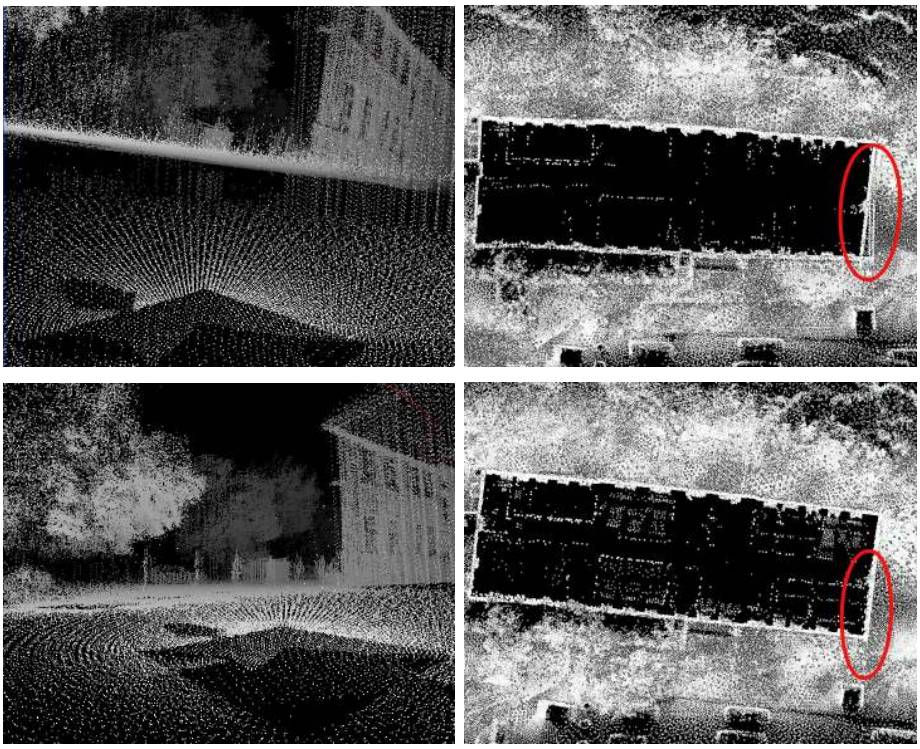


Fig. 2. Top left: 3D view of the scene with the accumulated error during the sequence A-B-C-D-A, where we recorded 135 3D scans. Top right: Scene in a bird's eye view. Bottom row: Consistent registration results. Left: Globally consistent registration in 3D view. Right: Bird's eye view.

iterations for minimizing the global error function. However, globally consistent algorithms minimize the error function in one step. Therefore, it is more likely to reach the global and hopefully the correct optimum. Fig. 2 shows the accumulated error when performing sequential ICP scan matching and the result of globally consistent scan matching. Fig. 3 stresses the difference between global and local optimal results and gives a schematic illustration of the difference.

This paper presents linear solutions to the global n-scan registration problem. The two-scan cases are extended to n-scans, thus a system of linear equations is built and finally solved to yield new scan poses. The presented methods are examined with respect to computational requirements and to approximation and implementation issues.

The paper is structured as follows: Next we summarize the state of the art for 3D registration of laser scans. Since this technique is often used in robotics for mapping, we will also briefly discuss current developments in this field.

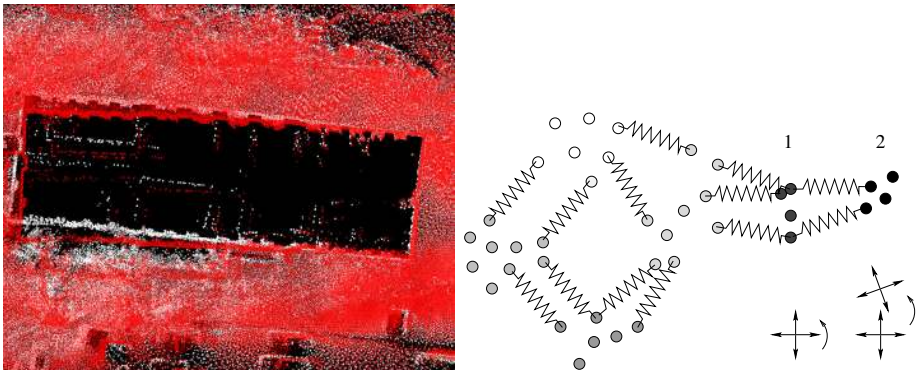


Fig. 3. Left: Locally consistent algorithms reduce the registration errors at the loop closing point but fail to distribute the error. Here in comparison with the global optimal result in bird's eye view. (cf. Fig. 2). Right: Schematic view of the registration of 6 Scans, whose points are represented by different gray values. Algorithms working locally optimal move every scan separately, while global optimal algorithms minimize all distances at the same time. This includes especially, that the transformation of scan 1 is continued to scan 2 (the rightmost one), i.e., incorporating a "leverage effect".

Then we present the two-scan registration methods. Section 4 presents the solutions to the n-scan registration problem followed by a detailed analysis. Finally, Section 6 presents the experiments and results. Inspired by these evaluations we develop an additional solution to the n-scan registration problem in Section 7, which turns out to subsume an already given solution. Finally, section 8 concludes.

2 State of the Art

2.1 The ICP Algorithm

The following method is the de-facto standard for registration of point sets. The complete algorithm was invented at the same time in 1991 by Besl and McKay [6], by Chen and Medioni [11] and by Zhang [51]. The method is called the *Iterative Closest Points (ICP) algorithm.*

Given two independently acquired sets of 3D points, $\hat{M}$ (model set) and $\hat{D}$ (data set) which correspond to a single shape, we want to find the transformation $(\mathbf{R}, \mathbf{t})$ consisting of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ which

minimizes the following cost function:

$$E(\mathbf{R}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t})||^2 , \quad (1)$$

All corresponding points can be represented in a tuple $(\mathbf{m}_i, \mathbf{d}_i)$ where $\mathbf{m}_i \in M \subset \hat{M}$ and $\mathbf{d}_i \in D \subset \hat{D}$. Two things have to be calculated: First, the corresponding points, and second, the transformation ($\mathbf{R}$, $\mathbf{t}$) that minimizes $E(\mathbf{R}, \mathbf{t})$ on the basis of the corresponding points. The ICP algorithm uses closest points as corresponding points. A sufficiently good starting guess enables the ICP algorithm to converge to the correct minimum.

Current research in the context of ICP algorithms mainly focuses on fast variants of ICP algorithms [39]. If the input are 3D meshes then a point-to-plane metric can be used instead of Eq. (1). Minimizing using a point-to-plane metric outperforms the standard point-to-point one, but requires the computation of normals and meshes in a pre-processing step.

The computation of closest points is the most expensive step in the ICP algorithm. Using the optimized k-d trees the cost for finding the closest point to a given query point is at average in the order of $O(\log N)$ [19], thus the overall cost is $O(N \log N)$ (expected time). Note: N can be very large; advanced high-precise 3D laser scanners such as the Zoller+Fröhlich yield a data rate up to 500000 3D points per second [54]. Improvements to k-d tree search have been presented. They include approximate k-d tree search [20], registration using $d2$-trees [31] and cached k-d tree search [32].

2.2 Marker and Feature-based Registration

To avoid issues with starting guess in the ICP framework, marker based registration uses defined artificial or natural landmarks as corresponding points. This manual data association ensures that by minimizing Eq. (1) the scans are registered at the correct location. Iterations are no longer required. Feature based algorithms, like using SIFT features, automatically extract the 3D position of natural features and do not need any iterations nor manual interference for registration [8].

While registering several 3D data sets using the ICP algorithm or marker and feature-based registration techniques, errors sum up. These errors are due to imprecise measurements and small registration errors. Therefore, globally consistent scan matching algorithm aim at reducing these errors.

2.3 Globally Consistent Scan Matching

Chen and Medioni [12] aimed at globally consistent range image alignment when introducing an incremental matching method, i.e., all new scans are registered against the so-called metascan, which is the union of the previously acquired and registered scans. This method does not spread out the error and is order-dependent.

Bergevin et al. [5], Stoddart and Hilton [41], Benjemaa and Schmitt [3, 4], and Pulli [38] present iterative approaches. Based on networks representing overlapping parts of images, they use the ICP algorithm for computing transformations that are applied after all correspondences between all views have been found. However, the focus of research is mainly 3D modeling of small objects using a stationary 3D scanner and a turn table; therefore, the used networks consist mainly of one loop [38], where the loop closing has to be smoothed. These solutions are locally consistent algorithms that stick to the mentioned analogy of the spring system [13] whereas true globally consistent algorithms minimize the error function in one step.

A probabilistic approach was proposed by Williams et al. [48], where each scan point is assigned a Gaussian distribution in order to model the statistical errors made by laser scanners. This causes high computation time due to the large amount of data in practice. Krishnan et al. [27] presented a global registration algorithm that minimizes the global error function by optimization on the manifold of 3D rotation matrices.

2.4 Current Trends

Recently, alternatives to ICP has been presented. Registration without ICP was described by Pottmann et al. [37] by relying on the geometry of the squared distance function of a scanned surface. In addition, the normal distribution transform (NDT) was proposed as an alternative to the ICP algorithm [7,30]. The NDT divides the 3D space into boxes and correlates all boxes of two scans, yielding more reliable scan matching with respect to the starting guess compared to the ICP algorithm at the expense of computational costs. An extension to the NDT algorithm to globally consistent scan matching is still missing.

2.5 Robotic 3D Mapping

Simultaneous Localization and Mapping (SLAM) is the problem of building a map of an unknown environment with a mobile robot while at the same time navigating in the environment, using the unfinished map. The map building problem is a fundamental problem in robotics, since the presence of an accurate map is needed in many applications. The development of solutions to the localization problem as well as the SLAM problem exploit probabilistic methods [42], resulting in a de facto standard, i.e., probabilistic robotics [44]. States of the robot and its environment are represented by probability distributions and the application of Bayes rule allows to integrate sensor measurements. Depending on the representation of the probability distributions we see Kalman filters (Gaussians) [16], grid based approaches (non-Gaussians) [43], and particle filters (sample based non-Gaussians) [45]. The latter two approaches keep for one state several hypothesis at the time. Thus, they are computationally more expensive, but add reliability to the robotic system.

Progress in sensing technology and the need to handle more degree of freedom, resulted in a step backwards and one-hypothesis systems are again set-up. However, probabilistic representations are still preferred, e.g. in [15,17,21,33].

3 2-Scan Registration

Four algorithms are currently known that solve the error function of the ICP algorithm in closed form [28]. These algorithms are briefly discussed next, followed by the description of our linear solutions.

3.1 Closed Form Solutions

Four algorithms are currently known that solve the error function of the ICP algorithm in closed form [28]. The difficulty of this minimization is to enforce the orthonormality constraint for the rotation matrix $\mathbf{R}$. Three of these algorithms separate the computation of the rotation $\mathbf{R}$ from the computation of the translation $\mathbf{t}$. These algorithms compute the rotation first and afterward the translation is derived using the rotation. For this separation, two point sets M' and D' have to be computed, by subtracting the mean of the points that are used in the matching:

$$\mathbf{c}_m = \frac{1}{N}\sum_{i=1}^{N}\mathbf{m}_i, \qquad \mathbf{c}_d = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d}_i \tag{2}$$

and

$$M' = \{\mathbf{m}'_i = \mathbf{m}_i - \mathbf{c}_m\}_{1,\ldots,N}, \qquad D' = \{\mathbf{d}'_i \ = \mathbf{d}_i \ - \mathbf{c}_d\}_{1,\ldots,N}. \tag{3}$$

After replacing Eq. (2) and (3) in the error function, $E(\mathbf{R}, \mathbf{t})$ Eq. (1) becomes:

$$\begin{aligned} E(\mathbf{R}, \mathbf{t}) &= \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i - \underbrace{(\mathbf{t} - \mathbf{c}_m + \mathbf{R}\mathbf{c}_d)}_{=\tilde{\mathbf{t}}}||^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i||^2 - \frac{2}{N}\tilde{\mathbf{t}} \cdot \sum_{i=1}^{N} (\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i) + \frac{1}{N} \sum_{i=1}^{N} \left|\left|\tilde{\mathbf{t}}\right|\right|^2 . \end{aligned} \tag{4}$$

In order to minimize the sum above, all terms have to be minimized. The second sum is zero, since all values refer to centroid. The third part has its minimum for $\tilde{\mathbf{t}} = \mathbf{0}$ or

$$\mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d.$$

Therefore the algorithm has to minimize only the first term, and the error function is expressed in terms of the rotation only:

$$E(\mathbf{R}, \mathbf{t}) \propto \sum_{i=1}^{N} ||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i||^2 . \tag{5}$$

(1) The first method was developed 1987 by Arun, Huang and Blostein [1]. The rotation $\mathbf{R}$ is represented as an orthonormal 3×3 matrix. The optimal rotation is calculated by $\mathbf{R} = \mathbf{V}\mathbf{U}^T$. Here the matrices $\mathbf{V}$ and $\mathbf{U}$ are derived by the singular value decomposition $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$ of a cross correlation matrix $\mathbf{H}$. This 3×3 matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \sum_{i=1}^{N} \mathbf{m}_i'^T \mathbf{d}'_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}, \tag{6}$$

where $S_{xx} = \sum_{i=1}^{N} \ m'_{x,i} d'_{x,i}$, $S_{xy} = \sum_{i=1}^{N} \ m'_{x,i} d'_{y,i}$, ...

(2) The second method is similar to the previous method and was independently developed in 1988 by Horn, Hilden and Negahdaripour [26]. Again, a correlation Matrix $\mathbf{H}$ according to Eq. (6) is calculated. Afterwards a so-called polar decomposition is computed, i.e., $\mathbf{H} = \mathbf{P}\mathbf{S}$, where $\mathbf{S} = (\mathbf{H}^T\mathbf{H})^{1/2}$. For this polar decomposition Horn et al. define a square root of a matrix [26]. Let $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{P}$ the matrices as described above.

Then the optimal rotation is given by

$$\mathbf{R} = \mathbf{P} = \mathbf{H}\left(\frac{1}{\sqrt{\lambda_1}}\mathbf{u}_1\mathbf{u}_1^T + \frac{1}{\sqrt{\lambda_2}}\mathbf{u}_2\mathbf{u}_2^T + \frac{1}{\sqrt{\lambda_3}}\mathbf{u}_3\mathbf{u}_3^T\right),$$

where $\{\lambda_i\}$ are the eigenvalues and $\{\mathbf{u}_i\}$ the corresponding eigenvectors of the matrix $\mathbf{H}^T\mathbf{H}$ [26].

(3) The third method finds the transformation for the ICP algorithm by using unit quaternions. This method was invented in 1987 by Horn [25]. The rotation represented as unit quaternion $\dot{\mathbf{q}}$, that minimizes (1), corresponds to the largest eigenvalue of the cross covariance matrix $\mathbf{N}$

$$\begin{pmatrix} (S_{xx}+S_{yy}+S_{zz}) & (S_{yz}+S_{zy}) & (S_{zx}+S_{xz}) & (S_{xy}+S_{yx}) \\ (S_{yz}+S_{zy}) & (S_{xx}-S_{yy}-S_{zz}) & (S_{xy}+S_{yx}) & (S_{zx}+S_{xz}) \\ (S_{zx}+S_{xz}) & (S_{xy}+S_{yx}) & (-S_{xx}+S_{yy}-S_{zz}) & (S_{yz}+S_{zy}) \\ (S_{xy}+S_{yx}) & (S_{yz}+S_{zy}) & (S_{zx}+S_{xz}) & (-S_{xx}-S_{yy}+S_{zz}) \end{pmatrix}.$$

(4) The fourth solution method for minimizing Eq. (1) uses so-called dual quaternions. This method was developed by Walker, Shao and Volz in 1991 [47]. Unlike the first three methods covered so far the transformation is found in a single step. There is no need to apply the trick with centroids to compute the rotation in a separate fashion. Here, the optimal transformation consisting of a rotation and translation is again a solution of the eigenvalue problem of a 4×4 matrix function that is built from corresponding point pairs.

3.2 Linearized Solutions

The closed-form solutions discussed so far are all non-linear, since they need an eigenvector/eigenvalue solver, e.g., in case of using the third method, a quartic equation must be solved [25]. Next, we present four linear solutions for minimizing the ICP error function Eq. (1). The advantage of these linear solutions is that they can be extended straightforward to n-scan registrations.

3.2.1 Registration using the Helix Transform

Under the assumption that the transformation $(\mathbf{R}, \mathbf{t})$ that has to be calculated by the ICP algorithm is small, we can approximate the solution by applying instantaneous kinematics. This solution was initially given by Hofer and Pottmann [24, 36] and is repeated here, to make the paper self-contained and to enable us to extend the solution in [24] to the global optimal case.

Instantaneous kinematics computes the displacement of a 3D point by an affine

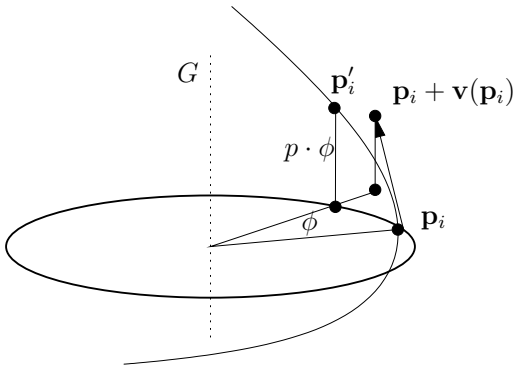


Fig. 4. The affine position of a 3D point $\mathbf{p}_i + \mathbf{v}(\mathbf{p}_i)$ is different from the rigid transformation that results in point $\mathbf{p}'_i$. Based on [24].

transformation via a so-called helical motion [36]. A 3D point $\mathbf{p}$ is mapped by adding a small displacement $\mathbf{v}(\mathbf{p})$ given by the parameters $\bar{\mathbf{x}}$ and $\mathbf{x}$, i.e.,

$$\mathbf{v}(\mathbf{p}) = \bar{\mathbf{x}} + \mathbf{x} \times \mathbf{p}.$$

To minimize the ICP error function Eq. (1) the displacement of the points in D has to minimize the distances between the point pairs. Thus Eq. (1) is rewritten as follows:

$$\begin{aligned} E(\bar{\mathbf{x}}, \mathbf{x}) &= \sum_{i=1}^{N} ||\mathbf{m}_i - (\mathbf{d}_i + \mathbf{v}(\mathbf{d}_i))||^2 \\ &= \sum_{i=1}^{N} ||\mathbf{m}_i - (\mathbf{d}_i + \bar{\mathbf{x}} + \mathbf{x} \times \mathbf{d}_i)||^2 \end{aligned} \tag{7}$$

Since Eq. (7) is a quadratic function with the two unknown variables $\bar{\mathbf{x}}, \mathbf{x}$, the optimal displacement to Eq. (7) is given by the solution of the following linear system: $\bar{\mathbf{x}} + \mathbf{x} \times \mathbf{d}_i = \mathbf{D}_i \cdot \mathbf{u}$ where $\mathbf{u} = (\mathbf{x}^T, \bar{\mathbf{x}}^T)^T$ and

$$\mathbf{D}_i = \begin{pmatrix} 0 & d_{z,i} & -d_{y,i} & 1 & 0 & 0 \\ -d_{z,i} & 0 & d_{x,i} & 0 & 1 & 0 \\ d_{y,i} & -d_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\sum_{i=1}^{N} \mathbf{D}_i^T \mathbf{D}_i \mathbf{u} = \sum_{i=1}^{N} \mathbf{D}_i^T (\mathbf{m}_i - \mathbf{d}_i) \tag{8}$$

The optimal displacement calculated by Eq. (8) corresponds to an affine motion. Figure 4 presents the displacement of a point using the affine transfor-

mation and rigid transformation. Therefore in a post processing step a rigid transformation $(\mathbf{R}, \mathbf{t})$ is calculated from $(\bar{\mathbf{x}}, \mathbf{x})$ as follows: If $\mathbf{x} = \mathbf{0}$ only a translation is present. In this case $\mathbf{t} = \bar{\mathbf{x}}$ holds. Otherwise an axis G consisting of a direction vector $\mathbf{g}$ and a momentum vector $\bar{\mathbf{g}}$ can be computed. The tuple $(\mathbf{g}, \bar{\mathbf{g}})$ are the Plücker coordinates of the axis G:

$$\mathbf{g} = \frac{\mathbf{x}}{||\mathbf{x}||}, \quad \bar{\mathbf{g}} = \frac{\mathbf{x} - p\mathbf{x}}{||\mathbf{x}||}, \quad p = \frac{\mathbf{x}^T \cdot \bar{\mathbf{x}}}{\mathbf{x}^2}$$

Based on G the point $\mathbf{d}_i$ has to be transformed as follows:

$$\mathbf{d}'_i = \mathbf{R}(\mathbf{d}_i - \mathbf{r}) + (p\ \phi)\mathbf{g} + \mathbf{r} \tag{9}$$

Here $\mathbf{R}$ is the rotation matrix

$$\mathbf{R} = \frac{1}{b_0^2+b_1^2+b_2^2+b_3^2} \begin{pmatrix} b_0^2 + b_1^2 - b_2^2 - b_3^2 & 2(b_1b_2 + b_0b_3) & 2(b_1b_3 - b_0b_2) \\ 2(b_1b_2 - b_0b_3) & b_0^2 - b_1^2 + b_2^2 - b_3^2 & 2(b_2b_3 + b_0b_1) \\ 2(b_1b_3 + b_0b_2) & 2(b_2b_3 - b_0b_1) & b_0^2 - b_1^2 - b_2^2 + b_3^2 \end{pmatrix}, \tag{10}$$

where $b_0 = \cos(\phi/2)$, $b_1 = g_x \sin(\phi/2)$, $b_2 = g_y \sin(\phi/2)$, and $b_3 = g_z \sin(\phi/2)$. $\mathbf{g} = (g_x, g_y, g_z)$ is the above mentioned direction vector of the axis G. Furthermore in Eq. (9) $\mathbf{r}$ is an arbitrary point on the axis G. Note: Eq. (10) is similar to the term for computing a rotation matrix from a unit quaternion. Please refer to [24] for more details.

3.2.2 Small Angle Approximation

Next, we develop a novel method based on the small angle approximation. Its basis is again the assumption that the transformation $(\mathbf{R}, \mathbf{t})$ to be calculated by the ICP algorithm is small. Given a rotation matrix $\mathbf{R}$ based on the Euler angles

$$\begin{pmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z & -\cos\theta_y \sin\theta_x \\ \sin\theta_x \sin\theta_z - \cos\theta_x \cos\theta_z \sin\theta_y & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z & \cos\theta_x \cos\theta_y \end{pmatrix} \tag{11}$$

we use the first-order Taylor series approximation that is valid for small angles

$$\sin\theta \approx \theta - \frac{\theta^3}{3} + \frac{\theta^5}{5} - \cdots$$
$$\cos\theta \approx 1 - \frac{\theta^2}{2} + \frac{\theta^4}{4} - \cdots$$

and apply it to (11). The resulting approximative rotation is

$$\mathbf{R} \approx \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_x\theta_y + \theta_z & 1 - \theta_x\theta_y\theta_z & -\theta_x \\ \theta_x\theta_z - \theta_y & \theta_x + \theta_y\theta_z & 1 \end{pmatrix}. \tag{12}$$

As a second approximation we assume that the result of a multiplication of small angles yields even smaller values that can be omitted as well. This eliminates second order and higher combination terms and Eq. (12) becomes:

$$\mathbf{R} \approx \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{pmatrix} = \mathbf{I}_{3\times 3} + \begin{pmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{pmatrix}. \tag{13}$$

We notice that this term is closely related to the Rodrigues formula but is no longer orthonormal.

Replacing this approximation (13) in the ICP error function Eq. (1) and rearranging the unknown variables in a vector yields:

$$\begin{aligned} \mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t}) &\approx \mathbf{m}_i - \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{pmatrix} \mathbf{d}_i - \mathbf{t} \\ &= \mathbf{m}_i - \mathbf{d}_i - \begin{pmatrix} 0 & d_{z,i} & -d_{y,i} & 1 & 0 & 0 \\ -d_{z,i} & 0 & d_{x,i} & 0 & 1 & 0 \\ d_{y,i} & -d_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_z \\ \theta_y \\ \theta_x \\ t_x \\ t_y \\ t_z \end{pmatrix}. \end{aligned} \tag{14}$$

Surprisingly, the resulting equation is equal to the notation of Eq. (7) by Hofer and Pottmann [24, 36]. The top part of the solution vector was called $\mathbf{c}$ while the bottom part resembles $\bar{\mathbf{c}}$. Therefore, the result has to be interpreted as a so-called helical motion and not using the small angle assumption. The small angle approximation fails, since the rotation calculated by the ICP algorithm always refers to the global coordinate system, i.e., represents a rotation about the origin $(0, 0, 0)$. While the helical motion takes care of this by calculating a rotation axis, our approximation does not regard this.

In order to make the small angle approximation of the rotation matrix Eq. (13) work, we have to apply the clever centroid trick, already used to derive Eq. (5).

This separates the rotation from the translation. The resulting term is

$$\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i \approx \mathbf{m}'_i - \mathbf{d}'_i - \begin{pmatrix} 0 & d'_{z,i} & -d'_{y,i} \\ -d'_{z,i} & 0 & d'_{x,i} \\ d'_{y,i} & -d'_{x,i} & 0 \end{pmatrix} \begin{pmatrix} \theta_z \\ \theta_y \\ \theta_x \end{pmatrix}$$

Suppose the vector $\mathbf{u}$ to be $\mathbf{u} = (\theta_z, \theta_y, \theta_x)$. Then the optimal displacement for the ICP error function composed using the approximation in Eq. (14) is given by the solution to the following linear equation system:

$$\mathbf{D}_i = \begin{pmatrix} 0 & d'_{z,i} & -d'_{y,i} \\ -d'_{z,i} & 0 & d'_{x,i} \\ d'_{y,i} & -d'_{x,i} & 0 \end{pmatrix}$$

$$\sum_{i=1}^{N} \mathbf{D}_i^T \mathbf{D}_i \mathbf{u} = \sum_{i=1}^{N} \mathbf{D}_i^T (\mathbf{m}'_i - \mathbf{d}'_i)$$

After the approximation of the rotation is found, the optimal translation is calculated in Eq. (4)

3.2.3 Uncertainty-based Registration

For some applications it is necessary to have a notion of the uncertainty of the poses calculated by the registration algorithm. The following is the extension of the probabilistic approach first proposed in [29] to 6 DoF. This extension is not straightforward, since the matrix decomposition, i.e., Eq. (16) cannot be derived from first principles. For a more detailed description of the extension refer to [9] and [10]. In addition to the pose $\mathbf{X}$, the pose estimate $\bar{\mathbf{X}}$ and the pose error $\Delta\mathbf{X}$ are required.

The positional error of a scan at its pose $\mathbf{X}$ is described by:

$$E = \sum_{i=1}^{m} \|\mathbf{X} \oplus \mathbf{d}_i - \mathbf{m}_i\|^2 = \sum_{i=1}^{m} \|\mathbf{Z}_i(\mathbf{X})\|^2$$

Here, $\oplus$ is the compounding operation that transforms a point $\mathbf{d}_i$ into the global coordinate system. For small pose errors $\Delta\mathbf{X}$, E can be linearized by use of a Taylor expansion:

$$\begin{aligned} \mathbf{Z}_i(\mathbf{X}) &\approx \bar{\mathbf{X}} \oplus \mathbf{d}_i - \mathbf{m}_i - \nabla\mathbf{Z}_i(\bar{\mathbf{X}})\Delta\mathbf{X} \\ &= \mathbf{Z}_i(\bar{\mathbf{X}}) - \nabla\mathbf{Z}_i(\bar{\mathbf{X}})\Delta\mathbf{X} \end{aligned}$$

Utilizing the matrix decomposition $\mathbf{M}_i\mathbf{H}$ of $\nabla\mathbf{Z}_i(\bar{\mathbf{X}})$ that separates the pose

$\mathbf{X}$, which is contained in $\mathbf{H}$ from the points $\mathbf{m}_i$ and $\mathbf{d}_i$, which are contained in $\mathbf{M}_i$:

$$\mathbf{Z}_i(\mathbf{X}) \approx \mathbf{Z}_i(\bar{\mathbf{X}}) - \mathbf{M}_i\mathbf{H}\Delta\mathbf{X}$$

Appropriate decompositions are given for the Euler angles, quaternion representation and the Helix transform in the following paragraphs. Because $\mathbf{M}_i$ is independent of the pose, the positional error E is approximated as:

$$E \approx (\mathbf{Z} - \mathbf{M}\mathbf{H}\Delta\mathbf{X})^T(\mathbf{Z} - \mathbf{M}\mathbf{H}\Delta\mathbf{X}),$$

where $\mathbf{Z}$ is the concatenation of all $\mathbf{Z}_i(\bar{\mathbf{X}})$ and $\mathbf{M}$ the concatenation of all $\mathbf{M}_i$'s.

E is minimized by the ideal pose:

$$\bar{\mathbf{E}} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}$$

and its covariance is given by

$$\mathbf{C} = s^2(\mathbf{M}^T\mathbf{M}),$$

where s^2 is the unbiased estimate of the covariance of the identically, independently distributed errors of $\mathbf{Z}_i$:

$$s^2 = (\mathbf{Z} - \mathbf{M}\bar{\mathbf{E}})^T(\mathbf{Z} - \mathbf{M}\bar{\mathbf{E}})/(2m - 3). \tag{15}$$

Note that $\bar{\mathbf{E}}$ is the minimum for the linearized pose $\mathbf{H}\Delta\mathbf{X}$. To obtain the optimal $\mathbf{X}$ the following transformation is performed:

$$\mathbf{X} = \bar{\mathbf{X}} - \mathbf{H}^{-1}\bar{\mathbf{E}},$$
$$\mathbf{C} = (\mathbf{H}^{-1})\mathbf{C}(\mathbf{H}^{-1})^T.$$

3.2.3.1 Euler Angles The representation of pose $\mathbf{X}$ in Euler angles, as well as its estimate and error is as follows:

$$\mathbf{X} = \begin{pmatrix} t_x \\ t_y \\ t_z \\ \theta_x \\ \theta_y \\ \theta_z \end{pmatrix}, \bar{\mathbf{X}} = \begin{pmatrix} \bar{t}_x \\ \bar{t}_y \\ \bar{t}_z \\ \bar{\theta}_x \\ \bar{\theta}_y \\ \bar{\theta}_z \end{pmatrix}, \Delta\mathbf{X} = \begin{pmatrix} \Delta t_x \\ \Delta t_y \\ \Delta t_z \\ \Delta\theta_x \\ \Delta\theta_y \\ \Delta\theta_z \end{pmatrix}$$

The matrix decomposition $\mathbf{M}_i\mathbf{H} = \nabla\mathbf{Z}_i(\bar{\mathbf{X}})$, i.e., the Jacobian, is given by:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & \bar{t}_z\cos(\bar{\theta}_x) + \bar{t}_y\sin(\bar{\theta}_x) & \bar{t}_y\cos(\bar{\theta}_x)\cos(\bar{\theta}_y) - \bar{t}_z\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) \\ 0 & 1 & 0 & -\bar{t}_z & -\bar{t}_x\sin(\bar{\theta}_x) & -\bar{t}_x\cos(\bar{\theta}_x)\cos(\bar{\theta}_y) - \bar{t}_z\sin(\bar{\theta}_y) \\ 0 & 0 & 1 & \bar{t}_y & -\bar{t}_x\cos(\bar{\theta}_x) & \bar{t}_x\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) + \bar{t}_y\sin(\bar{\theta}_y) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_y) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_x) & \cos(\bar{\theta}_x)\cos(\bar{\theta}_y) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_x) & -\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) \end{pmatrix}. \quad (16)$$

and

$$\mathbf{M}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & -d_{y,i} & -d_{z,i} \\ 0 & 1 & 0 & d_{z,i} & d_{x,i} & 0 \\ 0 & 0 & 1 & -d_{y,i} & 0 & d_{x,i} \end{pmatrix}.$$

As required, $\mathbf{M}_i$ contains all point information while $\mathbf{H}$ expresses the pose information. Thus, this matrix decomposition constitutes a pose linearization similar to those proposed in the preceding sections. Note that, while the matrix decomposition is arbitrary with respect to the column and row ordering of $\mathbf{H}$, this particular description was chosen due to its similarity to the 3D pose solution given in [29].

3.2.3.2 Quaternions The representation of the pose $\mathbf{X}$ as quaternions, as well as its estimate and error are given as follows:

$$\mathbf{X} = \begin{pmatrix} t_x \\ t_y \\ t_z \\ p \\ q \\ r \\ s \end{pmatrix}, \bar{\mathbf{X}} = \begin{pmatrix} \bar{t}_x \\ \bar{t}_y \\ \bar{t}_z \\ \bar{p} \\ \bar{q} \\ \bar{r} \\ \bar{s} \end{pmatrix}, \Delta\mathbf{X} = \begin{pmatrix} \Delta t_x \\ \Delta t_y \\ \Delta t_z \\ \Delta p \\ \Delta q \\ \Delta r \\ \Delta s \end{pmatrix}$$

The matrix decomposition $\mathbf{M}_i\mathbf{H} = \nabla\mathbf{Z}_i(\bar{\mathbf{X}})$ for quaternions is given by:

$$\mathbf{M}_i = \begin{pmatrix} 1 & 0 & 0 & d_{x,i} & 0 & -d_{z,i} & d_{y,i} \\ 0 & 1 & 0 & d_{y,i} & d_{z,i} & 0 & -d_{x,i} \\ 0 & 0 & 1 & d_{z,i} & -d_{y,i} & d_{x,i} & 0 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_{3\times3} & -2\cdot\mathbf{T} \\ 0 & 2\cdot\mathbf{U} \end{pmatrix}$$

$$\mathbf{T}^T = \begin{pmatrix} \bar{p}\bar{t}_x + \bar{s}\bar{t}_y - \bar{r}\bar{t}_z & -\bar{s}\bar{t}_x + \bar{p}\bar{t}_y + \bar{q}\bar{t}_z & \bar{r}\bar{t}_x - \bar{q}\bar{t}_y + \bar{p}\bar{t}_z \\ \bar{q}\bar{t}_x + \bar{r}\bar{t}_y + \bar{s}\bar{t}_z & -\bar{r}\bar{t}_x + \bar{q}\bar{t}_y - \bar{p}\bar{t}_z & -\bar{s}\bar{t}_x + \bar{p}\bar{t}_y + \bar{q}\bar{t}_z \\ \bar{r}\bar{t}_x - \bar{q}\bar{t}_y + \bar{p}\bar{t}_z & \bar{q}\bar{t}_x + \bar{r}\bar{t}_y + \bar{s}\bar{t}_z & -\bar{p}\bar{t}_x - \bar{s}\bar{t}_y + \bar{r}\bar{t}_z \\ \bar{s}\bar{t}_x - \bar{p}\bar{t}_y - \bar{q}\bar{t}_z & \bar{p}\bar{t}_x + \bar{s}\bar{t}_y - \bar{r}\bar{t}_z & \bar{q}\bar{t}_x + \bar{r}\bar{t}_y - \bar{s}\bar{t}_z \end{pmatrix}$$

$$\mathbf{U} = \begin{pmatrix} \bar{p} & \bar{q} & \bar{r} & \bar{s} \\ \bar{q} & -\bar{p} & \bar{s} & -\bar{r} \\ \bar{r} & -\bar{s} & -\bar{p} & \bar{q} \\ \bar{s} & \bar{r} & -\bar{q} & -\bar{p} \end{pmatrix}$$

Again, all point information is restricted to $\mathbf{M}_i$ and all pose information to $\mathbf{H}$, so that the matrices describe a linearization of the quaternion transformation. Since this specific linearization was derived without regarding the normalized nature of the unit quaternions, this approach may yield non-unit quaternions. As non-unit quaternions do not constitute valid poses, the resulting quaternion must additionally be normalized before use.

4 n-Scan Registration

The n-scan registration using linearization allows us to compute global optimal poses in one step given point correspondences between adjacent scans. These scans are given by a graph, where each link, $j \to k$ denotes a set of point pairs, i.e., closest points. Following the notation of ICP, scan j serves as the model set, while scan k serves as data set. Next we present four novel linear methods for the parameterization of the rotation.

4.1 Global Registration using the Helix Transform

We minimize the error function for global consistent scan matching, i.e., we improve the approach given in [36] which is only locally consistent to include

off-diagonal elements in the resulting system of equations [23]. The error function is extended to include all poses $\mathbf{X}_j = (\mathbf{x}_j^T, \bar{\mathbf{x}}_j^T)^T$:

$$
\begin{aligned}
E &= \sum_{j\to k}\sum_{i}(\mathbf{m}_i - \mathbf{d}_i + (\bar{\mathbf{x}}_j + \mathbf{x}_j \times \mathbf{m}_i) - (\bar{\mathbf{x}}_k + \mathbf{x}_k \times \mathbf{m}_i))^2 \qquad (17)\\
&= \sum_{j\to k}\Bigg(\sum_{i}((\bar{\mathbf{x}}_j + \mathbf{x}_j \times \mathbf{m}_i) - (\bar{\mathbf{x}}_k + \mathbf{x}_k \times \mathbf{m}_i))^2\\
&\quad + \sum_{i} 2((\mathbf{m}_i - \mathbf{d}_i)\cdot(\bar{\mathbf{x}}_j + \mathbf{x}_j \times \mathbf{m}_i) + (\mathbf{d}_i - \mathbf{m}_i)\cdot(\bar{\mathbf{x}}_k + \mathbf{x}_k \times \mathbf{m}_i))\\
&\quad + \sum_{i}(\mathbf{m}_i - \mathbf{d}_i)^2\Bigg)
\end{aligned}
$$

Reformulating E as

$$E = \mathbf{X}^T \cdot \mathbf{B} \cdot \mathbf{X} + 2\mathbf{A} \cdot \mathbf{X} + \sum_i (\mathbf{m}_i - \mathbf{d}_i)^2,$$

allows us to solve for the optimal poses $\mathbf{X}$ in the linear equation system:

$$\mathbf{BX} = \mathbf{A}$$

$\mathbf{A}$ is attained as follows:

$$
\begin{aligned}
2\mathbf{AX} &= \sum_{j\to k}\sum_{i} 2((\mathbf{m}_i - \mathbf{d}_i)(\bar{\mathbf{x}}_j + \mathbf{x}_j \times \mathbf{m}_i) + (\mathbf{d}_i - \mathbf{m}_i)(\bar{\mathbf{x}}_k + \mathbf{x}_k \times \mathbf{m}_i))\\
\mathbf{A}_j &= \sum_{j\to k}\sum_{i}\begin{pmatrix} \mathbf{m}_i \times (\mathbf{m}_i - \mathbf{d}_i) \\ \mathbf{m}_i - \mathbf{d}_i \end{pmatrix} + \sum_{k\to j}\sum_{i}\begin{pmatrix} \mathbf{m}_i \times (\mathbf{d}_i - \mathbf{m}_i) \\ \mathbf{d}_i - \mathbf{m}_i \end{pmatrix}
\end{aligned}
$$

In other words, for each link $j \to k$ $\begin{pmatrix} \mathbf{m}_i \times (\mathbf{m}_i - \mathbf{d}_i) \\ \mathbf{m}_i - \mathbf{d}_i \end{pmatrix}$ is added to $\mathbf{A}_j$ while the same amount is subtract from $\mathbf{A}_k$.

$\mathbf{B}$ is defined by

$$\mathbf{X}^T \cdot \mathbf{B} \cdot \mathbf{X} = \sum_{j\to k}\sum_{i}((\bar{\mathbf{x}}_j + \mathbf{x}_j \times \mathbf{m}_i) - (\bar{\mathbf{x}}_k + \mathbf{x}_k \times \mathbf{m}_i))^2,$$

and given as

$$
\begin{aligned}
\mathbf{B}_{j,j} &= \sum_{\substack{j\to k\\ k\to j}}\sum_{i}\mathbf{M}_i\\
\mathbf{B}_{j,k} &= \sum_{j\to k}\sum_{i} -\mathbf{M}_i
\end{aligned}
$$

where

$$\mathbf{M}_i = \begin{pmatrix} m_{y,i}^2 + m_{z,i}^2 & -m_{x,i}m_{y,i} & -m_{x,i}m_{z,i} & 0 & -m_{z,i} & m_{y,i} \\ -m_{x,i}m_{y,i} & m_{x,i}^2 + m_{z,i}^2 & -m_{y,i}m_{z,i} & m_{z,i} & 0 & -m_{x,i} \\ -m_{x,i}m_{z,i} & -m_{y,i}m_{z,i} & m_{x,i}^2 + m_{y,i}^2 & -m_{y,i} & m_{x,i} & 0 \\ 0 & m_{z,i} & -m_{y,i} & 1 & 0 & 0 \\ -m_{z,i} & 0 & m_{x,i} & 0 & 1 & 0 \\ m_{y,i} & -m_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix}.$$

4.2 Global Registration using the Small Angle Approximation

Similarly to Eq. (17), the error metric is extended to include all poses $\mathbf{X}_j$:

$$E = \sum_{j \to k} \sum_i \left| \mathbf{R}_j \mathbf{m}_i + \mathbf{t}_j - (\mathbf{R}_k \mathbf{d}_i + \mathbf{t}_k) \right|^2 .$$

Using the centroids $\mathbf{c}_d$ and $\mathbf{c}_m$, where $\mathbf{m}'_i = \mathbf{m}_i - \mathbf{c}_m$, $\mathbf{d}'_i = \mathbf{d}_i - \mathbf{c}_d$, E is restated as:

$$\begin{aligned} E &= \sum_{j \to k} \sum_i \left| \mathbf{R}_j \mathbf{m}'_i + \mathbf{R}_j \mathbf{c}_m + \mathbf{t}_j - (\mathbf{R}_k \mathbf{d}'_i + \mathbf{R}_k \mathbf{c}_d + \mathbf{t}_k) \right|^2 \\ &= \sum_{j \to k} \sum_i \left| \mathbf{R}_j \mathbf{m}'_i - \mathbf{R}_k \mathbf{d}'_i - (\mathbf{t}_k - \mathbf{t}_j + \mathbf{R}_k \mathbf{c}_d - \mathbf{R}_j \mathbf{c}_m) \right|^2 \\ &= \sum_{j \to k} \Bigg(\sum_i \left| \mathbf{R}_j \mathbf{m}'_i - \mathbf{R}_k \mathbf{d}'_i \right|^2 \\ &\qquad - 2 \sum_i (\mathbf{t}_k - \mathbf{t}_j + \mathbf{R}_k \mathbf{c}_d - \mathbf{R}_j \mathbf{c}_m)(\mathbf{R}_j \mathbf{m}'_i - \mathbf{R}_k \mathbf{d}'_i) \\ &\qquad + \sum_i \left| \mathbf{t}_k - \mathbf{t}_j + \mathbf{R}_k \mathbf{c}_d - \mathbf{R}_j \mathbf{c}_m \right|^2 \Bigg) . \end{aligned}$$

The term $-2 \sum_i (\mathbf{t}_k - \mathbf{t}_j + \mathbf{R}_k \mathbf{c}_d - \mathbf{R}_j \mathbf{c}_m)(\mathbf{R}_j \mathbf{m}'_i - \mathbf{R}_k \mathbf{d}'_i)$ equates to zero, because all values refer to the centroids. This enables us to solve for the rotation of all poses independent of their translation.

4.2.1 Computing the Rotation

Minimizing the first term of the restated error metric allows us to derive the optimal rotation of all poses. Since the rotation is a nonlinear operation we utilize the following linearization: We apply the same small angle approximation

as in Eq. (14):

$$\begin{aligned}\mathbf{R}_j\mathbf{m}_i &= \begin{pmatrix} 0 & m_{z,i} & -m_{y,i} \\ -m_{z,i} & 0 & m_{x,i} \\ m_{y,i} & -m_{x,i} & 0 \end{pmatrix} \cdot \mathbf{X}_j + \mathbf{m}_i \\ &= \mathbf{M}_i \cdot \mathbf{X}_j + \mathbf{m}_i.\end{aligned}$$

The rotation is represented as $\mathbf{X}_j = (\theta_{z,j}, \theta_{y,j}, \theta_{x,j})^T$. The following rotational error will be minimized:

$$\begin{aligned}E_R &= \sum_{j\to k}\sum_i \left(\mathbf{M}_i \cdot \mathbf{X}_j - \mathbf{D}_i \cdot \mathbf{X}_k - (\mathbf{m}_i - \mathbf{d}_i)\right)^2 \\ &= \sum_{j\to k}\sum_i \left(\mathbf{M}_i \cdot \mathbf{X}_j - \mathbf{D}_i \cdot \mathbf{X}_k\right)^2 + (\mathbf{m}_i - \mathbf{d}_i)^2 \\ &\qquad - 2\left(\mathbf{M}_i \cdot \mathbf{X}_j - \mathbf{D}_i \cdot \mathbf{X}_i\right) \cdot (\mathbf{m}_i - \mathbf{d}_i).\end{aligned}$$

The error term is rewritten

$$E_R = \mathbf{XBX} + 2\mathbf{AX} + (\mathbf{m}_k - \mathbf{d}_k)^2$$

in order to solve the linear equation system

$$\mathbf{BX} + \mathbf{A} = 0.$$

With $\mathbf{B}$ given by:

$$\mathbf{B}_{j,j} = \sum_{j\to k}\sum_i \mathbf{D}_i^T \cdot \mathbf{D}_i + \sum_{k\to j}\sum_i \mathbf{M}_i^T \cdot \mathbf{M}_i$$

case $j < k$:

$$\mathbf{B}_{j,k} = -\sum_{j\to k}\sum_i \mathbf{M}_i^T \cdot \mathbf{D}_i$$

case $j > k$:

$$\mathbf{B}_{j,k} = -\sum_{j\to k}\sum_i \mathbf{D}_i^T \cdot \mathbf{M}_i,$$

and $\mathbf{A}$ by:

$$\mathbf{A}_j = \sum_{k\to j}\sum_i \begin{pmatrix} (m_{z,i}-d_{z,i})\cdot d_{y,i} - (m_{y,i}-d_{y,i})\cdot d_{z,i} \\ (m_{x,i}-d_{x,i})\cdot d_{z,i} - (m_{z,i}-d_{z,i})\cdot d_{x,i} \\ (m_{y,i}-d_{y,i})\cdot d_{x,i} - (m_{x,i}-d_{x,i})\cdot d_{y,i} \end{pmatrix} - \sum_{j\to k}\sum_i \begin{pmatrix} (m_{z,i}-d_{z,i})\cdot m_{y,i} - (m_{y,i}-d_{y,i})\cdot m_{z,i} \\ (m_{x,i}-d_{x,i})\cdot m_{z,i} - (m_{z,i}-d_{z,i})\cdot m_{x,i} \\ (m_{y,i}-d_{y,i})\cdot m_{x,i} - (m_{x,i}-d_{x,i})\cdot m_{y,i} \end{pmatrix}$$

4.2.2 Computing the Translation

With the optimal rotations successfully calculated, the optimal translation is determined by minimizing the term:

$$E_T = \sum_{j\to k}\sum_i \left(\mathbf{t}_k - \mathbf{t}_j + \mathbf{R}_k\mathbf{c}_d - \mathbf{R}_j\mathbf{c}_m\right)^2.$$

$\mathbf{R}_k\mathbf{c}_d - \mathbf{R}_j\mathbf{c}_m$ is abbreviated with $\mathbf{R}_{j,k}$:

$$E_T = \sum_{j\to k}\sum_i (\mathbf{t}_k - \mathbf{t}_j)^2 - 2(\mathbf{t}_k - \mathbf{t}_j)\mathbf{R}_{j,k} + \mathbf{R}_{j,k}^2.$$

Using the matrix notation:

$$E_T = \mathbf{T}^T\mathbf{B}\mathbf{T} + 2\mathbf{A}\mathbf{T} + \sum_{j\to k}\sum_i \mathbf{R}_{j,k}^2$$

this is achieved by solving the linear equation system:

$$\mathbf{B}\mathbf{T} + \mathbf{A} = 0$$

Here, $\mathbf{B}$ is given by:

$$\mathbf{B}_{j,j} = \sum_{\substack{j\to k \\ k\to j}} \mathbf{I}_{3\times 3}$$

case $j < k$:

$$\mathbf{B}_{j,k} = -\sum_{j\to k} \mathbf{I}_{3\times 3}$$

case $j > k$:

$$\mathbf{B}_{j,k} = -\sum_{j\to k} \mathbf{I}_{3\times 3},$$

and $\mathbf{A}$ by:

$$\mathbf{A}_j = \sum_{j \to k} \mathbf{R}_j \mathbf{c}_m - \mathbf{R}_k \mathbf{c}_d - \sum_{k \to j} \mathbf{R}_j \mathbf{c}_m - \mathbf{R}_k \mathbf{c}_d.$$

4.3 Uncertainty-based Global Registration

Under the assumption that two poses $\mathbf{X}'_j$ and $\mathbf{X}'_k$ are related by the linear error metric $\mathbf{E}'_{j,k}$ we wish to minimize the Mahalanobis distance that describes the global error of all the poses:

$$\begin{aligned} W &= \sum_{j \to k} (\bar{\mathbf{E}}_{j,k} - \mathbf{E}'_{j,k})^T \mathbf{C}_{j,k}^{-1} (\bar{\mathbf{E}}'_{j,k} - \mathbf{E}'_{j,k}) \\ &= \sum_{j \to k} (\bar{\mathbf{E}}_{j,k} - (\mathbf{X}'_j - \mathbf{X}'_k)) \mathbf{C}_{j,k}^{-1} (\bar{\mathbf{E}}'_{j,k} - (\mathbf{X}'_j - \mathbf{X}'_k)). \end{aligned} \tag{18}$$

The error between two poses is modeled by the Gaussian distribution $(\bar{\mathbf{E}}_{j,k}, \mathbf{C}_{j,k})$. In matrix notation, W becomes:

$$W = (\bar{\mathbf{E}} - \mathbf{H}\mathbf{X})^T \mathbf{C}^{-1} (\bar{\mathbf{E}} - \mathbf{H}\mathbf{X}).$$

Here $\mathbf{H}$ is the signed incidence matrix of the pose graph, $\bar{\mathbf{E}}$ is the concatenated vector consisting of all $\bar{\mathbf{E}}'_{j,k}$ and $\mathbf{C}$ is a block-diagonal matrix comprised of $\mathbf{C}_{j,k}^{-1}$ as submatrices. Minimizing this function yields new optimal pose estimates. The minimization of $\mathbf{W}$ is accomplished via the following linear equation system:

$$\begin{aligned} (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H}) \mathbf{X} &= \mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{E}} \\ \mathbf{B}\mathbf{X} &= \mathbf{A}. \end{aligned}$$

The matrix $\mathbf{B}$ consists of the submatrices

$$\mathbf{B}_{j,k} = \begin{cases} \sum_{k=0}^{n} \mathbf{C}_{j,k}^{-1} & (j = k) \\ \mathbf{C}_{j,k}^{-1} & (j \neq k). \end{cases}$$

The entries of $\mathbf{A}$ are given by:

$$A_j = \sum_{\substack{k=0 \\ k \neq j}}^{n} \mathbf{C}_{j,k}^{-1} \bar{\mathbf{E}}_{j,k}.$$

In addition to $\mathbf{X}$, the associated covariance of $\mathbf{C}_X$ is computed as follows:

$$\mathbf{C}_X = \mathbf{B}^{-1}$$

The actual positional error of two poses $\mathbf{X}_j$ and $\mathbf{X}_k$ is not linear:

$$E_{j,k} = \sum_{i=1}^{m} \|\mathbf{X}_j \oplus \mathbf{d}_i - \mathbf{X}_k \oplus \mathbf{m}_i\|^2 = \sum_{i=1}^{m} \|\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k)\|^2 .$$

Analogous to the simple 2-scan case the linearized pose difference $\mathbf{E}'_{j,k}$ is obtained by use of a Taylor expansion of $\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k)$:

$$\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k) \approx \mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k) - \left(\nabla_{\mathbf{X}_j}\mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k)\Delta\mathbf{X}_j - \nabla_{\mathbf{X}_k}\mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k)\Delta\mathbf{X}_k\right).$$

Here, $\nabla_{\mathbf{X}_j}$ refers to the derivative with respect to $\mathbf{X}_j$. Utilizing the same matrix decomposition $\mathbf{M}_i\mathbf{H}$ of $\nabla\mathbf{Z}_i(\bar{\mathbf{X}})$ as in the 2-scan case $\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k)$ is approximated as:

$$\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k) \approx \mathbf{Z}_i(\bar{\mathbf{X}}_j, \bar{\mathbf{X}}_k) - \mathbf{M}_i\mathbf{E}'_{j,k},$$

where $\mathbf{E}'_{j,k}$ is the linear error metric given by:

$$\begin{aligned}\mathbf{E}'_{j,k} &= (\mathbf{H}_j\Delta\mathbf{X}_j - \mathbf{H}_k\Delta\mathbf{X}_k)\\ &= (\mathbf{X}'_j - \mathbf{X}'_k).\end{aligned}$$

$\mathbf{E}'_{j,k}$ is linear in the quantities $\mathbf{X}'_j$ that will be estimated by the algorithm. Again, the minimum of $\mathbf{E}'_{j,k}$ and the corresponding covariance are given by

$$\begin{aligned}\bar{\mathbf{E}}_{j,k} &= (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}\\ \mathbf{C}_{j,k} &= s^2(\mathbf{M}^T\mathbf{M}).\end{aligned}$$

Here $\mathbf{Z}$ is the concatenated vector consisting of all $\mathbf{Z}_i = \bar{\mathbf{X}}_j \oplus \mathbf{d}_i - \bar{\mathbf{X}}_k \oplus \mathbf{m}_i$.

Note that the results have to be transformed in order to obtain the optimal pose estimates, just like in the simple 2-scan case.

$$\begin{aligned}\mathbf{X}_j &= \bar{\mathbf{X}}_j - \mathbf{H}_j^{-1}\mathbf{X}'_j,\\ \mathbf{C}_j &= (\mathbf{H}_j^{-1})\mathbf{C}_j^X(\mathbf{H}_j^{-1})^T.\end{aligned}$$

5 Summary of n-Scan Registration Methods

In the previous section we have derived different global registration methods. We assumed to have a network of overlapping scans. All methods have in common that the problem is reduced to solving a system of linear equations. Table 1 summarizes the computational complexity of the presented methods. It presents the number of linear equations to be solved. Note: With growing 3D scene sizes the constants usually hidden in the Landau notation are important

Table 1
Overview of the complexity of n-Scan registration methods. n denotes the number of 3D scans.

Registration Algorithm	Number of Equations
Helix Transform	$(6n)^3$
Small Angle Approximation	$(3n)^3 + (3n)^3$
Uncertainty-Based (Euler)	$(6n)^3$
Uncertainty-Based (Quaternion)	$(7n)^3$

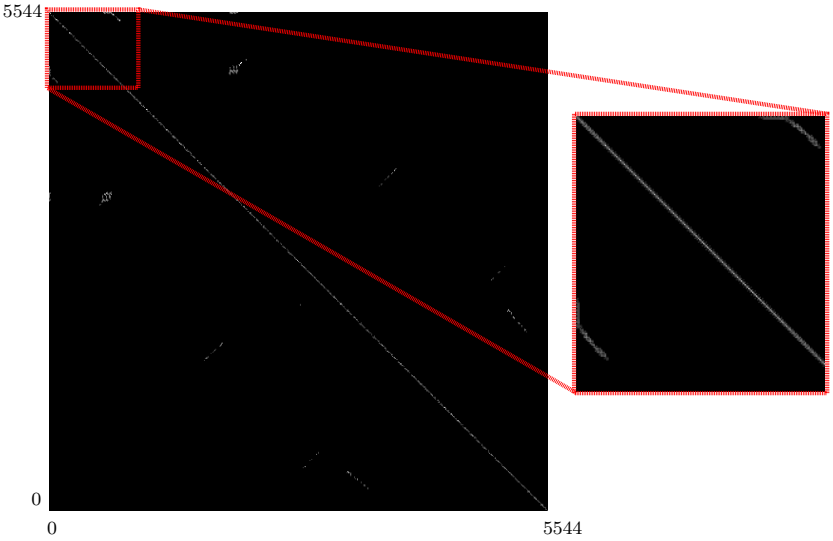


Fig. 5. Sparse matrix of the linear system of equations. Matrix entries that are not zero are printed in white. This matrix occurs when optimizing over all 3D scans of the scene presented in Fig. 1.

in practical applications.

The appendix contains the proofs that the resulting linear system can be solved by a Cholesky decomposition, since it is positive definite. Furthermore, in practical applications it turns out that the matrix is sparse (cf. Fig. 5). Therefore, algorithms for solving sparse systems, i.e., the sparse Cholesky decompositions are applicable [14].

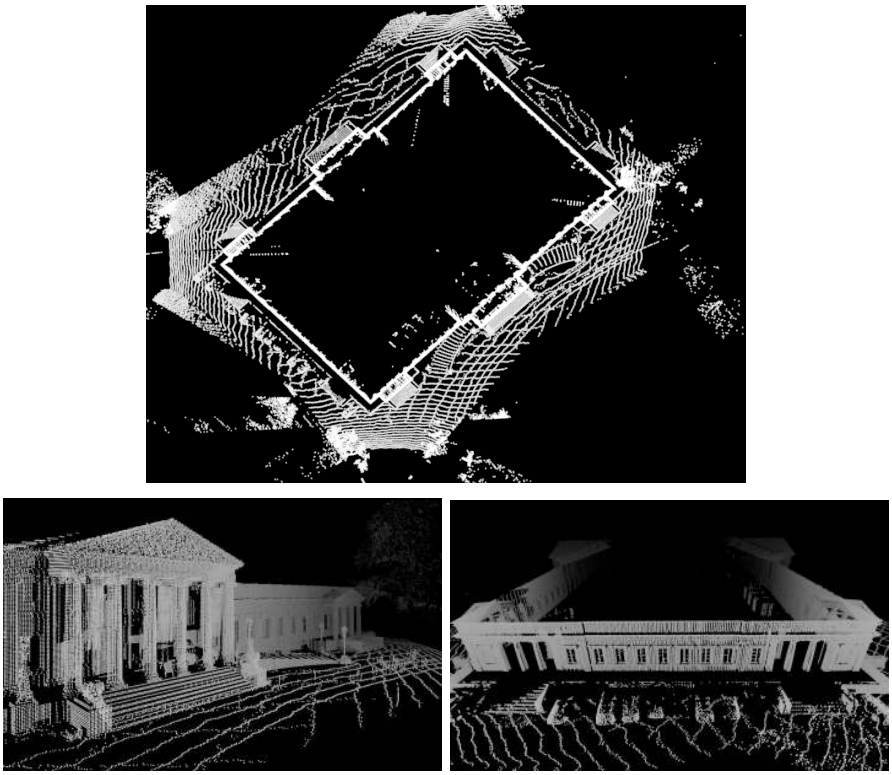


Fig. 6. The data set Rosenstein palace. Top: Bird eye view. Bottom: 3D views.

6 Experiments and Results

6.1 Registration of Terrestial 3D Laser Scans

Initial tests have been carried out with data obtained from Rosenstein palace, near Stuttgart using a Leica HDS 3000 laser scanner [40]. The data set consists of 5 3D scans covering all sides of the palace. After a visual inspection of the scan matching result, e.g., as given in Fig. 6 we made the conclusion that all the linearization methods are in principle able to correctly match 3D scans globally.

Next, we aim to systematically evaluate 3D scan matching as it appears in practical applications. Unfortunately, a simulated evaluation of the minimization step, i.e. using a randomly generated point cloud as a source for several 3D scans with small errors in their poses is infeasible. The usual procedure is to minimize the error metric in the proposed fashion using perfect point correspondences, thus eliminating any error introduced by incorrect point pairs.

Since locally accumulating minimization errors lead to incorrect correspondences, this unfairly rewards local approaches while global registration algorithms are penalized.

6.2 Systematic Tests in an Urban Environment

These experiments have been made using the data of the Robotic 3D Scan Repository [53]. Implementations of the four methods can be found in [55]. The data set Hannover2 (cf. Fig. 2) has been acquired in an urban area and contains 924 3D scans each containing up to 35000 3D data points. It was acquired by a robot carrying a continuously rotating 3D scanner [49]. For the data set Hannover2 ground truth data in form of a 2D map provided by the German land registry office (Katasteramt) is available. This map contains the buildings with a precision of 1 cm. This quality was ensured at the time the map was generated by geodesy using conventional surveying equipment. In addition, we obtained airborne based 3D data. The accuracy of the airborne data is in the cm range, too, since a precise DGPS and IMU in combination with a 2D airborne scanner has been used. Based on this data so-called reference data is generated as follows (see Fig. 7): The 2D map contains lines representing buildings and it is extrapolated to 3D by creating 3D points, i.e., for every line, we generate 3D points from the ground level up to 10 meters with a 25 cm discretization. These 3D points are fused with the 3D data from the airplane. The result is a precise 3D reference map. Using this 3D reference map, we generate ground truth poses for all 924 3D laser scans by matching the scans with the reference map. To these poses we will refer to as "ground truth".

6.2.1 Registration of two Scans

We analyzed the registration of two 3D scans to ensure our methods work correctly. Fig. 8 shows the convergence of the ICP algorithm using the different minimization algorithms. The helix transform, the small angle approximation, and the uncertainty based optimization using Euler angles behave like the closed form solutions, i.e., the value of the error function Eq. (1) is identical to the closed form solution. Thus the convergence speed and the final pose are equal. The convergence graphs from the remaining 923 registrations are similar to the one presented in Fig. 8. We conclude, that the step length computed by our linearized methods are close to the optimal step length. For the experiments we used a maximal point-to-point distance threshold of 25 cm. However, the quaternion based method shows slightly different behaviour, which is due to the fact that it requires normalization afterwards.

Fig. 7. Top left: Schema of the airborne based acquisition of reference data. Top right: 3D map consisting of aerial laser data and extrapolated 2D reference data. Bottom: Airborne and 3D map (black) with superimposed 3D scans (grey).

Table 2 shows the computational requirements for our linear solutions in comparison with the SVD [1] and quaternion non-linear solution [25]. The runtime is averaged over ten runs over the whole data set (5 scans Rosenstein palace, 924 scans HANNOVER2). The closed form solutions are slightly outperformed by the linear ones, since only a linear system of equations has to be solved using the Choleskey decomposition. Since Uncertainty-Based (Euler) computation requires the computation of sine and cosine it is slightly less perfomant.

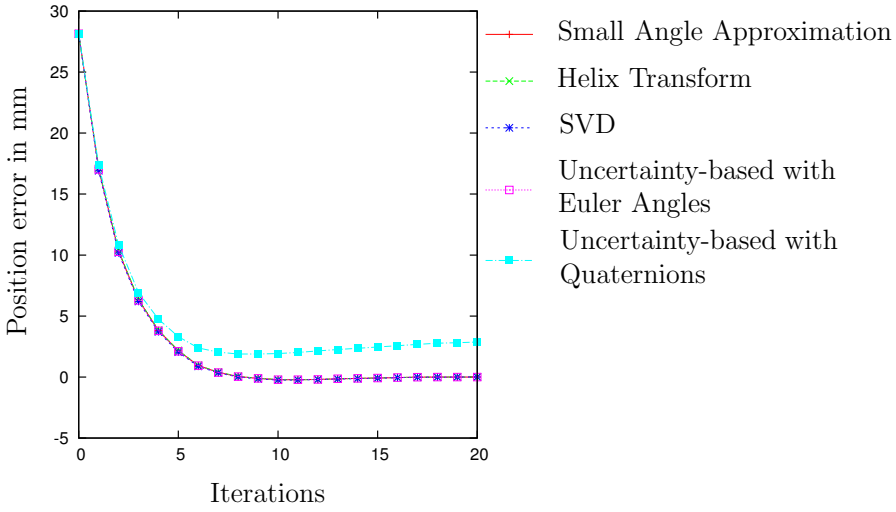


Fig. 8. Convergence of the registration of two scans using different algorithms.

Table 2
Run time comparison (Intel(R) Q9450 @ 2.66GHz, single threaded) for our linear solutions for the 2-Scan problem in comparison to closed form solutions. The timing for Rosenstein palace have been averaged oover 4 3D-scan registrations. The data of the set HANNOVER2 were reduced using an octree with edge-length 10 cm and averaged over 923 scan registrations. Significant run time improvements are obtained in the n scan case (see also Table 1 and Table 3).

Registration Algorithm	Run time (Rosenstein palace)	Run time (HANNOVER2)
Helix Transform	2.232 sec	0.1407 sec
Small Angle Approximation	2.230 sec	0.1402 sec
Uncertainty-Based (Euler)	2.437 sec	0.1581 sec
Uncertainty-Based (Quaternion)	2.388 sec	0.1450 sec
Closed form (SVD) [1]	2.502 sec	0.1803 sec
Closed form (Quaternion) [25]	2.475 sec	0.1613 sec

6.2.2 Registration of n-Scans

We compared the incrementally used ICP algorithm by matching a sequence of 924 3D scans (A video is given using the following link: http://plum.eecs.jacobs-university.de/submissions/large_slam.mpg). Here we matched every 3D scan against its predecessor and take into account that errors sum up. After we returned with the scanner to the origin a loop was closed. We applied the global relaxation and analyzed the error function minimization. Fig. 9 and Fig. 10 presents the results for two 3D scans (No. 80 and No. 150).

The Figures show the position error of these scans (top) that was computed using the difference to the reference position and the orientational error (bottom) that was computed by first calculating the rotation needed to map the scan matching result to the reference orientation and then computing the axis-angle representation. The resulting angle describes the made error. The figures report the error for two loop closings, i.e., the first relaxation was up to iteration 150, the second one up to iteration 300. In Fig. 9 the optimization starts with a correct guess, and the scans are moved to gain a better mutual fit, but relocates them away from ground truth. The ground truth poses are *unknown* to the optimization process. In Fig. 10, we see that all methods converge to stable different minima and with the exception of the quaternion algorithm, the final minimum is close to the ground truth.

The first loop uses the first 150 3D scans. Scan No. 80 (cf. Fig. 9) resides in the middle of the loop. Following the analogy of the spring system (cf. Fig. 3) scans in the middle of the loops are moved only very slowly, since the main inconsistencies occur at the loop closing, e.g., at Scan No. 150 (cf. Fig. 10) and this needs to be propagated through the whole spring system.

Fig. 11 and Fig. 12 show the precision of two 3D scans that reside in the second loop closing. Two methods, namely the small angle approximation and the uncertainty based quaternion solution show convergence to incorrect minima for scan No. 200, while for scan No. 384 only the quaternion solutions seams to be incorrect. Note that the error of the small angle approximation is due to misalignment errors on the y-coordinate, i.e., the height of the scans.

The best performance in terms of position accuracy is achieved by the helix transform and the uncertainty based optimization using Euler angles (cf. Fig. 13 and table 3). As expected, globally consistent scan matching reduces the position error at the positions, where the loop is closed, e.g., Fig. 13 at scan index 100, where the first loop was closed and at the indices 300–400 and 600–700. The locally consistent reference method shows increasing error.

The rotational error is low for all relaxation methods. The reason for this is that the used starting guesses for optimization are already quite good as they are produced by the locally applied ICP algorithm and the IMU starting guess. Please refer to [9] for more details for the system overview. The rotational errors have a large variation range and are therefore hard to judge.

Run times for the benchmarking data set are also given in table 3. In terms of run time, the small angle approximation outperforms all other methods due to the small number of equations in the linear system (see table 1) and the simple computing scheme for the matrix entries. The uncertainty-based methods are not so performant due to the additionally required computations like the transformation of the poses with matrix $\mathbf{H}$. Please note, the overall

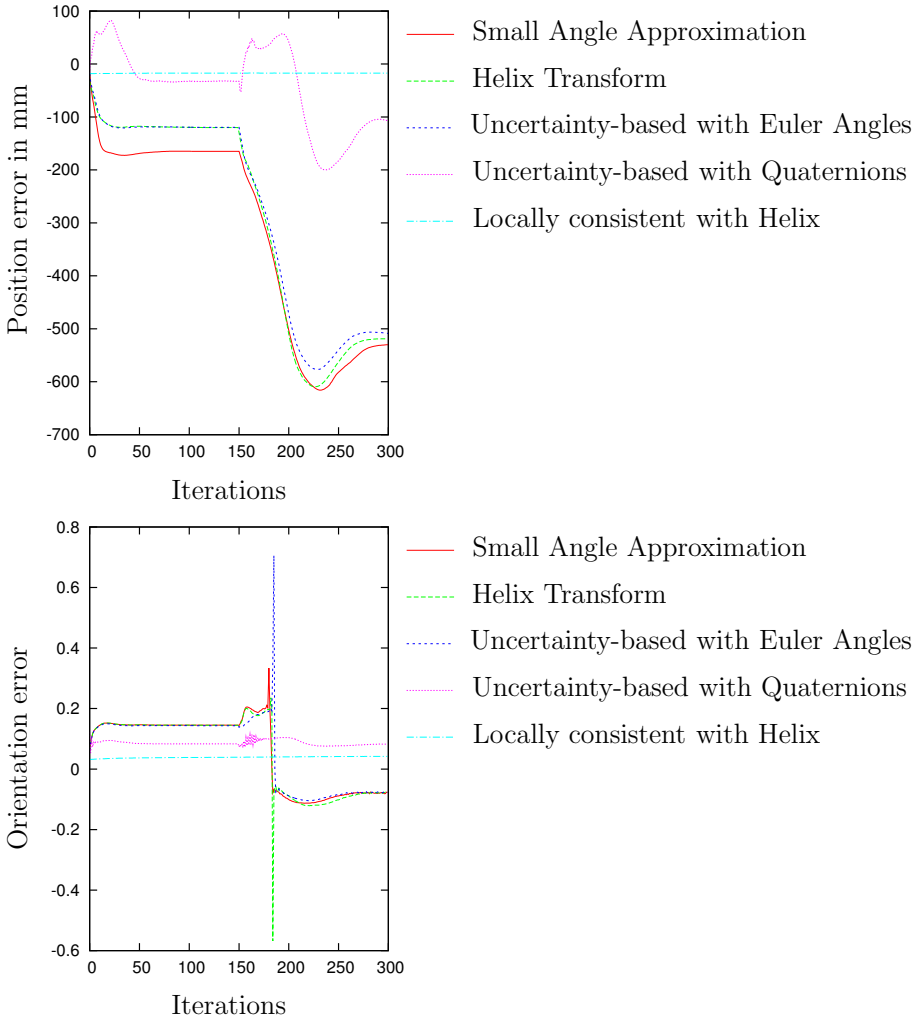


Fig. 9. Convergence of the registration of scan number 80 (top: translational error, bottom: rotational error) using different minimization algorithms. The first 150 iterations correspond to the first relaxation (trajectory A-B-C-D-A-B) while the second 150 iterations represent the result after registration of 384 3D scans (trajectory A-B-C-D-A-B-E-F-A).

run time is dominated by the search for closest points, which is not given in the table.

We have made experiments with iterating the minimization of the global error function, without re-searching for closest points. This approach is used by bundle adjustment [46] to find a good minimum. Since no major pose changes are computed further iterations, we conclude, that the computed step length

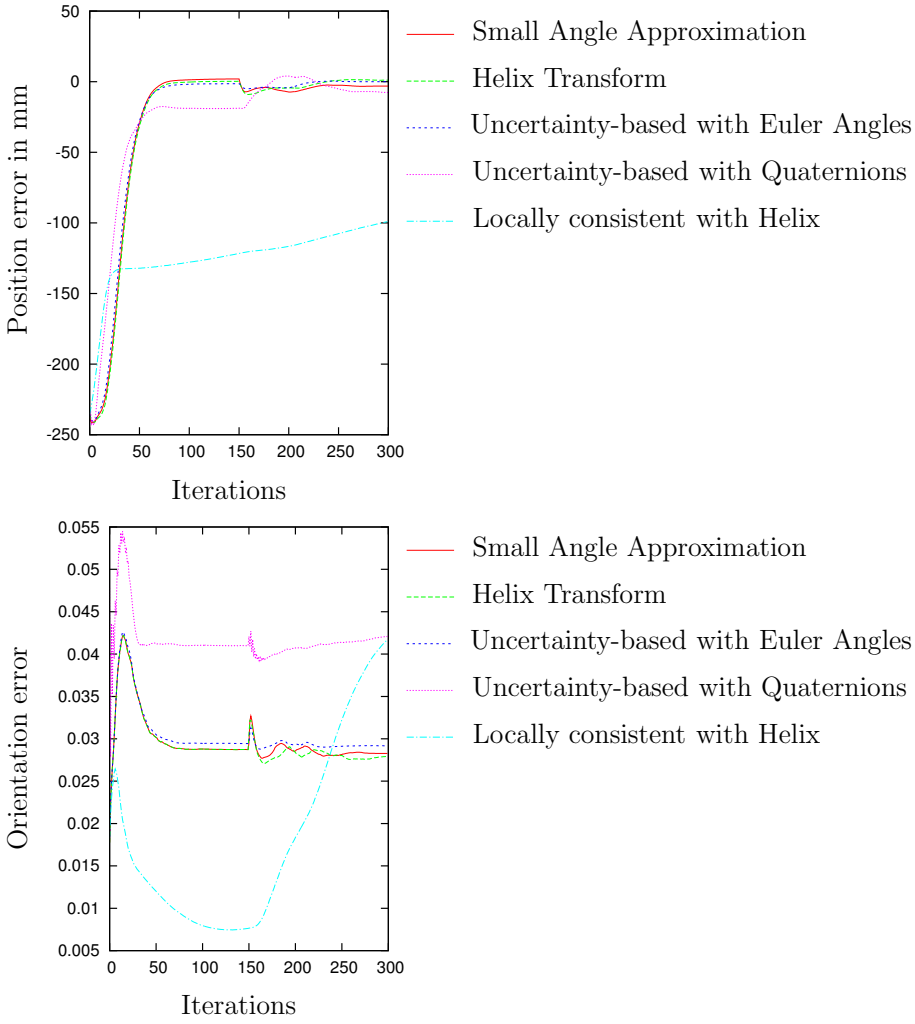


Fig. 10. Same as Fig. 9 but for scan number 150 using the different minimization algorithms.

is close to the optimal one. The one-step solution using new closest points is more effective.

Additional experiments for the registration of n scans with other data sets provided by the Robotic 3D Scan Repository [53] support the statement, that the helix transform and the uncertainty based optimization using Euler angles perform best.

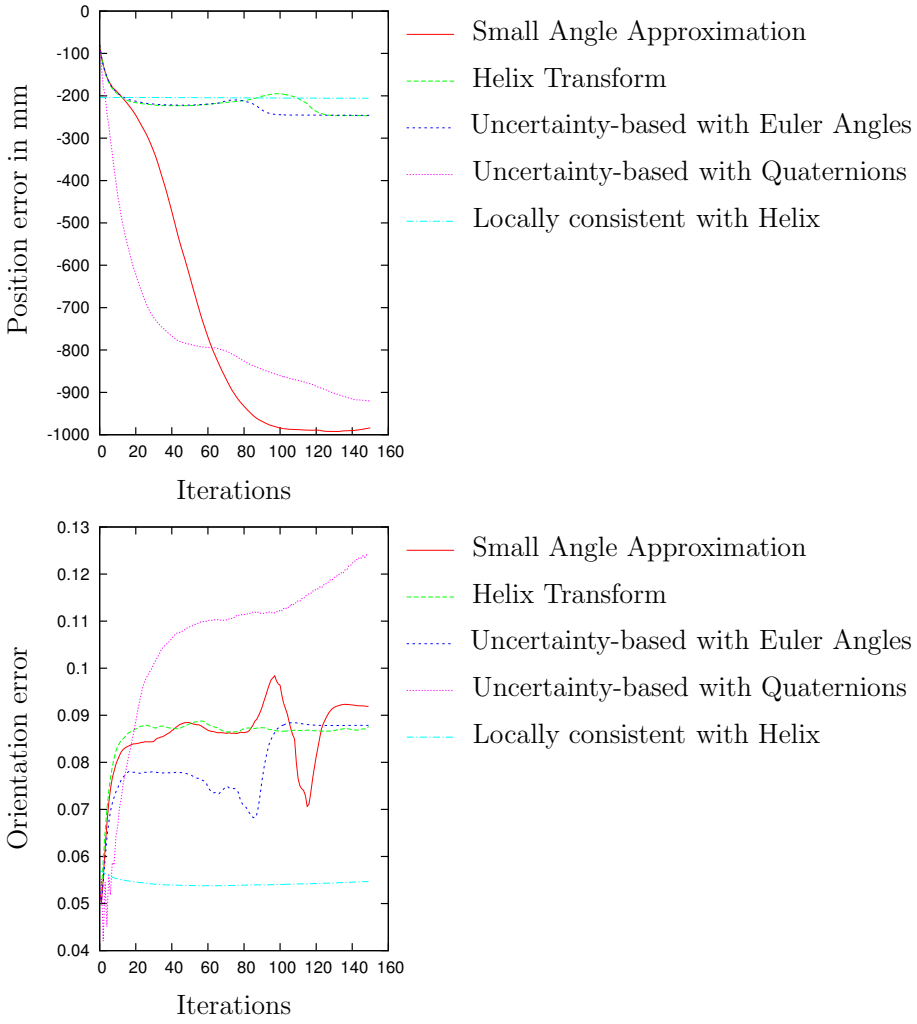


Fig. 11. Convergence of the registration of scan number 200 (top: tranlational error, bottom: rotational error) using different minimization algorithms. These 150 iterations correspond to the second relaxation (sequence: A-B-C-D-A-B-E-F-A).

6.3 Interpretation of the Results

The experiments show that global registration of many 3D scans yield a complex and fragile optimization system. Small variances in the calculated matrices yield different closest point pairs in the following iteration, which in turn result in different matrices. Most stable results with respect to the final pose estimates of the 3D scans can be reported for the approximation of the error function using the helix transform and the uncertainty based optimization

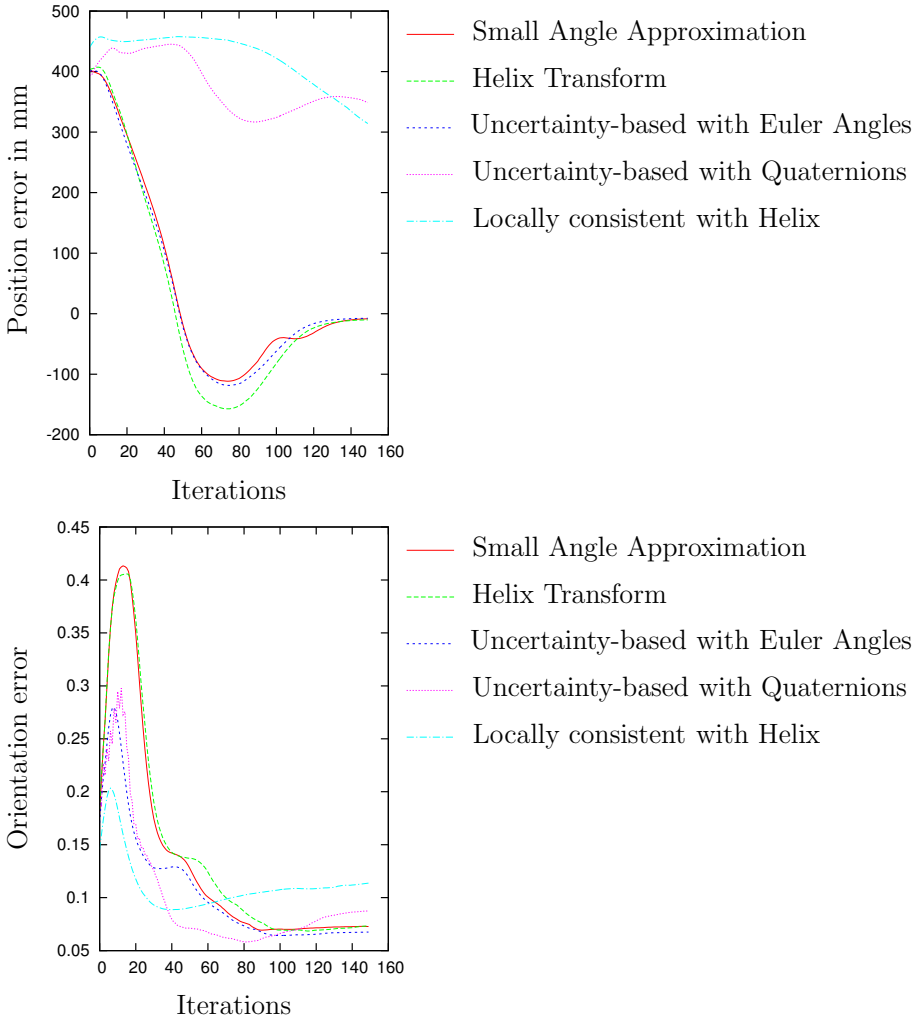


Fig. 12. Same as Fig. 11 but for scan number 384 using the different minimization algorithms.

using Euler angles. The quaternion based approach needs a renormalization step to compute orthonormal rotation matrices, i.e., it leaves the group of rotations. Thus, it is most likely to fail and we don't recommend using it. Table 4 shows our judgment of the difficulty to deriving the linearization and implementing it.

All minimization algorithms deform the map while optimizing, such that the outer parts of the map, i.e., the height of the regions close to C, L and I are raised. After several optimization steps the map converges due to the iterative fashion of the algorithm. Linearization usually has the effect that

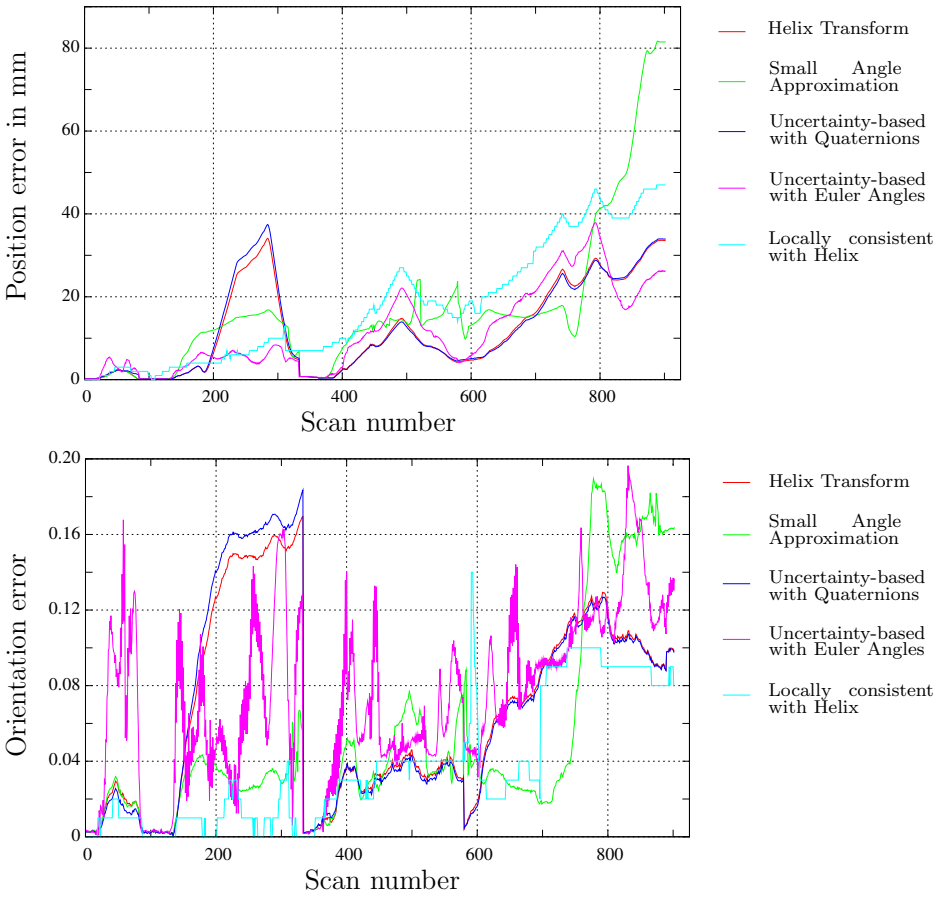


Fig. 13. Final error, after sequential registration of all 3D scans and globally consistent scan matching (distance to ground truth positions (top) and rotational error (bottom). (see Fig. 1 and the corresponding video.)

mapped loops are too large [18]. The local optimal reference method do not make major changes to the locations of scans and the result using this method largely depends on the quality of initial pose guesses (see also Fig. 3, left).

Our experiments have shown that the helix transform performs qualitatively as good as the uncertainty-based algorithm using Euler angles. Thus, we now propose a novel algorithm that combines the positive aspects, i.e., we develop an uncertainty-based helix registration method.

Table 3
Run times and final summed error over all 924 3D scans. Run times apply only for the SLAM back-end and were taken on an Intel(R) Q9450 @ 2.66GHz, using single threaded computations.

Registration Algorithm	Run time	Summed Position Error	Summed Orientation Error
Helix Transform	44.134 sec	10418,54	6263.81
Small Angle Approximation	34.001 sec	11026.47	6235.12
Uncertainty-Based (Euler)	50.459 sec	10819.15	6133.44
Uncertainty-Based (Quaternion)	84.592 sec	15261.31	4635.24
Locally consistent with Helix	40.103 sec	16411.85	3813.83

Table 4
Difficulty of deriving and implementing a global optimal solution. Straightforward calculations and implementations are marked with "+", while more difficult approaches are marked with "−".

Registration Algorithm	Deriving the Solution	Implementation Simplicity
Helix Transform	+	+
Small Angle Approximation	++	++
Uncertainty-Based (Euler)	−−	+
Uncertainty-Based (Quaternion)	−	+

7 An Uncertainty-based Registration using the Helix Transform

The detailed analysis above has motivated the idea of an uncertainty-based algorithm using the helical motion. As with the uncertainty-based registration with Euler angles or quaternions the Mahalanobis distance Eq. (18) is minimized in order to compute the global pose estimates and their covariances. The minimum is given as the solution to the linear equation system $\mathbf{BX} = \mathbf{A}$, where $\mathbf{B}$ and $\mathbf{A}$ are constructed with the quantities $\bar{\mathbf{E}}_{j,k}$ and $\mathbf{C}_{j,k}^{-1}$ as in Section 4.3. $\bar{\mathbf{E}}_{j,k} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}$ and $\mathbf{C}_{j,k}^{-1} = s^2(\mathbf{M}^T\mathbf{M})$ are calculated according to the corrected values for $\mathbf{M}$ and $\mathbf{Z}$ that follow from the derivation for the Helix transform error metric:

$$\begin{aligned} E &= \sum_{j\to k}\sum_{i} \left\|\mathbf{m}_i - \mathbf{d}_i + (\bar{\mathbf{x}}_j + \mathbf{x}_j \times \mathbf{m}_i) - (\bar{\mathbf{x}}_k + \mathbf{x}_k \times \mathbf{m}_i)\right\|^2 \\ &= \sum_{i=1}^{m} \left\|\mathbf{Z}_i(\mathbf{X}_j, \mathbf{X}_k)\right\|^2 . \end{aligned}$$

Since the derivation is identical to the one in Section 4.3 we give the matrix decomposition that is central for calculating $\bar{\mathbf{E}}_{j,k}$ and $\mathbf{C}_{j,k}^{-1}$:

$$\mathbf{M}_i = \begin{pmatrix} 0 & m_{z,i} & -m_{y,i} & 1 & 0 & 0 \\ -m_{z,i} & 0 & m_{x,i} & 0 & 1 & 0 \\ m_{y,i} & -m_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix}$$
$$\mathbf{H} = \mathbf{I}_{6\times 6}.$$

Comparing both helix transform based registrations reveals that the only difference is that the uncertainty based registration introduces the value s^2, see Eq. 15, which scales the covariances. For a constant s over all pairs or the simple 2-scan case the solution given by the non uncertainty-based registration is identical to the one without using uncertainties.

8 Conclusions and Outlook

This paper contained three major contributions: Firstly, we summarized existing locally and globally optimal scan matching methods and extended them by a working global helix-based registration, by a small angle approximation, and an uncertainty-based registration using the helix transform. All methods approximate a closed-form solution for iterative closest point algorithms. As our experiments have demonstrated, it is not recommended to use the uncertainty-based registration using quaternions, since it does not stay in the $SO(3)$ group and requires re-normalization. Secondly, we discussed the run time of the algorithms by stating the number of equations that have to be solved. Thirdly, we exhaustively evaluated the global scan matching by comparing scan poses with a genuine truth and discussed the outcomes. A minor contribution of the paper is that we showed the linear system of equations is for the registration using the helix transform is equal to the system one derives when using the small angle approximation. Another minor contribution is that we have demonstrated that the use of the helix transform in a probabilistic context changes the system of equations only by a scaling factor.

The presented algorithms establish the foundation for mapping in 3D using 3D laser scanners. The applications are large scale systems ranging from terrestrial laser scanning, survey and geodesy to robotic mapping. In the latter research field probabilistic methods are state of the art [42]. This paper has showed, that deterministic modelings of the scan registration problem, i.e. by using the helix transform, yield the same performance as probabilistic ones, i.e., uncertainty based optimization using Euler angles or differ only about a constant factor, for example the uncertainty based optimization using the helix transform.

Needless to say a lot of work remains to be done. In future work we are aiming at finding a closed form solution to get rid of the linearization and the problems due to it. Subsection 3.1 outlines the math for the two scan case. The open question remains, if such methods are extendable to multiple scan registrations. Initial progress is available here by Yguel, who derived a closed form solution for the 2-scan registration problem, incorporating the Mahalanobis distance [50].

Furthermore, in future work we plan to apply the proposed algorithms to large scale experiments, i.e., to 3D mapping of cities. Then, the back-end, i.e., solving the system of linear equations becomes the bottleneck, while it is currently the front-end search for closest points, i.e., $n \ll N$, where n is the number of poses and N is the number of 3D points per scan. This will need to include recent results from the SLAM community, e.g., the work presented in [17, 34, 35] that aim to reduce the run time of the SLAM back-end. In this paper we used CSPARSE, a sparse Cholesky decomposition, to speed up the matrix operations [10, 14].

Acknowledgments

We would like to thank Dorit Borrmann (University of Osnabrück) for jointly implementing the uncertainty based registration methods and Kai Lingemann and Joachim Hertzberg (both University of Osnabrück) for supporting our work. We thank Jan Böhm from University of Stuttgart for providing the data set Rosenstein palace. Furthermore, the authors thank Oliver Wulf and Bernardo Wagner (both Leibniz University of Hannover, Germany) for making the 3D data set Hannover2 publicly available and Claus Brenner (Leibniz University of Hannover) for letting us use the airborne 3D data.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698 – 700, 1987. 9, 27, 28

[2] T. Barrera, A. Hast, and E. Bengtsson. Incremental spherical linear interpolation. In *SIGRAD*, volume 13, pages 7–13, 2004. 2

[3] R. Benjemaa and F. Schmitt. Fast Global Registration of 3D Sampled Surfaces Using a Multi-Z-Buffer Technique. In *Proceedings IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '97)*, Ottawa, Canada, May 1997. 7

[4] R. Benjemaas and F. Schmitt. A Solution For The Registration Of Multiple 3D Point Sets Using Unit Quaternions. *Computer Vision – ECCV '98*, 2:34 – 50, 1998. 7

[5] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multi-view registration technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(5):540 – 547, May 1996. 7

[6] P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992. 5

[7] P. Biber, H. Andreasson, T. Duckett, and A. Schilling. 3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, Sendai, Japan, September 2004. 7

[8] J. Böhm and S. Becker. Automatic marker-free registration of terrestrial laser scans using reflectance features. In *Proceedings of 8th Conference on Optical 3D Measurment Techniques*, pages 338–344, Zurich, Switzerland, July 2007. 6

[9] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally consistent 3d mapping with scan matching. *Journal of Robotics and Autonomous Systems (JRAS)*, 56(2):130–142, February 2008. 14, 29

[10] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. The Efficient Extension of Globally Consistent Scan Matching to 6 DOF. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '08)*, Atlanta, GA, USA, June 2008. 14, 37

[11] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '91)*, pages 2724 – 2729, Sacramento, CA, USA, April 1991. 5

[12] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Image Vision Comput.*, 10(3):145 – 155, 1992. 7

[13] S. Cunnington and A. Stoddart. N-View Point Set Registration: A Comparison. In *Proceedings of the 10th British Machine Vision Conference (BMVC '99)*, Nottingham, UK., 1999. 2, 7

[14] T. A. Davis. Algorithm 849: A concise sparse Cholesky factorization package. *ACM Trans. Math. Softw.*, 31(4), 2005. 24, 37

[15] A. J. Davison, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007. 8

[16] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229 – 241, June 2001. 8

[17] U. Frese. Efficient 6-DOF SLAM with Treemap as a Generic Backend. In *Proceedings of the IEEE International Conference on Robotics and Automation ((ICRA '07)*, pages 4814–4819, Rome, Italy, April 2007. 8, 37

[18] U. Frese and G. Hirzinger. Simultaneous Localization and Mapping – A Discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17 – 26, Seattle, USA, August 2001. 34

[19] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transaction on Mathematical Software*, 3(3):209 – 226, September 1977. 6

[20] M. Greenspan and M. Yurick. Approximate K-D Tree Search for Efficient ICP. In *Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '03)*, pages 442 – 448, Banff, Canada, October 2003. 6

[21] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3472–3478, 2007. 2, 8

[22] C. Hertzberg. A framework for sparse, non-linear least squares problems on manifolds. Master's thesis, Universität Bremen, Bremen, Germany, November 2008. 2

[23] M. Hofer. Personal communication, 7. July 2008. 18

[24] M. Hofer and H. Pottmann. Orientierung von Laserscanner-Punktwolken. *Vermessung & Geoinformation*, 91:297 – 306, 2003. 10, 11, 12, 13

[25] B. K. P. Horn. Closed–form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629 – 642, April 1987. 10, 27, 28

[26] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed–form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127 – 1135, July 1988. 9, 10

[27] S. Krishnan, P. Y. Lee, J. B. Moore, and S. Venkatasubramanian. Global Registration of Multiple 3D Point Sets via Optimization on a Manifold. In *Eurographics Symposium on Geometry Processing*, 2000. 7

[28] A. Lorusso, D. Eggert, and R. Fisher. A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations. In *Proceedings of the 4th British Machine Vision Conference (BMVC '95)*, pages 237 – 246, Birmingham, England, September 1995. 2, 8

[29] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. In *Autonomous Robots*, volume 4, pages 333–349, 1997. 14, 16

[30] M. Magnusson and T. Duckett. A Comparison of 3D Registration Algorithms for Autonomous Underground Mining Vehicles. In *Proceedings of the Second European Conference on Mobile Robotics (ECMR '05)*, pages 86 – 91, Ancona, Italy, September 2005. 7

[31] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. Registration of point cloud data from a geometric optimization perspective. In R. Scopigno and D. Zorin, editors, *Eurographics Symposium on Geometry Processing*, pages 23–32, 2004. 2, 6

[32] A. Nüchter, K. Lingemann, and J. Hertzberg. Cached k-d tree search for icp algorithms. In *Proceedings of the 6th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '07)*, pages 419 – 426, Montreal, Canada, August 2007. 6

[33] E. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008. 8

[34] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)*, 2006. 37

[35] L. Paz, J. Tardos, and J. Neira. Divide and Conquer: EKF SLAM in $O(n)$. *IEEE Transactions on Robotics*, (to appear). 37

[36] H. Pottmann, S. Leopoldseder, and M. Hofer. Simultaneous registration of multiple views of a 3D object. *ISPRS Archives*, 34(3A):265 – 270, 2002. 10, 11, 13, 17

[37] H. Pottmann, S. Leopoldseder, and M. Hofer. Registration without ICP. *Computer Vision and Image Understanding (CVIU)*, 95(1):54 – 71, July 2005. 7

[38] K. Pulli. Multiview Registration for Large Data Sets. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling (3DIM '99)*, pages 160 – 168, Ottawa, Canada, October 1999. 7

[39] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modellling (3DIM '01)*, pages 145 – 152, Quebec City, Canada, May 2001. 6

[40] S. Schuhmacher and J. Böhm. Georeferencing of terrestrial laserscanner data for applications in architectural modeling. In *Proceedings 3D-ARCH 2005: Virtual Reconstruction and Visualization of Complex Architectures*, number XXXVI in PART 5/W17, 2005. 25

[41] A. Stoddart and A. Hilton. Registration of multiple point sets. In *Proceedings of the 13th IAPR International Conference on Pattern Recognition*, pages 40–44, Vienna, Austria, August 1996. 7

[42] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. 8, 36

[43] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots*, 31(5):1 – 25, 1997. 8

[44] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, September 2005. 8

[45] S. Thrun, D. Fox, and W. Burgard. A real-time algorithm for mobile robot mapping with application to multi robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, San Francisco, CA, USA, April 2000. 8

[46] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000. 2, 30

[47] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54:358 – 367, November 1991. 10

[48] J. Williams and M. Bennamoun. Multiple View 3D Registration using Statistical Error Models. In *Vision Modeling and Visualization*, 1999. 7

[49] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner. Benchmarking urban six-degree-of-freedom simultaneous localization and mapping. *Journal of Field Robotics (JFR)*, 25(3):148–163, March 2008. 26

[50] M. Yguel. Rigid Transformation Estimation in Point Registration Procedures: Study of the Use of Mahalanobis Distances. Research Report RR-6684, INRIA, 2008. 37

[51] Z. Zhang. Iterative point matching for registration of free–form curves. Technical Report RR-1658, INRIA–Sophia Antipolis, Valbonne Cedex, France, 1992. 5

Internet Ressources

[52] MESA Imaging, http://www.mesa-imaging.ch/, 2008. 2

[53] Robotic 3D Scan Repository. http://kos.informatik.uni-osnabrueck.de/3Dscans/, 2008. 26, 31

[54] Zoller und Fröhlich Laserscanner, http://www.zf-laser.com/, 2008. 6

[55] 6D Simultaneous Localization and Mapping, http://slam6d.sourceforge.net/, 2009. 26

[56] ifm electronic, O3D200 sensor, http://www.ifm.com/ifmuk/web/dsfs!O3D200.html, 2009. 2

[57] PMDTechnologies GmbH, http://www.pmdtec.com/, 2009. 2

Appendix

For all registration algorithms described in this paper, some matrix $\mathbf{B}$ is inverted. Because these matrices can become very large, it is desirable to use fast matrix inversion techniques such as the Cholesky decomposition. In order for this to be effective, the matrix in question needs to be positive definite. The following sections provide proofs of this property for each registration algorithm. In each section the notation for the respective algorithms are utilized.

8.1 Helix Transform

$\mathbf{B}$ is by the definition:

$$\mathbf{X}^T \cdot \mathbf{B} \cdot \mathbf{X} = \sum_{j \to k} \sum_i ((\bar{\mathbf{c}}_j + \mathbf{c}_j \times \mathbf{m}_i) - (\bar{\mathbf{c}}_k + \mathbf{c}_k \times \mathbf{m}_i))^2 \geq 0$$

positive semi definite. Since any n-scan problem includes the link $0 \to 1$ it is sufficient to show

$$\sum_i (\bar{\mathbf{c}}_1 + \mathbf{c}_1 \times \mathbf{m}_i)^2 \neq 0,$$

for all $\mathbf{c}_1, \bar{\mathbf{c}}_1 \neq 0$. The solution space of $\bar{\mathbf{c}}_1 + \mathbf{c}_1 \times \mathbf{m}_i = 0$ is in the form of two lines parallel to $\mathbf{c}_1$. In practice no 3D scan satisfies such a condition. □

8.2 Small Angle Approximation

The solution matrix $\mathbf{B}$ for the rotation is positive semi definite by definition:

$$\mathbf{X}^T \cdot \mathbf{B} \cdot \mathbf{X} = \sum_{j \to k} \sum_i (\mathbf{M}_i \cdot \mathbf{X}_j - \mathbf{D}_i \cdot \mathbf{X}_k)^2 (\mathbf{m}_i - \mathbf{d}_i).$$

It is only necessary to show

$$\sum_i (\mathbf{D}_i \cdot \mathbf{X}_1)^2 > 0$$

or equivalently

$$\sum_i \mathbf{X}_1 \times \mathbf{d}_i \neq 0,$$

for all $\mathbf{X}_1 \neq 0$. This holds true for all $\mathbf{d}_i \neq 0$. □

The matrix $\mathbf{B}$ that is required for the recovery of the translation is defined by:

$$\mathbf{X}^T \cdot \mathbf{B} \cdot \mathbf{X} = \sum_{j \to k} \sum_{i} (\mathbf{t}_k - \mathbf{t}_j)^2.$$

This is always greater than zero for any $\mathbf{t}_k \neq \mathbf{t}_j$. Since $\mathbf{t}_0$ equals zero $\mathbf{B}$ is positive definite. □

8.3 Uncertainty-based Global Registration

First, it is proven that $\mathbf{B}$ is positive definite under the condition that the covariances $\mathbf{C}_{i,j}$ are positive definite. Second, the property of positive definiteness is shown for the covariances of both the Euler and quaternion representation.

Induction base $m = n$: Assuming a graph with $n + 1$ nodes and n links, the matrix $\mathbf{B}$ is transformed into a matrix $\mathbf{B}'$, by

$$\mathbf{B}' = \mathbf{I}_D \mathbf{B} \mathbf{I}_D^T,$$

with an upper-right triangular matrix $\mathbf{I}_D$ of 6-dimensional identity matrices

$$\mathbf{I}_D = \begin{pmatrix} \mathbf{I}_6 & \dots & \mathbf{I}_6 \\ & \ddots & \vdots \\ 0 & & \mathbf{I}_6 \end{pmatrix}$$

Since $\mathbf{B}'$ is given by

$$\begin{aligned} \mathbf{B}'_{j,j} &= C^{-1}_{j-1,j} \\ \mathbf{B}'_{j,k} &= 0 \qquad (j \neq k) \end{aligned}$$

and all covariances are positive definite, $\mathbf{B}'$ itself is positive definite. The same holds for $\mathbf{B}$, as $\mathbf{I}_D$ is invertible.

Inductive step $m \to m + 1$: Let $\mathbf{B}$ be a positive definite matrix that corresponds to a graph with $n + 1$ nodes and m links. An additional link between the nodes $\mathbf{X}_j$ and $\mathbf{X}_k$ is inserted, with the positive definite covariance $\mathbf{C}_{i,j}$. Without loss of generality, $\mathbf{X}_i$ is to be the fixed pose at 0. Thus, the resulting matrix $\mathbf{B}'$ is changed only at submatrix

$$\mathbf{B}'_{k,k} = \mathbf{B}_{k,k} + \mathbf{C}^{-1}_{j,k}.$$

$\mathbf{B}'$ is positive definite, iff

$$\mathbf{X}^T \mathbf{B}' \mathbf{X} > 0 \qquad \mathbf{X} \neq 0,$$

which is equivalent to

$$\sum_{m,l=1}^{n} \mathbf{X}_m^T \mathbf{B}'_{m,l} \mathbf{X}_l > 0 \qquad \mathbf{X}_m \neq 0. \tag{19}$$

Eq. (19) is expanded to

$$\begin{aligned}\sum_{m,l=1}^{n} \mathbf{X}_m^T \mathbf{B}'_{m,l} \mathbf{X}_l &= \mathbf{X}_k^T \mathbf{B}'_{k,k} \mathbf{X}_k + \sum_{\substack{m,l=1 \\ m \neq k \neq l}}^{n} \mathbf{X}_m^T \mathbf{B}_{m,l} \mathbf{X}_l \\ &= \mathbf{X}_k^T \mathbf{C}_{j,k}^{-1} \mathbf{X}_k + \sum_{m,l=1}^{n} \mathbf{X}_m^T \mathbf{B}_{m,l} \mathbf{X}_l \\ &= \mathbf{X}_k^T \mathbf{C}_{j,k}^{-1} \mathbf{X}_k + \mathbf{X}^T \mathbf{B} \mathbf{X} > 0.\end{aligned}$$

$\mathbf{B}'$ is a positive definite matrix. □

Euler angles: The covariance $\mathbf{C}_{j,k}$ is given by

$$\mathbf{C}_{j,k} = s^2 \mathbf{M}^T \mathbf{M}.$$

It can be safely assumed that $s^2 > 0$. Therefore, $\mathbf{C}_{j,k}$ is positive definite iff

$$\mathbf{X}^T \mathbf{M}^T \mathbf{M} \mathbf{X} > 0$$

or simply:

$$\sum_i \mathbf{M}_i \mathbf{X} \neq 0,$$

which is equivalent to:

$$\sum_i \begin{pmatrix} \theta_x \\ \theta_z \\ -\theta_y \end{pmatrix} \times \mathbf{m}_i \neq \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

for $\mathbf{X} \neq 0$. While this is not true iff all points $\mathbf{m}_i$ lie on two parallel lines, $\mathbf{C}_{j,k}$ is positive definite for all real data sets. □

Quaternions: As above, $\mathbf{C}_{j,k}$ is positive definite iff:

$$\mathbf{X}^T \mathbf{M}^T \mathbf{M} \mathbf{X} > 0$$

or simply:

$$\sum_i \mathbf{M}_i \mathbf{X} \neq 0$$

which is equivalent to:

$$\sum_i p\mathbf{m}_i - \begin{pmatrix} q \\ r \\ s \end{pmatrix} \times \mathbf{m}_i \neq \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

for $\mathbf{X} \neq 0$. As above this is true for all real data sets. □

2.3 Point Cloud Processing

An important property of laser scanners is that they produce a large number of measurements in a relatively short period of time. This is especially true for 3D laser scanners that produce several tens of thousands of range measurements per second or more. While this high measurement rate is beneficial for quickly creating dense representations of the environment, the large amount of data also comes at a cost. An average scanning job will typically result in several gigabytes of point cloud data. This data needs to be stored for the long term as well as for processing. For both, reducing the memory footprint of a point cloud is of concern. In addition to low memory consumption, fast access operations are crucial for processing. Modern laser scanners acquire up to and sometimes even more than one million points per second. With three coordinates per point represented by a 4 byte floating point value, this adds up to about 11.4 MB of data per second. The main memory of typical high-end PCs used for processing will be filled within minutes of scanning. Standard compression methods will reduce the file size of raw point clouds considerably. This is often done for storing the data for long term use or for exchanging 3D scans. However, standard compression will make it impossible to quickly process the data. Visualizing point clouds reasonably, requires a way to quickly access the subset of points that are visible on screen. For reconstruction tasks a user often selects structures represented by a small number of points. A manual selection of one or multiple points is usually carried out by a process called ray casting. Fast ray casting is dependant on a suitable representation of 3D space, which also serves as a data structure for storing the points. Of chief concern for this thesis is point access for rigid registration. In this context nearest neighbor queries need to be carried out efficiently.

For these purposes a specialized version of the octree [50] was developed. This version of the data structure was specifically created to be able to losslessly compress point cloud data in runtime memory without loosing the capability to use and process the data itself. In fact, the developed data structure speeds up certain operations on the data.

2.3.1 An Octree for Efficient Processing of 3D Laser Scans

The registration algorithms presented in the previous sections rely heavily on nearest neighbour search. This is by far the most time-consuming step when matching point clouds using ICP and similar algorithms. The following paper presents the octree data structure that was developed for processing, storing, displaying and matching point clouds. It also presents a novel nearest neighbor search algorithm on octrees.

One Billion Points in the Cloud — An Octree for Efficient Processing of 3D Laser Scans

Jan Elseberg, Dorit Borrmann, Andreas Nüchter

Jacobs University Bremen gGmbH
Automation Group, School of Engineering and Science
Campus Ring 1, 28759 Bremen, Germany
j.elseberg|d.borrmann|a.nuechter@jacobs-university.de

Abstract

Automated 3-dimensional modeling pipelines include 3D scanning, registration, data abstraction, and visualization. All steps in such a pipeline require the processing of a massive amount of 3D data, due to the ability of current 3D scanners to sample environments with a high density. The increasing sampling rates make it easy to acquire Billions of spatial data points. This paper presents algorithms and data structures for handling these data. We propose an efficient octree to store and compress 3D data without loss of precision. We demonstrate its usage for an exchange file format, fast point cloud visualization, sped-up 3D scan matching, and shape detection algorithms. We evaluate our approach using typical terrestrial laser scans.

Keywords: octree, tree data structure, data compression, frustum culling, ray casting, RANSAC, nearest neighbor search

1. Introduction

Laser range scanning provides an efficient way to actively acquire accurate and dense 3D point clouds of object surfaces or environments. 3D point clouds provide a basis for rapid modeling in industrial automation, architecture, agriculture, construction or maintenance of tunnels and mines, facility management, and urban and regional planning. Modern terrestrial and kinematic laser scan systems acquire data at an astonishing rate. For example, the Faro Focus^{3D} delivers up to 976,000 3D points per second and the Velodyne HDL-64E yields 1.8 million range measurements per second. Kinematic

laser scan systems often use multiple line scanners and also produce a huge amount of 3D data points. A common way to deal with the data is to process only a small subset of it. While this is an acceptable way of handling the data for some applications it calls into question why so many measurements were acquired in the first place.

In this paper we describe a spatial data structure called an octree. Using innovations from the computer graphics community, we develop an octree implementation with advantageous properties. First, we prefer a data structure that stores the raw point cloud over a highly approximative voxel representation. While the latter one is perfectly justifiable for some use cases, it is incompatible with tasks that require exact point measurements like data visualization and scan matching. Second, the octree ought to be fast, i.e., access, insert and delete operations must be in $O(\log n)$, where n is the number of stored points. Last and most important, the data structure has to be memory efficient. We present an easy to implement octree encoding that fulfills these requirements and is capable of storing one billion points in 8 GB of memory. It is also possible to employ disk caching with the the data structure, i.e., larger data sets can be streamed from a mass storage device when processing it.

For achieving our goal to process one billion points in main memory on a standard computer, we have implemented an efficient data structure for 3D point clouds. In addition to the octree, this paper presents several algorithms exploiting the properties of our data structure. These algorithms include a novel and efficient RANSAC implementation for shape detection and a novel nearest neighbor search algorithm for ICP-based scan matching. Furthermore, we show that our memory efficient encoding is also universal enough to allow for other uses such as fast visualization. All presented algorithms are available under the GPL license in the *3DTK – The 3D Toolkit* and can be downloaded from `http://threedtk.de`. The toolkit contains a small show application that uses the frustum culling and processes 1 billion points while still enabling the user to navigte smoothly through the point cloud. In case more points have to be rendered than the graphics card is able to process in a given time, the point density is reduced dynamically, by sending only a fraction of points to the graphics card. Thus, the user is able to inspect large 3D scenes. Figure 1 shows a scene with dynamic point reduction. The scene was recorded on the campus of the Jacobs University using a Riegl VZ-400 scanner. The data set contains 876,820,018 3D points and is easily processable with our octree implementation. The data set and win-

dows executables are given at at `http://plum.eecs.jacobs-university.de/download/isprs2011/JacobsUni.zip`. To view the 3D point cloud using the described frustum culling on a 64-Bit Windows system, please unzip it and change into the directory `win_x64`. Afterwards, call the program as `show -s 31 -e 80 --loadOct ../dat/`, which loads the provided 50 terrestrial 3D scans (starting with scan number 31 ending with scan number 80) and shows them in a small OpenGL-based viewing program.

2. Octrees for storing 3D point clouds

An octree is a tree data structure that is used for indexing 3-dimensional data. It is the generalization of binary trees and quadtrees, which store one and 2-dimensional data respectively. Each node in an octree represents the volume formed by a rectangular cuboid, often, also in our implementation, simplified to an axis aligned cube. This is analogous to representing a line segment, or rectangle for binary and quadtrees. Consequently an octree node has up to eight children, each corresponding to one octant of the overlying cube/node. A node having no children usually implies that the corresponding volume can be uniformly represented, i.e., no further subdivision is necessary to disambiguate. This convention is not completely applicable when storing points, which are technically dimensionless, i.e., there is no volume associated with them. When storing a point cloud, we must therefore define a stopping rule for occupied volumes. We define both a maximal depth and a minimal number of points as a stopping criteria. If either the maximal depth is exceeded or the number of points is below the given limit leaf nodes, instead of inner nodes, are generated. Defining a maximal depth is equivalent to defining the smallest possible leaf size, also referred to as the voxel size. A list of points is stored in each occupied leaf. By applying two simple criteria we avoid building a perfect octree, i.e., an octree where all leaves are at the same depth and all other nodes have exactly 8 children. First and foremost the uniformity criteria above is applied to volumes without points, such that subdivision is not necessary in empty nodes. In fact, we only create child nodes for octree volumes that contain points. All nodes without children are interpreted as empty space. Second, we do not subdivide a volume further that contains only a single point. Laser scanners sample only the surface of objects, and usually provide only a single distance measurement per angle pair. This leads to a 3-dimensional point cloud that is not fully volumetric.

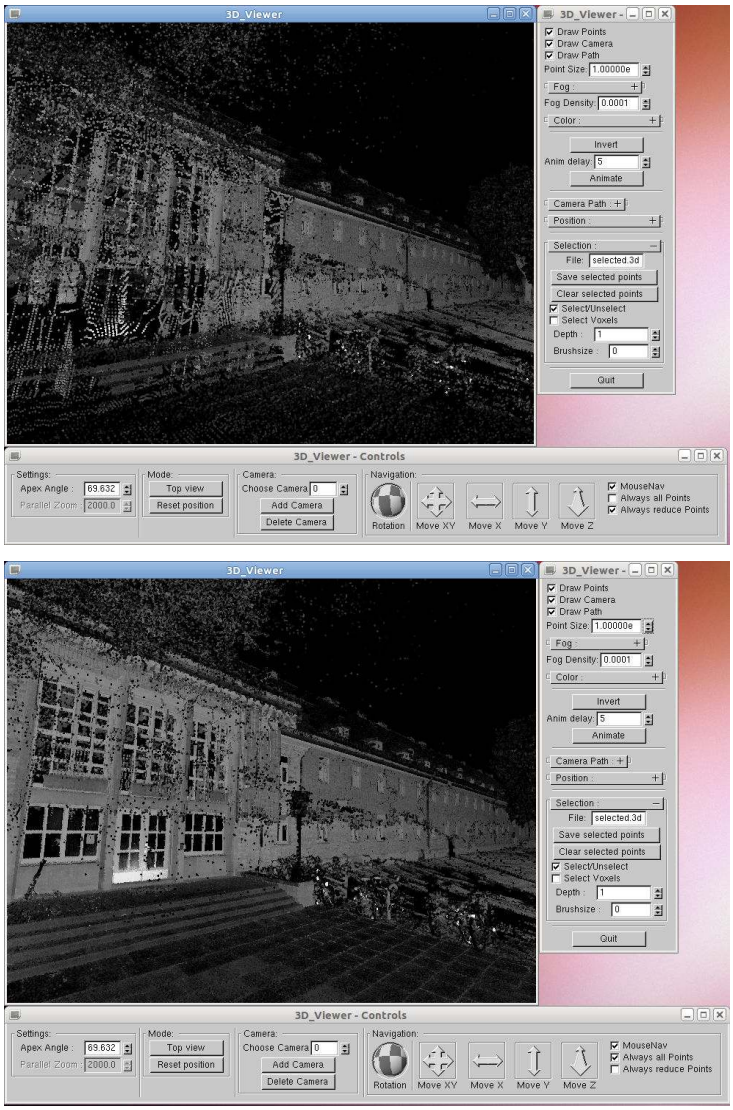


Figure 1: Dynamic point reduction in *3DTK – The 3D Toolkit* (`http://threedtk.de`). Top: reduced point cloud to obtain a frate rate of 20 fps. Bottom: All 3D points.

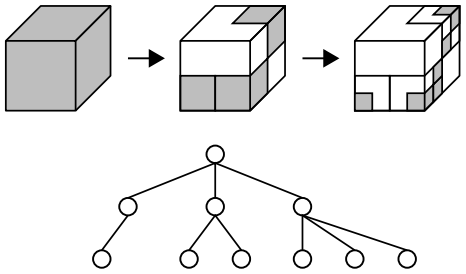


Figure 2: Left: Spatial subdivisions of an octree up to level 3. Occupied leaf nodes are shaded grey. Right: The corresponding tree structure of the sparse datastructure.

Consequently, most space is not occupied, and therefore most octree nodes will only have few children. The octree data structure is therefore ideally suited to store and retrieve 3D laser scanner data efficiently. A recursive refinement of an octree is illustrated in Figure 2.

2.1. Memory efficient encoding of an octree

Implementations of an octree often do not prioritize memory efficiency. A relevant property of an octree is that uniform subvolumes are represented as single nodes that do not subdivide further. For applications where the octree stores a pure voxelmap, this property compresses the data to a large degree. Of course, this is in comparison to a simple 3D grid stored linearly in memory. Thus arises the term sparse voxel octree in the computer graphics community, where an octree structure is used to efficiently access otherwise large amounts of data.

On the opposite side are the serialized pointer-free encodings. These have the highest potential for memory efficiency, since they do not need to store the actual octree structure. One such encoding is given by Girardeau-Montaut et al. (2005) and is implemented in the point cloud comparison software CloudCompare (Girardeau-Montaut, 2011). They employ the Morton order to store only the leaf level of an octree at 16 bytes per leaf. The Morton order or Z-order is an ordering of, in this case, 3-dimensional data (Morton, 1966). This approach has several drawbacks. Traversing the Octree is not possible in the classical sense. Instead binary search algorithms have to be applied when looking for a certain voxel.

Serialization is a very useful tool when storing the data for later use or when communicating over channels with limited bandwidth. Schnabel and

Klein (2006) combine a serialized octree with arithmetic coding to provide superior point cloud compression. While this approach excels at compressing point clouds, it becomes infeasible to efficiently process the data for most applications, i.e. finding a point in this encoding is in $O(n)$ as the tree has to be traversed from the root in breadth-first order until the point has been found. A more detailed discussion of octree related work is given in Section 5.

We opt for a pointer based octree since it allows for several operations and applications which are not feasible with the above designs. Modification, i.e. adding or deleting points from a serialized octree involves shifting the entire region before or after the position where the modification takes place. Wand et al. (2007) use a pointer based octree for interactive editing of large point cloud data. They also employ out-of-core storage to deal with data sets that do not fit into main memory. This technique, too, is out of the question for serialized octrees, since it can not as easily be predicted which data chunk needs to be fetched. Even though our efficient octree design is capable of the above applications, this paper focusses on the algorithmic questions of visualization and processing.

Many implementations store redundant information in each octree node. In computer graphics, for example, neighbor pointers as well as a parent pointer are used to facilitate extremely fast ray tracing at the cost of additional memory. Another encoding, that redundantly stores the position and size in each node is given in Figure 3, where `center` and `size` store the position and size of the node while `child` is an array of pointers to the 8 children. This allows to stop subdivision for empty nodes having to divide the volume into equal subvolumes, thus potentially reducing the number of nodes required for storage. On a standard 64-bit architecture, each node requires 100 bytes of memory. Although the implementation is somewhat naive, it nicely illustrates the possible memory gains that can be achieved.

We create an efficient octree implementation that is free of redundancies and is nevertheless capable of fast access operations. Our implementation allows for access operations in $O(\log n)$. Add and delete operations are also in $O(\log n)$. In the worst case long blocks of memory will have to be allocated or deallocated in a leaf node.

Most information about inner nodes of an octree is computed when recursing through the structure. The depth of a node is calculated as the depth of its parent plus one. Due to the properties of the regular octant subdivisions the size of a node is a function of its depth. Similarly the position of a node is computed by displacing the position of the parent node by half of

```
struct OcTree {
  float center[3];
  float size[3];
  OcTree *child[8];
  int nr_points;
  float **points;
};
```

Figure 3: Definition of an octree with redundant information and eight pointers to child nodes. The size of this node is 100 Bytes.

the cell size in the appropriate direction. The direction follows trivially from the size of the node and its position in the parent's list of children. Only the root node must therefore store some information about the octree as a whole, i.e., the position and size of the spanned volume. In the same manner parent pointers may be computed, or rather remembered, by pushing visited parents onto a stack. It is even possible to compute neighbor pointers by a fast indexing scheme. Section 4.2 explains this process. Unfortunately, as this operation requires backtracking along the parent stack it is necessarily less efficient than computing the other properties. For the sake of memory efficiency, we omit any information that is computable by traversing the tree. However, referring to the baseline implementation in Figure 3 the removed redundancy accounts for only 24 of 100 bytes. 64 remain for child pointers and 12 bytes for the point storage. We downsized this by moving the information about whether or not a node exists from the node itself into its parent. We add a single byte, where each bit corresponds to one octant of the node. This allows us to remove the constraint to always store 8 children, so that only those child nodes need to exist that contain valuable information in the first place. We can therefore remove 56 further bytes by storing only a single pointer to all children. Adding another byte, where each bit signals whether the corresponding octant is a leaf allows the removal of the point information that is unnecessary in inner nodes. Two alternative encodings that result from these considerations are presented in Figure 4.

Our encoding consists of three parts. The child pointer is the largest part of each node and is implemented as a relative pointer to the first child. All other valid children are arranged linearly in memory as shown in Figure 4. The pointer can vary in size for different systems. For 64 bit architectures we have chosen 6 bytes. As this is sufficient to address a total of 256 terabyte, there is no need for an additional bit signaling for a far pointer as proposed

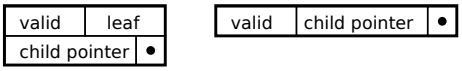


Figure 4: The two proposed encodings of an octree node optimized for memory efficiency. The child pointer as the relative pointer is the largest part of an octree node, but varies in size to accomodate different systems. In our implementation for 64 bit systems, it is 48 bit. `valid` and `leaf` are 8 bit large. Left: The proposed encoding with separate bit fields for valid and leaf. An entire node is thus contained in only 8 bytes of memory. Right: Alternative solution resulting in a constant depth octree.

by Laine and Karras (2010). Using a far pointer flag would require more sophisticated memory management, but it would enable one to reduce the size of the child pointer to two or fewer bytes. There is a second, easier way to reduce the number of bytes required for the child pointer, if we are willing to sacrifice $O(\log n)$ add and delete operations. In this case, the octree can be stored in a linear array in breadth first order, with each child pointer simply indexing the array. However, further discussion will focus on using this 6 byte pointer implementation.

`valid` and `leaf` are each a single byte large, one bit for each subvolume. `valid` bits signal whether the corresponding octant is present, while `leaf` bits signal whether the corresponding child is a leaf node. This encoding is somewhat redundant, as non-valid children cannot be leaf nodes. There are only $3^8 = 6561$ combinations possible. These could be compressed and represented with only 13 instead of 16 bits. Due to concerns about the runtime efficiency and the relatively minor reduction of the memory requirements, we decided against such a compression. It is possible to remove the leaf byte, by enforcing a constant depth of the octree (cf. Figure 4, right). Similar to the other properties of a node, that are determined recursively, the depth of a node is always well defined. While the size of an octree node is reduced by enforcing a constant depth this procedure is certain to increase the number of nodes. Let n be the number of bytes used for a node implementation without a leaf byte, the percentaged increase in the number of nodes is limited by $1/n$ to achieve increased memory efficiency. For our implementation, this amounts to approximately 15% increase that is allowed at most. We found that point clouds acquired by laser scanners are so sparse that they require a smaller percentage only for shallow trees. Increasing the resolution and thereby the depth also increases the number of nodes significantly, which in turn increases the size in memory exponentially as demonstrated in Table 1.

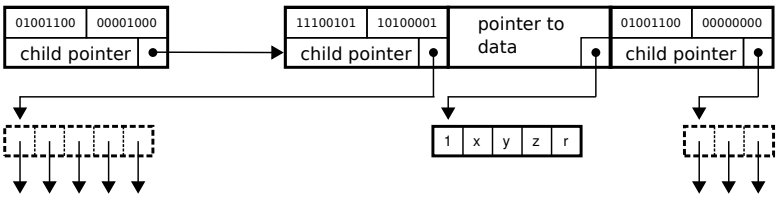


Figure 5: An example of a simple octree as it is stored using the proposed encoding. The node in the upper left has three valid children, one of which is a leaf. Therefore, the child pointer only points to 3 nodes stored consecutively in memory. The leaf node in this example is a simple pointer to an array which stores both the number of points and the points with all their attributes.

Even for the most compact dataset balance is obtained for a voxelsize of 10 cm. For resolutions above that, the difference in required memory space is so small as to be insignificant for most applications. Counterintuitively, then, the 8 byte encoding is the more robust and more memory efficient choice for data that is as sparse or more sparse than laser scanner point clouds.

Our implementation stores points in the leaf nodes, thus they need to be represented differently from inner nodes. In Figure 5, leaf nodes are pointers to arrays of points. The first entry is always the total number of points, then sequentially the information for each point, i.e., the coordinates and additional attributes such as reflectance. In that representation, leaf nodes would be n bytes larger than inner nodes, where n is the number of bytes used to encode the number of points. In our case it is more than sufficient to reserve $n = 4$ bytes for this purpose. Such a point list representation is then already more memory efficient than the usual `float**`, as it cuts down on a pointer.

2.2. Octree based compression of 3D point clouds

Our octree encoding drastically decreases the overhead for obtaining the data structure itself (cf. Table 1). As opposed to the reference implementation the memory for the point cloud exceeds now the overhead (cf. Fig. 6). Therefore, we seek to compress the point list as well. For a simple technical reason we like to store each point coordinate using only two bytes. Two bytes are exactly the resolution at which most laser scanners measure additional point attributes, such as reflectance, deviation. To store floating point coordinates in only two bytes without significant loss of precision, we use each bit of the two byte coordinate as $s/2^{16}$ increments to the lower left front corner

of the rectangular cuboid of the leaf node, where s is the side length of the cuboid. This is similar to color quantization as used for example by Gervauts and Purgathofer (1990).

Data of terrestrial laser scanners represented as four byte floating point value has a precision of approx. 100 μm (100 micrometer) at the maximal distance of 500 m. At a smaller distance, e.g., at 1.5 m, the precision increases to 1 μm. To achieve the same 1 μm precision the smallest volume in the octree must have a side length of 6.5 cm. Assuming a desired precision of 10 μm, which is still 2 orders of magnitude smaller than typical specified measurement precisions, the largest node is allowed to have a side length of 65 cm. At this voxel size the octree overhead is minimal even for large scans.

2.3. Efficient Construction of an Octree

When constructing octrees from unorganized pointclouds care must be taken not to exceed the available memory while still processing the high number of points as fast as possible. We give an algorithm that has negligible memory requirements and a runtime of $O(nlog(n))$ in the number of points. First, the bounding box of the point cloud is computed and the root node is initialized. The construction of the tree is then accomplished by sorting the pointlist similar to quicksort. At each node the given list of points is first reorded with respect to the center of the node in one dimension. The resulting two sublists are then both reordered with respect to the center in the next dimension. The same is done with the last dimension. This operation results in 8 possibly empty lists of points, one for each possible child. As this is done recursively in quicksort fashion for every node the initial list is sorted into the Morton order. The construction of the tree that takes place simultaneously is therefore in $O(nlog(n))$. The necessary memory never exceeds the point list plus the octree structure.

2.4. Experiments and results

To demonstrate the effectiveness of the proposed octree encodings, we computed the required memory for the octree data structure (without the points) with different depths. This was done for three representative scans of differing density as given in Figure 6. The data is given in Table 1 to 3 for different leaf sizes. It is important to note that the leaf size is only half of the side length of the leaf nodes, due to the way the octree volume is stored in our implementation. For all tests, the root volume and therefore all octree volumes axis aligned cubes. The size and position of the root is such

that it represents the smallest cube possible to contain the entire data set. The time needed to construct the reference as well as the proposed octree has also been determined experimentally. For each leaf size the average construction time for the octrees out of 10 trials performed on an Intel(R) Xeon(R) E5520@2.27GHz is shown in Table 1 to 3.

The first data set has the smallest number of points, and is the least dense, i.e., it contains many points with large distances to their neighbors. Consequently, the performance of the constant depth octree very quickly deteriorates (cf. Table 1. Already, at a depth of 5 the memory benefit of the smaller node representation is negated. In the larger and denser datasets (cf. Table 2 and Table 3), the octree reaches a depth of 10 before the constant depth requirement results in a larger memory footprint. Yet, even in the most dense dataset, several things speak against this constraint. At no resolution the memory benefit is particularly significant in the first place. The maximal memory saving in the bunker Valentin data set is only approx 20 kB. Compared to the size of the point cloud itself, which is about 700 MB this is neglegible. On the other hand, when the constant depth restriction results in a larger memory footprint, the penalty is several orders of magnitude larger than the benefit could ever be. The penalty also tends to increase polynomially with the tree depth. On level 15 the constant depth implementation requires almost 100 MB more memory. This is of course caused by the drastically increasing number of nodes required to pad the tree. A further consequence of this padding is the increased computations involved in iterating over the octree structure.

The time needed for the construction of either octree is near linear in the depth of the octree.. The reference octree is always faster by some fixed amount of time. This time difference is caused by the reallocation of the point data in the leaves, which happens in the proposed octree but not in the reference. In fact, if the construction of the referencealso reallocates the points in a more compact fashion, it is slower than the proposed variant.

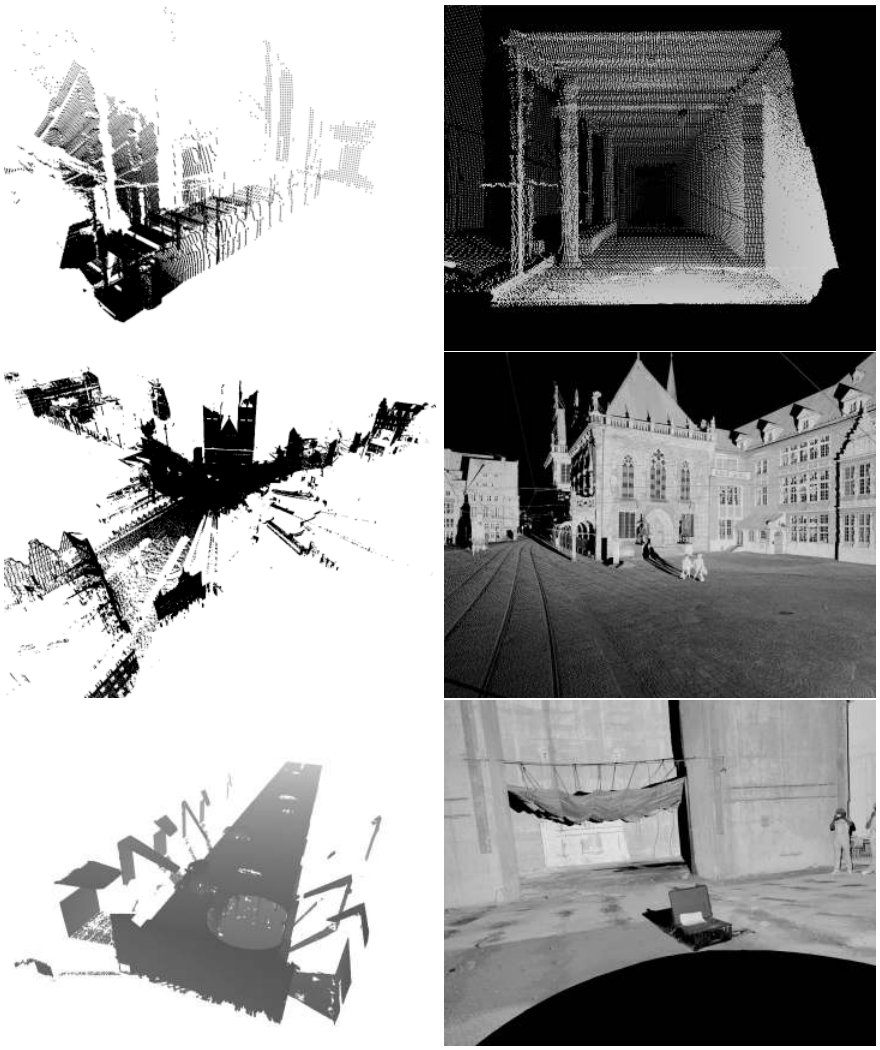


Figure 6: Three point clouds are used for the following analysis. The left point cloud is a 3D scan that was acquired by the mobile robot Kurt3D using an actuated SICK LMS200 laser scanner in an office environment with 81,360 points ($\approx$ 1.5 MB). Statistics for this data set is given in Table 1. The middle scan is a high resolution scan taken in the city center in Bremen using the Riegl VZ-400 3D scanner. The point cloud contains 15,896,875 points ($\approx$ 303 MB). Refer to Table 2 for data on this point cloud. The right scan was also acquired by the RIEGL VZ-400, but in a large scale indoor environment, the bunker Valentin in Bremen-Farge. The point cloud is therefore very dense with 22,538,374 points ($\approx$ 420 MB). Results are given in Table 2.

Table 1: Memory requirements for the sparse Kurt3D point cloud using several octree implementations and different resolutions. The first column gives the size of the smallest leaf in the tree, i.e., half of its side length. The second and third columns give the number of nodes and leaf nodes for the 8 byte per node implementation. The real size in memory as well as the average construction time follows in the next two columns. After those, the number of nodes for the 7 byte implementation is given as well as the percentual increase and the real size in memory. Construction time has been omitted, as it is virtually equally to the 8 byte implementation. The memory requirements and construction time for the implementation with 100 bytes per node are listed in the last two columns. (cf. Figure 3).

	8 byte				7 byte			100 byte	
Leaf size (cm)	# Nodes	# Leaves	Mem. Size	Constr. time (ms)	# Nodes	Incr. %	Mem. Size	Mem. Size	Constr. time (ms)
876	1	8	104 B	4.0	1	0	103 B	954 B	1.1
438	9	26	384 B	4.6	9	0	375 B	3.71 kB	2.0
219	34	77	1.19 kB	5.2	35	2.9	1.16 kB	11.87 kB	2.6
109.5	103	202	3.24 kB	6.0	112	8.7	3.2 kB	33.28 kB	3.2
54.76	282	505	8.31 kB	6.7	314	11.3	8.25 kB	86.81 kB	3.7
27.38	698	1352	21.8 kB	7.6	819	17.3	21.95 kB	230.12 kB	4.2
13.69	1777	3688	58.47 kB	8.6	2171	22.1	59.45 kB	621.05 kB	4.8
6.846	4121	8840	139.04 kB	10.2	5859	42.1	147.09 kB	1.55 MB	5.8
3.423	9327	19064	303.38 kB	11.9	14699	57.5	331.66 kB	3.57 MB	6.4
1.711	17219	34697	554.11 kB	13.4	33763	96	652.7 kB	7.25 MB	7.0
0.855	27668	52836	855.37 kB	14.3	68460	147	1.11 MB	12.85 MB	7.9
0.427	37351	68253	1.11 MB	14.3	121296	224	1.66 MB	20.09 MB	8.7

Table 2: Statistics for the Bremen city data set (cf. Table 1).

	8 byte				7 byte			100 byte	
Leaf size (cm)	# Nodes	# Leaves	Mem. Size	Constr. time (ms)	# Nodes	Incr. %	Mem. Size	Mem. Size	Constr. time (ms)
8560	6	12	192 B	1019.2	6	0	186 B	1.9 kB	557.1
4280	18	28	480 B	1321.2	18	0	462 B	4.8 kB	694.1
2140	46	86	1.4 kB	1535.7	46	0	1.3 kB	13.9 kB	939.2
1070	129	296	4.5 kB	1706.1	132	2.3	4.4 kB	45.3 kB	1165.7
535	408	800	12.8 kB	1909.7	428	4.9	12.5 kB	130.1 kB	1279.3
267	1166	2595	40.4 kB	2081.8	1228	5.3	39.7 kB	405.2 kB	1529.4
133	3616	7993	124.8 kB	2299.7	3823	5.7	122.6 kB	1.25 MB	1656.5
66.8	11130	24587	384.1 kB	2473.3	11816	6.1	377.7 kB	3.85 MB	1895.5
33.4	33965	75999	1.18 MB	2687.5	36403	7.1	1.16 MB	11.91 MB	2002.3
16.7	102728	233413	3.62 MB	2873.3	112402	9.4	3.58 MB	36.65 MB	2146.5
8.35	302573	687529	10.67 MB	3134.9	345815	14.2	10.67 MB	109.53 MB	2290.3
4.17	814040	1808993	28.22 MB	3432.0	1033344	26.9	28.94 MB	301.28 MB	2471.9
2.08	1927234	4166979	65.42 MB	3721.0	2842337	47.4	69.9 MB	742.98 MB	2576.0
1.04	4031140	7783889	125.65 MB	3901.8	7009316	73.8	142.47 MB	1.568 GB	2899.6
0.52	5592151	10142923	166.45 MB	4077.7	14793205	164.5	225.26 MB	2.643 GB	3199.9

Table 3: Statistics for the bunker Valentin data set (cf. Table 1).

	8 byte				7 byte			100 byte	
Leaf size (cm)	# Nodes	# Leaves	Mem. Size	Constr. time (ms)	# Nodes	Incr. %	Mem. Size	Mem. Size	Constr. time (ms)
5589	1	8	104 B	1330.0	1	0	103 B	954 B	565.1
2794	9	22	336 B	1631.9	9	0	327 B	3.28 kB	824.1
1397	30	59	948 B	1906.6	31	3.3	925 B	9.54 kB	1102.8
698	88	209	3.21 kB	2173.8	90	2.2	3.13 kB	31.69 kB	1367.7
349	294	755	11.41 kB	2403.9	299	1.7	11.15 kB	111.72 kB	1665.0
174	1038	2611	39.63 kB	2718.8	1054	1.5	38.71 kB	388.49 kB	1977.2
87.3	3602	8761	133.94 kB	2964.6	3665	1.7	130.78 kB	1.31 MB	2136.9
43.6	12106	30330	460.8 kB	3146.0	12426	2.6	450.94 kB	4.53 MB	2434.0
21.8	41098	101743	1.54 MB	3409.5	42756	4.0	1.52 MB	15.31 MB	2688.1
10.9	134366	324058	4.96 MB	3739.3	144499	7.5	4.9 MB	49.667 MB	2767.0
5.45	412174	969512	14.93 MB	4079.1	468557	13.6	14.91 MB	152.43 MB	3123.8
2.72	1158782	2679693	41.42 MB	4544.5	1438069	24.1	42.22 MB	436.48 MB	3227.9
1.36	2977454	6617215	103.22 MB	5057.0	4117762	38.2	108.23 MB	1.137 GB	3503.0
0.68	6356651	13130561	208.41 MB	5303.6	10734977	68.8	232.71 MB	2.529 GB	3895.8

Table 4: Compression results for the data sets. The compression ratio between the original binary data and the octree representation as well as the average error for representing the point of the data sets and their octree representation are given.

Data set	File size `.txt` MB	File size binary 64 (32 Bit) MB	File size compressed octree MB	Ratio %	Error μm
Kurt3D	1.907	1.862 (0.931)	0.472	50.73	4.165
Bremen city	477.0	424.4 (242.5)	121.6	50.14	5.102
Bunker Valentin	665.7	601.8 (343.9)	172.1	50.04	6.677

3. An open file format for exchanging 3D point clouds

Due to the great diversity of terrestrial 3D modeling applications several software products exists, which are suitable for special tasks. For airborne and kinematic laser scans there exists a common lidar data exchange format, the `.las` format (Samberg, 2007). "This binary file format is an alternative to proprietary systems or a generic ASCII file interchange system used by many companies. A problem with proprietary systems is that data cannot be easily taken from one system or process flow to another. In addition, processing performance is degraded because the reading and interpretation of ASCII elevation data can be very slow and the file size can be extremely large, even for small amounts of data."(American Society for Photogrammetry and Remote Sensing, 2011). This applies also for terrestrial 3D modeling pipelines, but there is currently no standard format available. Kern et al. (2009) defines a binary point cloud format that hence reduces the processing time.

The octree as given in Section 2 is well suited for storing large point cloud data, as it is a lossless compression, which reduces the size of a point cloud by a factor of roughly two and comes with a fast indexing. Table 4 presents the compression results for the three data sets. The serialization of the proposed octree results in an efficient method for storing point clouds.

4. Efficient algorithms on octrees

4.1. Adaptive visualization using frustum culling

To exploit the octree structure for fast visualization we implemented frustum culling. The octree then works as a hierarchy of bounding volumes. Much like in the software `qsplat` (Rusinkiewicz and Levoy, 2000) it is used

to quickly decide which points are visible. In addition, it also enables one to dynamically vary the level of detail to enforce a high framerate by rendering only a faction of points.

In computer graphics the frustum is the visible region of space. For a perspective projection, this region is a rectangular pyramid, constituted of 6 planes. Frustum culling is the process of distinguishing which parts of the scene are within and which parts are outside of the frustum, thus finding the visible parts of the scene. Non-visible elements do not need to be drawn, hence increasing the performance. Algorithm 1 describes how to efficiently implement the frustum culling using the octree data structure. The algorithm is initally called at the root of the octree, with the full viewing frustum.

Algorithm 1 display(set<plane> frustum, octree node)

```
set<plane> childfrustum;
for all p ∈ frustum do
  if PlaneAABBCollision(p, this) = within then
    childfrustum.insert(p)
  else if PlaneAABBCollision(p, this) = outside then
    return
  end if
end for
if isLeaf(node) then
  drawPoints(node)
else
  for all child ∈ node.children do
    display(childfrustum, child)
  end for
end if
```

Algorithm 1 employs the function PlaneAABBCollision(), which determines the position of an octree volume with respect to a plane, i.e., whether the plane is within the volume or outside. The latter case has two subcases. Each plane of the frustum is oriented such that it is trivial to determine on which side a given point is. Consequently, if an octree volume is outside with respect to a plane, the node and all its children are not visible. If, on the other hand, the volume is entirely on the inside, culling with the respective plane is discontinued for all children.

To summarize, the given algorithm first determines which frustum planes are relevant for the children. Simultaneously, it checks wether the current

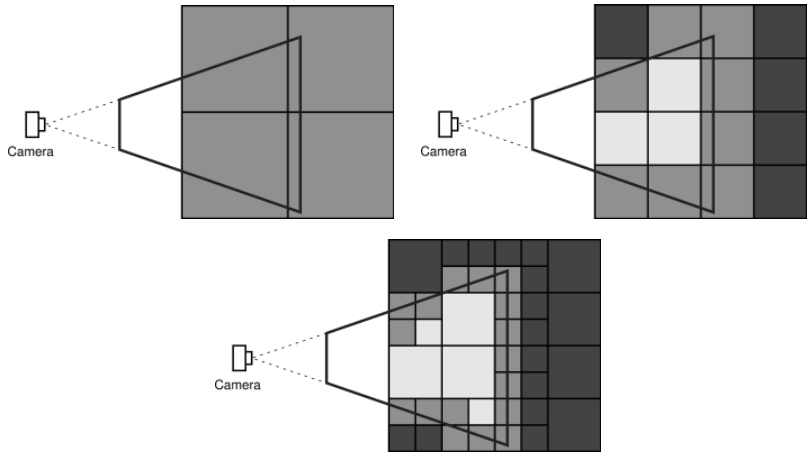


Figure 7: The principle of frustum culling using octrees. Gray nodes are known to be partially within the frustum, so that culling must continue. Light gray Nodes are known to be entirely within the frustum, so that culling is discontinued. Dark gray nodes are entirely on the outside of the frustum.

octree volume is visible and terminates accordingly. The second step is to either display the points in the current leaf, or to recursively continue culling in the child nodes. In this fashion, one foregoes unneccessary collision checks both inside and outside of the viewing frustum. Furthermore, checks are also minimized for the intersection case, as only relevant frustum planes will be used. Figure 7 illustrates this principle.

The main bottleneck in a software culling implementation is transmitting the point information to the graphics card when a large number of them is within the viewing frustum. Using the octree structure this problem can be mitigated in several ways. First, when an octree volume would appear as a single pixel it is sufficient to also send only a single vertex instead of all contained points without decreasing image quality. Second, we may dynamically adjust rendering quality to allow for navigation in the scene, similar to Richter and Döllner (2010) and Dachsbacher et al. (2003). As before, we send only single vertices in any octree volume that falls below a level-of-detail threshold (number of pixels on screen). The size of the rendered vertex is adjusted accordingly. With thresholds greater than 1, this trades resolution for speed. An example of levels of details are given in Figure 8.

We have conducted experiments to find the optimal voxel resolution. Fig-

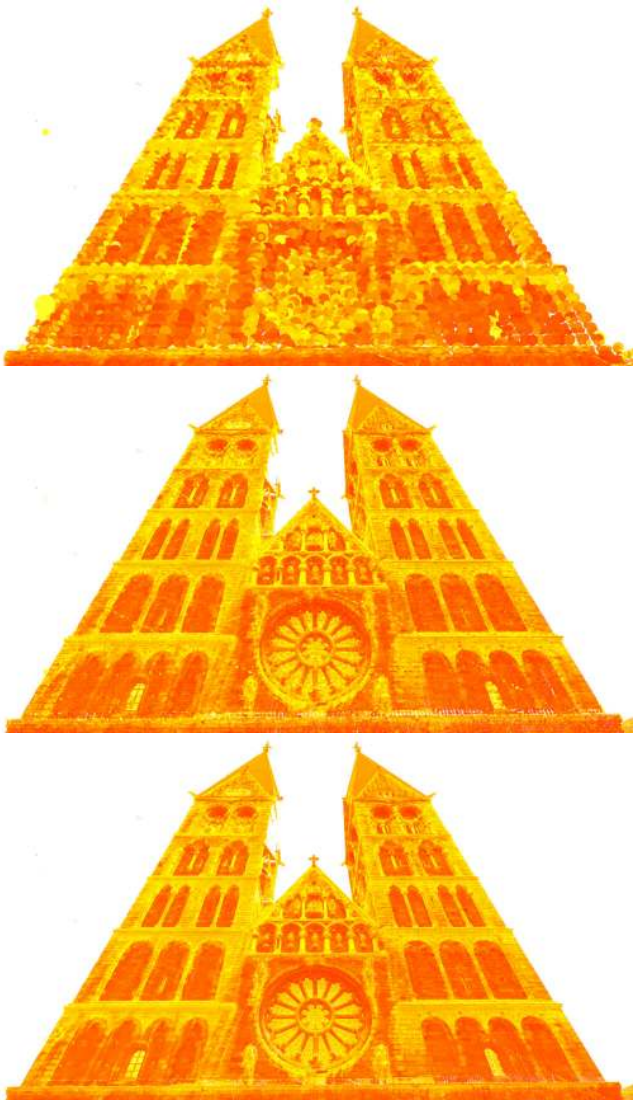


Figure 8: Three levels of detail for the octree visualisation The point cloud is taken from the Bremen city data set.

ure 9 presents framerates for a point display employing the frustum culling implemented on the CPU. Several different voxel resolutions as well as the standard implementation with no software culling are plotted. To achieve comparable situations the point display has rendered a 3D-flight-through, where the camera moves on a path through the Bremen city data set. The camera started at a position where the entire point cloud was visible, then proceeded to move through the point cloud close to the point of origin until it reached a position within the volume where about 1% of the point cloud is visible. One observes a low framerate in the beginning, and an ever increasing speedup. The time for the rendering of each frame was measured by a clock with millisecond accuracy, so that the maximal framerate is 1000 frames per second.

The speedup gained by the frustum culling is small when large portions of the point cloud occupy the frustum. Reducing the number of points increases the framerate considerably more when using the octree culling as compared to the default point display. Note, that for scans that occupy an area this large it is more typical to only view a small fraction of the scene.

Comparing performances for various voxel sizes, we see that while large voxel sizes still result in some speedup, performance becomes more and more unsteady when a large number of points is to be displayed. Framerates for large voxelsizes experience sudden jumps when voxels enter or leave the viewing frustum. This becomes apparent for a voxel size of 267 cm, but only somewhat noticable with the next smaller size of 133 cm. Generally, voxel sizes between 10 cm and 100 cm are optimal for these types of scans.

4.2. Ray casting

Ray casting is the method of finding an intersection of a ray with a surface. It is an important problem in computer graphics, where it is used to render images, typically of iso-surfaces (Roth, 1982; Knoll et al., 2006), and in robotics, where it is commonly used to sample potentional measurements (Thrun et al., 2000; Wurm et al., 2010). For rendering images, each pixel is followed from the viewpoint to the first object that is encountered. This is different from ray-tracing, where rays are reflected from objects so that multiple rays per image pixel are cast into the geometry. Ray-casting is thus a simplified ray-tracing technique.

Before efficient ray casting is possible in an octree without neighbor pointers, we detail how to perform fast indexed node access and neighbor

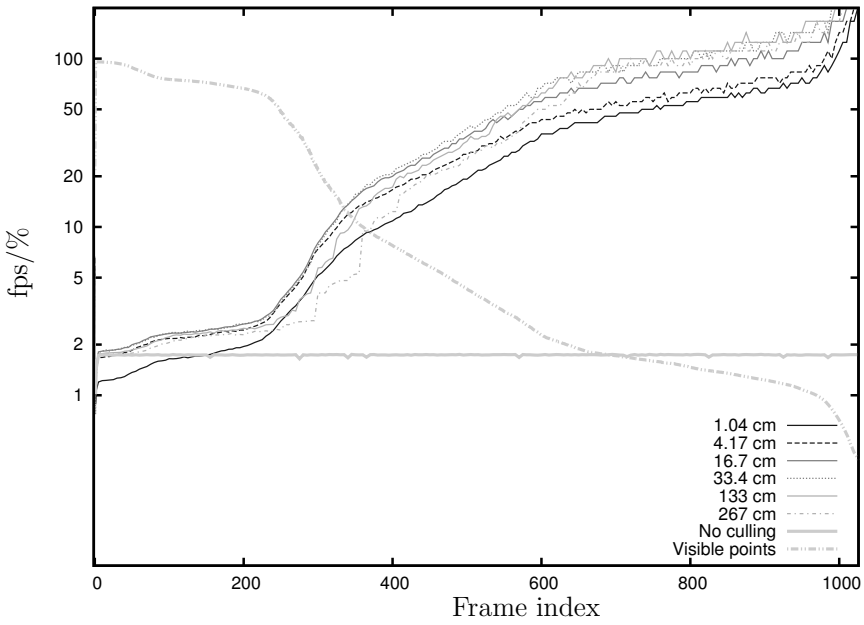


Figure 9: Framerates of the octree culling with different voxel sizes over the frame index in the rendered video. The percentage of visible points in each frame is also plotted. Please note that the y-axis is on a logarithmic scale to be able to differentiate in the areas of both low and high framerates.

traversal. Indexed node access is a lookup-query with (x, y, z) integer coordinates valid on the deepest level of the octree, i.e., an octree with depth d has integer coordinates 0 to $2^d - 1$ in each dimension. Naive lookup implementations perform *collision checks* with the octree planes. As this is an expensive task, we use integer coordinates for an efficient traversal of the octree with only a few bit operations. Similarly to Knoll et al. (2009) we assume the existence of a pre-computed array childBitDepth with childBitDepth$[d] = 1 \ll (\text{maxDepth} - d - 1)$. An efficient lookup is performed as in Algorithm 2. The key to this algorithm lies in the second line. Here the integer coordinates are mapped to the index of the child that contains the given coordinates. The algorithm also shows how parent pointers are simulated by a simple trace that is extended during the traversal of the tree. The function to find a neighbor node is merely an extended lookup. To

find a neighbor of a given node, the node in the parent trace is selected that is the deepest that still contains the desired index. This can be efficiently computed by comparing the current node index with the desired index (cf. Algorithm 3). Then a lookup starting at that parent is started to locate the corresponding neighbor.

Algorithm 2 lookup(Vector3i index, octree node, octree *parentTrace, int depth)

```
loop
  int childBit = childBitDepth[depth]
  int childIndex = (index.x & childBit ≠ 0) ≪ 2 | (index.y & childBit ≠
  0) ≪ 1 | (index.z & childBit ≠ 0)
  octree *parentTrace[depth] = &node
  depth++
  node = node.children[childIndex]
  if isLeaf(node) then
    return
  end if
end loop
```

Algorithm 3 findNeighbor(Vector3i dindex, Vector3i cindex, octree *parentTrace)

```
int depth = mostSignificantBit( (cindex.xˆdindex.x) | (cindex.yˆdindex.y)
| (cindex.zˆdindex.z) );
lookup(dindex, parentTrace[depth], parentTrace, depth);
```

We eliminated the need for neighbor pointers for this algorithm. This comes at the cost of increased computational complexity. In the worst case we must backtrace completely up to the root and completely back down again. While these situations do not occur very often the traversal to a neighbor is still in $O(\log n)$. With the help of this function it is now feasible to implement ray casting, which is given in Algorithm 4. Here, Bresenham implements the 3-dimensional version of the well-known Bresenham line-algorithm, such that it consecutively returns 3-dimensional integer coordinates, that are traversed by the given ray.

This naive ray casting can still be improved by collecting rays into packets

Algorithm 4 castRay(Ray ray, octree root)

```
octree *parentTrace[maxDepth]
int depth = 0
octree node = lookup(ray.origin, root, parentTrace, depth)
Vector3i ci = Bresenham(ray);
Vector3i cci = ci;
loop
  while contains(node, ci) do
    cci = ci;
    ci = Bresenham(ray)
  end while
  node = findNeighbor(ci, cci, parentTrace)
  if isLeaf(node) then
    drawPoints(node)
    return
  end if
end loop
```

as presented recently by Knoll et al. (2009). Using a technique called coherent octree traversal, access operations to the data structure is further minimized.

4.3. Nearest neighbor search for scan matching

Nearest neighbor search (NNS) is a part of many scan matching algorithms for establishing corresponding points. The most prominent example of this is the Iterative Closest Point (ICP) algorithm, which spends most of its processing time in the lookup of closest points (Besl and McKay, 1992). It is therefore of utmost importance for this application to reduce the computing time of this task. Naively implemented, finding a closest point requires iterating over the entire dataset, i.e., it is in $O(n)$, where n is the number of points in the point cloud. This expensive running time is avoided by employing metric data structures. The most popular data structure to speed up NNS in scan matching is the k-d tree (Friedman et al., 1977), where $k = 3$. As k-d trees are binary trees they allow an efficient implementation of NNS. Principally, octrees should allow for the same efficiency. In fact, due to the octree's regular subdivision it ought to be better suited for NNS than the popular k-d tree (Arya et al., 1998). The complication arises during the implementation of NNS in an octree. The key to an efficient traversal to the

node containing the nearest neighbor for both tree variants is the order in which children are visited. The number of nodes that we need to visit is best reduced by the closest child first criteria (best bin first), i.e., the order of traversal is determined by the distance to the query point. This is trivial to do for the binary k-d tree, but requires some effort for an octree which may have one to eight child nodes.

For any octree node with 8 children there is a total of 48 possible sequences in which to traverse the children. Every child corresponds to an octant of the entire coordinate space. The query point may fall into any of those 8 octants. For each of those cases there are 6 possible traversals determined by the order of proximity of the query point to the three split planes. Therefore, NNS in an octree has to make proximity checks to three split planes, sort them and select the appropriate sequence of traversal for a closest child first search for every traversed node. Compared to this, the traversal order for a k-d tree order is instantly determined by a single proximity check, thereby avoiding uneccessary computations if nodes need not to be visited.

The regular subdivisions of an octree are still leveraged for an NNS that is in most cases faster or as fast as in a k-d tree. The largest benefit is that fast indexing as in section 4.2 is possible in an octree. This allows us to directly traverse to the deepest octree node, which contains the bounding sphere of the query point, with a constant number of floating point operations. The full NNS with closest child first and backtracking is then performed on this node. The initial node lookup is considerably faster than the equivalent operation, essentially a point lookup that is already in the tree, is in a k-d tree. However, the speedup gained by this is clearly dependant on the maximal allowed distance to the query point. The smaller the maximal distance, the deeper the initial node lookup can traverse on average. The deeper said node is, the fewer computations are performed in the following NNS. To evaluate the performance of the NNS for varying maximal distances, we performed ICP scan matching employing the octree and a k-d tree NNS. Results are given in Table 5 and Figure 10. Despite the previous argumentation, the octree based NNS does *not* suffer considerably more from larger maximal distances than the kd tree based NNS. We observe however, that there is an increase in the variance for the time required for the NNS using the octree. Conversely, the variance of the k-d tree based NNS is stable over all distances. This suggests that the octree is more vulnerable to the combination of large maximal distances and unfavorable starting pose estimates.

For ease of implementation and to further reduce the number of floating

Table 5: Average computing time for the ICP algorithm employing nearest neighbor search using k-d trees and octrees. The average was computed over 100 and 20 runs of ICP for the Kurt3D data set and the larger Riegl data sets, respectively. Default parameters were used for every test, with a maximal point-to-point distance of 25 cm and 50 ICP iterations. Random noise was added to the initial pose estimate for each run. The translational error was linearly distributed between -25 and 25 cm whereas the rotational error was linearly distributed between -10° and 10°. Construction time for the trees is not included, all times are in milliseconds.

Data set	k-d tree	octree
Kurt3D	3043.099	2386.881
Bremen city	355848.476	314506.905
Bunker Valentin	837675.381	784241.905
Kurt3D reduced	757.514	625.683
Bremen city reduced	91735.667	74706.85
Bunker Valentin reduced	91153.0	74821.238

point operations, we restrict the number of traversals to 8 instead of 48. Since the order of traversal only depends on the octant into which the query point falls, there is no need for proximity checks or sorting. Consequently, no floating point operations are required in our NNS implementation except in the leaves of the octree, where the list of stored points are checked against the query point. The approach is summarized in Algorithm 5. We use the function FindClosestInLeaf(), which is a straightforward check of the points stored in the leaf. The initial lookup for finding the deepest octree node that contains the bounding box around the query point is a modified indexed lookup. For this purpose the two diametrically opposed corners of the bounding box are converted into integer coordinates. The tree is then traversed as in Algorithm 2 until both indices disagree on the child that is to be traversed next.

Algorithm 5 FindClosest

Input: query point q, maximal allowed distance d
  lookup deepest node N containing bounding box of q
  convert q to octree coordinate i
  return FindClosestInNode(N, q, i, d)

Algorithm 6 FindClosestInNode

Input: query point q and its coordinate i
Input: maximal allowed distance d and the current node N
1: compute child index c from i
2: **for** $j = 0$ to 8 **do**
3: get next closest child $C = N$.children[`sequence[`c`][`j`]`];
4: **if** C is outside of bounding ball **then**
5: **return** currently closest point p
6: **else**
7: **if** C is a leaf **then**
8: FindClosestInLeaf(C, q, d)
9: update currently closest point p
10: **else**
11: FindClosestInNode(C, q, i, d)
12: update currently closest point p
13: **end if**
14: **end if**
15: **end for**
16: **return** currently closest point p

4.4. RANSAC for efficient parameter estimation

The Random Sample Consensus (RANSAC) algorithm is an approach for estimating parameters of a model that best describes a set of sample points (Fischler and Bolles, 1981). While it is traditionally used for line and plane detection RANSAC can also be used for any other parameterized model. It is an iterative algorithm that repeatadly draws a small number of samples from the data to be modelled. From this subset of samples the parameters of the model are computed and the number of points in the entire data set that intersect with the model is calculated. As the process is repeated the model with the largest number of points is selected as the result.

The most expensive step of the algorithm is determining the number of points that agree with the candidate model. In an unorganized point cloud this requires iterating over all points. We employ the octree data structure and obtain a significant speedup. Similar to Section 4.2 where a frustum is checked against an octree, the candidate model is recursively intersected with the octree nodes to determine which nodes may contain points on the model. The process is esemplarily depicted for the 2-dimensional case in Figure 11.

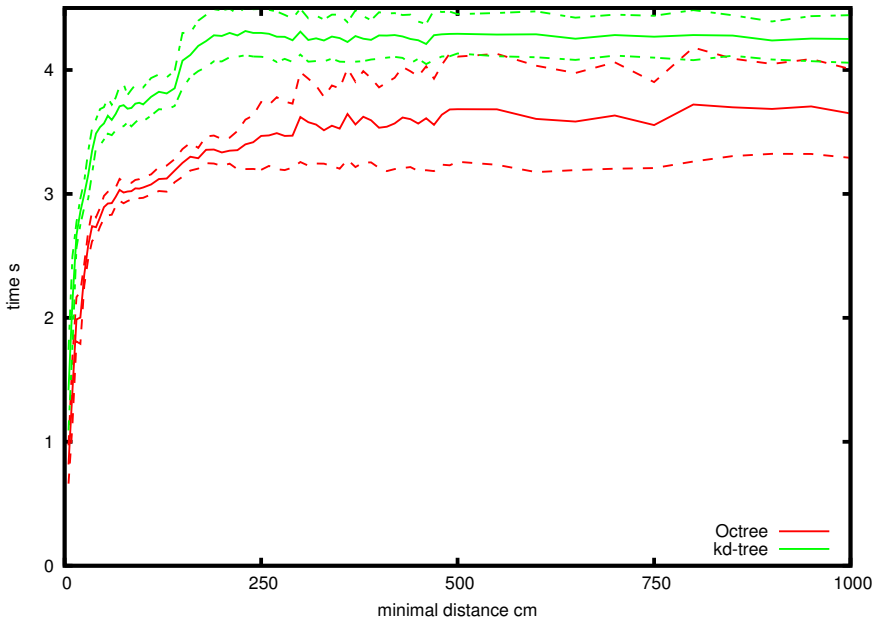


Figure 10: Plot of the average and standard deviation of the computing time of ICP using k-d trees and octrees with respect to the maximal allowed matching distance. For each data point 100 runs of ICP, with 50 iterations each, were done with noise applied to the initial starting pose estimate as in table 5.

After a candidate line has been generated, intersection tests are performed on the children of nodes known to be at least partially on the line. This process is executed from top to bottom. Finally, the points in the leaves are counted. In this way a large number of points is automatically excluded from being checked against the model, thus leading to a speedup. A comparison of the computing time of a standard RANSAC for plane detection with octree-enabled version is given in Table 6. Even including the construction time of the octree we achieve significant speed-up. Examples of detected planes for the 3 data sets are given in Figure 12.

In addition octrees are able to speed up RANSAC by a smart sampling scheme. Schnabel et al. (2007) have shown that by carefully selecting samples that are in proximity to each other, the number of iterations required to detect a shape with a certain probability is reduced by several orders of

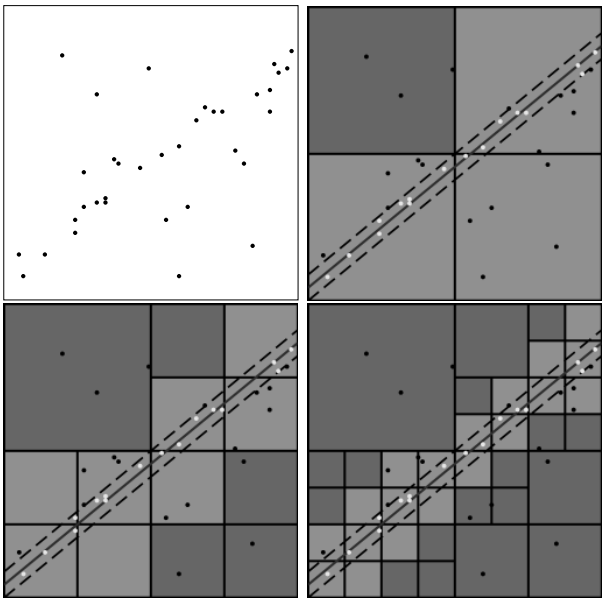


Figure 11: Speed-up of RANSAC using a tree data structure. Top left: The initial sample set, where a line should be detected. Top right: A line has been generated from a sample set and is intersected with the octree. The dashed lines signify the maximal distance threshold of RANSAC. Dark gray Nodes are outside of the model, light gray nodes intersect the line. Bottom: The intersection test continues only in those children of the nodes that are known to intersect the model.

magnitude. This is done by first selecting a sample in some random leaf l, and then selecting further samples only from children of a randomly selected parent node of l.

5. Related work

Since its introduction in the early 80's by Meagher (1982), the octree data structure has experienced widespread use among various fields that deal with large quantities of 3-dimensional data, especially computer graphics (Knoll et al., 2006, 2009; Gobbetti et al., 2008; Laine and Karras, 2010) but also theorethical physics (Bielak et al., 2005) and, of course, robotics (Wurm et al., 2010).

In computer graphics and visualization the octree has recently had a

Table 6: Average computing time of RANSAC for plane detection with 5000 iterations with as well as without the octree. Results were averaged over 100 runs. For each data set 5000 iterations of the RANSAC were done. The octree times include the construction of the octree (≈8 s for Bremen city and ≈10 s for the Bunker data set). All times are in milliseconds.

Data set	no octree	octree	speed-up
Kurt3D	1666.57	176.69	9.43
Bremen city	388551.55	11084.81	35.05
Bunker Valentin	539536.13	26399.15	20.43

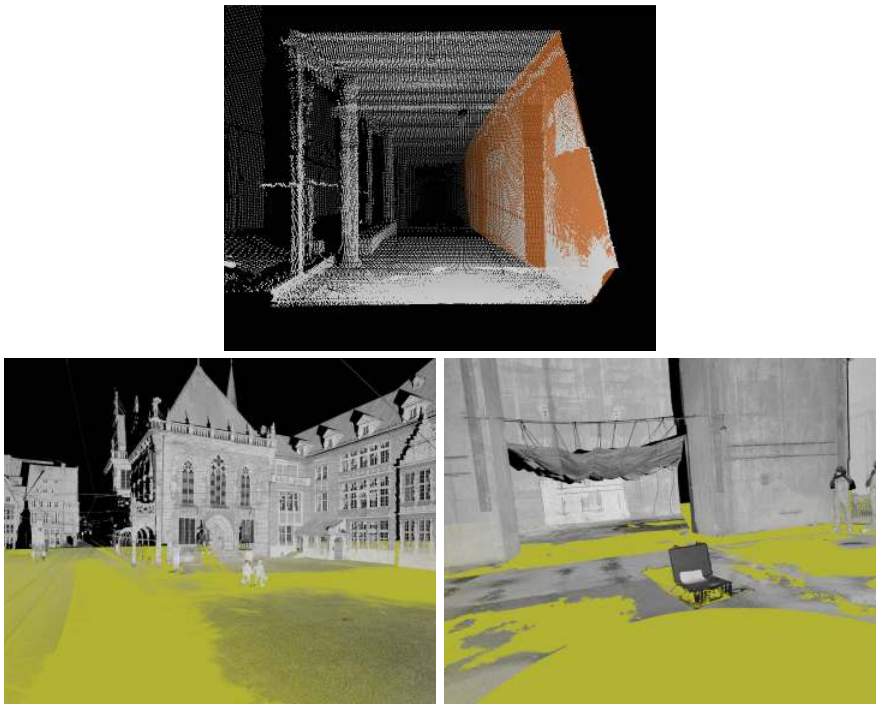


Figure 12: Extracted planes using RANSAC for the three data sets presented in Figure 6.

resurgence under the term *sparse voxel octree*. Here, the octree is used for efficient ray tracing and casting. An octree simultaneously represents large datasets with a small memory footprint and provides ray casting in $O(\log n)$,

which yields superior performance, since realtime visualization using ray tracing is usually in $O(n)$, where n is the number of objects in the scene. In this application octrees are representations of a voxelmap only, i.e., leaf nodes do not store other data, apart from the inherent properties needed for the visualization of a voxel such as color and normals. Laine and Karras (2010) have presented an octree encoding for the GPU which is similar to ours. Their implementation stores countours in each node to improve visualization quality in addition to the usual color and normal information. Knoll et al. (2006, 2009) have developed powerful ray tracing algorithms on octrees. They employ the fast indexing scheme as presented by Frisken and Perry (2002) and improve upon standard ray traversal with slice-based coherent octree traversal.

QSplat is a non-octree based visualization program for point cloud data (Rusinkiewicz and Levoy, 2000). Points, their normals and colors are stored in a hierarchy of bounding spheres. Responsive display of massive amounts of data can be maintained by dynamically reducing the complexity of the object or scene. This is achieved by visualizing the hierarchy only to a certain depth. The bounding sphere data structure, although compact, is not well suited to applications other than visualization. Construction thereof is non-trivial and ambiguous. Due to its regularity, an octree is more compact and suited for other applications, too.

Several libraries exist which employ the octree data structure for storing and processing point clouds, like PCL (Radu Bogdan Rusu et al., 2011; Rusu and Cousins, 2011), octomap (K. M. Wurm et al., 2011), CloudCompare (Girardeau-Montaut, 2011) and `xgrt` (M. Wand and A. Berner and M. Bokeloh and A. Fleck and M. Hoffmann and P. Jenke and B. Maier and D. Staneker and R. Parys, 2011). We give the node sizes for each library and compare them to our approach in Table 7.

Furthermore, octrees are used in the color quantization algorithm as designed by Gervauts and Purgathofer (1990) to minimize memory requirements. Each level in their octree represents one bit of the colors contained therein, beginning with the most significant bit in the first level. This is similar to our compression of points, where the more significant bits are implicitly stored in the octree data structure.

Table 7: Memory requirements for a single node of different libraries in bytes on 64bit systems. An x notifies that the library is templated, and a number of bytes must be added to the total byte count. The number depends on the size of the type stored in the tree and can be expected to be several bytes.

Library	Node size	Remarks
`xgrt`	$144 + x$	
octomap	$72 + x$	
PCL	$64 + x$	
PCL low memory base	$25 + x$	(since ver 1.1.1. Sept. 2011)
CloudCompare	16	size of leaf node
3DTK	8	

6. Conclusions and outlook

This paper presents a novel implementation of a basic data structure for 3D point clouds. The data structure, an octree, exceeds the purpose of storing data. The pipeline for obtaining 3-dimensional models is sped-up. To this end, we have presented an efficient exchange file format, fast point cloud visualization, effective 3D scan matching, and a clever plane detection algorithm.

In future work we will continue using our octree for efficient 3D point cloud processing, e.g., for globally consistent scan registration (Borrmann et al., 2008), for automatically deriving semantic information, for dynamic maps, i.e., maps that can handle changes of the scene, and for next-best-view planning.

Acknowledgments

We would like to thank Nikolaus Studnicka (RIEGL Laser Measurement Systems) for his idea to propose the octree as exchange file format. Furthermore, we would like to thank Prashant K.C. for acquiring many terrestrial laser scans on our campus.

References

M. Wand and A. Berner and M. Bokeloh and A. Fleck and M. Hoffmann and P. Jenke and B. Maier and D. Staneker and R. Parys, November 2011. xgrt. `http://www.gris.uni-tuebingen.de/xgrt` (last accessed August 14, 2012).

American Society for Photogrammetry and Remote Sensing, February 2011. Common lidar data exchange format. `http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html` (last accessed August 14, 2012).

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., Wu, A. Y., 1998. An Optimal Algorithms for Approximate Nearest Neighbor Searching in Fixed Dimensions. Journal of the ACM (JACM) 45 (6), 891–923.

Besl, P., McKay, N., 1992. A method for Registration of 3-D Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 14 (2), 239–256.

Bielak, J., Ghattas, O., Kim, E. J., 2005. Parallel Octree-based Finite Element Method for Large-Scale Earthquake Ground Motion Simulation. Computer Modeling in Engineering and Sciences 10 (2), 99 – 112.

Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A., Hertzberg, J., February 2008. Globally Consistent 3D Mapping with Scan Matching. Journal Robotics and Autonomous Systems (JRAS) 56 (2), 130–142.

Dachsbacher, C., Vogelgsang, C., Stamminger, M., 2003. Sequential point trees. In: ACM SIGGRAPH 2003 Papers. SIGGRAPH '03. ACM, New York, NY, USA, pp. 657–662.

Fischler, M. A., Bolles, R. C., 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM 24 (6), 381–395.

Friedman, J. H., Bentley, J. L., Finkel, R. A., September 1977. An algorithm for finding best matches in logarithmic expected time. ACM Transaction on Mathematical Software 3 (3), 209–226.

Frisken, S. F., Perry, R. N., 2002. Simple and Efficient Traversal Methods for Quadtrees and Octrees. Journal of Graphics Tools 7 (3), 1–11.

Gervauts, M., Purgathofer, W., 1990. A simple method for color quantization: Octree quantization. In: Glassner, A. S. (Ed.), Graphics Gems. Academic Press Professional, Inc., San Diego, CA, USA, pp. 287–293.

Girardeau-Montaut, D., November 2011. Cloudcompare. `http://www.danielgm.net/cc` (last accessed August 14, 2012).

Girardeau-Montaut, D., Roux, M., Marc, R., Thibault, G., 2005. Change detection on points cloud data acquired with a ground laser scanner. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 3/W19), 30–35.

Gobbetti, E., Marton, F., Iglesias, G., Jose, A., July 2008. A Single-pass GPU Ray Casting Framework for Interactive Out-of-Core Rendering of Massive Volumetric Datasets. The Visual Computer: International Journal of Computer Graphics 24 (7), 797–806.

K. M. Wurm et al., May 2011. Octomap. `http://octomap.sourceforge.net/` (last accessed August 14, 2012).

Kern, F., Pospis, M., Prümm, O., February 2009. Das Datenaustauschformat Binary Pointcloud (BPC) für TLS-Punktwolken. In: Luhmann/Müller (Ed.), Photogrammetrie Laserscanning Optische 3D-Messtechnik, Beiträge der Oldenburger 3D-Tage. Wichmann, Oldenburg, Germany, pp. 20–30.

Knoll, A., Wald, I., Parker, S., Hansen, C., September 2006. Interactive isosurface ray tracing of large octree volumes. In: Proceedings of the IEEE Symposium on Interactive Ray Tracing. pp. 115–124.

Knoll, A. M., Wald, I., Hansen, C. D., February 2009. Coherent multiresolution isosurface ray tracing. The Visual Computer: International Journal of Computer Graphics 25 (3), 209–225.

Laine, S., Karras, T., 2010. Efficient Sparse Voxel Octrees. In: Proceedings of the ACM SIGGRAPH symposium on Interactive 3D Graphics and Games (I3D '10). ACM, New York, NY, USA, pp. 55–63.

Meagher, D., 1982. Geometric Modeling using Octree Encoding. Computer Graphics and Image Processing 19 (2), 129 – 147.

Morton, 1966. A Computer Oriented Geodetic Data Base and a new Technique in File Sequencing. Tech. Rep. Ottawa, Ontario, Canada, IBM Ltd.

Radu Bogdan Rusu et al., May 2011. Point cloud library. `http://pointclouds.org/` (last accessed August 14, 2012).

Richter, R., Döllner, J., 2010. Out-of-Core Real-time Visualization of Massive 3D Point Clouds. In: Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa. AFRIGRAPH '10. ACM, New York, NY, USA, pp. 121–128.

Roth, S. D., February 1982. Ray casting for modeling solids. Computer Graphics and Image Processing 18 (2), 109–144.

Rusinkiewicz, S., Levoy, M., 2000. QSplat: A Multiresolution Point Rendering System for Large Meshes. In: Proceedings of the ACM SIGGRAPH. ACM Press/Addison-Wesley Publishing Co., pp. 343–352.

Rusu, R. B., Cousins, S., May 2011. 3D is here: Point Cloud Library (PCL). In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11), ICRA Communications. Shanghai, China, pp. 1–4.

Samberg, A., 2007. An Implementation of the ASPRS LAS Standard. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 3/W52), 363–372.

Schnabel, R., Klein, R., July 2006. Octree-based Point-Cloud Compression. In: Symposium on Point-Based Graphics 2006. Eurographics, pp. 111–120.

Schnabel, R., Wahl, R., Klein, R., 2007. Efficient RANSAC for Point-Cloud Shape Detection. Computer Graphics Forum 26 (2), 214–226.

Thrun, S., Fox, D., Burgard, W., April 2000. A Real-time Algorithm for Mobile Robot Mapping with Application to Multi Robot and 3D Mapping. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00). Vol. 1. San Francisco, CA, USA, pp. 321–328.

Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., Schilling, A., September 2007. Interactive editing of large point clouds. In: Botsch, M., Pajarola, R. (Eds.), Proceedings

Symposium on Point-Based Graphics (PBG '07). The Eurographics Association, Prague, Czech Republik, pp. 37–46.

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W., 2010. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In: Proceedings of the IEEE ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation. Anchorage, AK, USA.

2.3.2 Comparison of Nearest-Neighbor-Search Strategies and Implementations for Efficient Shape Registration

The next paper is an in-depth look at how well state-of-the-art implementations of spatial data structures perform in the context of the 3D point cloud registration problem. It was published in the Journal of Software Engineering for Robotics (`http://www.joser.org`), (3(1):2–12) in 2012. The nearest-neighbour search strategy as outlined in the previous section is also analyzed in the paper. Of course, octrees are not the only data structure that can be used with point cloud data. There are other spatial data structures like the k-d tree, the R-tree, space filling curves and more.

The most popular one is certainly the k-d tree. Like the octree, the kd-tree is a hierarchical subdivision of space. The root node of a k-d tree spans the entire space. The root node as well as every other inner node partitions its allotted space into two smaller spaces by a dividing hyperplane which is orthogonal to one of the coordinate axes. Thus, every inner node has two child nodes. Unlike the octree each subdivision does not necessarily create equal subspaces, although the two subspaces together encompass the space of the parent space. These rules typically result in deeper trees than an equivalent octree. However, since unequal subdivisions are allowed, the k-d tree is able to adapt more to the data than the octree. Finally, traversing the k-d tree only involves a simple check of a single coordinate value.

An R-tree is a hierarchical data structure designed for spatial data with higher dimensions like line segments or polygons. It is thus mainly used for geographic information systems. All nodes represent an axis aligned bounding box of arbitrary dimension. A node may have arbitrarily many child nodes. A child's bounding box must be entirely contained within the parents bounding box. The bounding boxes of child nodes may overlap and may not fill up the entirety of the bounding box of the parent node. Due to the relatively unrestricted nature of these rules there are many ways of constructing an R-tree. Another consequence is that the construction process is complex and time-consuming.

Mathematically a space filling curve is a continuous function with a domain between 0 and 1 whose image is the entirety of some arbitrary higher dimensional space. In this context it is a function that completely fills a 3-dimensional Euclidean space. Space filling curves thus define a mapping of 3D space onto a 1D value. This can be realized for example by the Morton order [52]. The Morton order is essentially what is obtained when performing a depth-first traversal of an octree and connecting all visited leaf nodes. As the depth of the octree approaches infinity the resulting curve fills up the entire euclidean space. This order can be used to efficiently store and access point clouds [14]. In practice the depth of the octree needs not to be infinity as space is easily discretized and point clouds are finite.

Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration

Jan Elseberg[1,*] Stéphane Magnenat[2] Roland Siegwart[2] Andreas Nüchter[1]

[1] Automation Group, Jacobs University Bremen gGmbH, 28759 Bremen, Germany

[2] Autonomous Systems Lab, ETH Zürich, Switzerland *jan.elseberg@jacobs-university.de

I. Introduction

Shape registration is the problem of finding the rigid transformation that, given two shapes, transforms one onto the other. The given shapes do not need to be wholly identical. As long as a partial overlap is possible, shape registration seeks the transformation that accomplishes just that overlap. For the extent of this paper we consider a shape to be a set of points in three-dimensional Cartesian space, i.e. a point cloud. The registration problem is most commonly solved using the iterative closest point (ICP) algorithm [1]. In addition to the two shapes, it assumes an estimate of the transformation to be computed. Such an estimate is usually available in contexts where shape matching is of importance. ICP is an iterative algorithm that alternates between determining, based on the current transformation estimate, the nearest neighbors (NN) of every point of one shape, and updating the estimate based on the NN. Updating the estimate is a relatively simple mathematical operation that is linear in the number of neighboring points. Naively implemented, the nearest-neighbor search (NNS) is in $O(nm)$ where n and m are the number of points in the respective point clouds. Exploring every possible pairing of point can be avoided by employing spatial data structures. The average runtime of NNS is then usually in the order of $O(n \log m)$ but is still by far the most computationally expensive part of shape registration. This paper gives a comprehensive empirical analysis of NNS algorithms in the domain of 3-dimensional shape matching. For this purpose, section II gives an overview of spatial data structures and the state of the art on comparison of search algorithms. Section III presents the novel search

Regular paper – Manuscript received April 19, 2009; revised July 11, 2009.
This work was partially supported by the AUTOSCAN project (BMWi KF2470003DF0).

TABLE I
PROPERTIES FOR ALL TESTED NNS LIBRARIES.

Library	revision	Data structure	k-NN search	fixed radius	ranged search	optimized for
3DTK	rev. 470	octree	×	×	✓	shape registration & efficient storage
ANN [2]	Ver. 1.1.1	k-d tree	✓	✓	×	
CGAL [3]	Ver. 3.5.1-1	k-d tree	×	✓	×	
FLANN [4]	bcf3a56e5fed2d4dc3a340725fa341fa36ef79a4	k-d tree	✓	✓	×	high dimensions
libnabo [5]	Ver. 1.0.0	k-d tree	✓	×	✓	
SpatialIndex [6]	Ver. 1.4.0-1.1	R-tree	✓	×	×	
STANN [7]	Ver. 0.71 beta	SFC	✓	×	×	multithreading

algorithm on octrees and the novel efficient implementation of NNS in k-d trees. We explain our experimental setup in section IV, provide results in section V and conclude the paper in section VI.

II. RELATED WORK

Spatial data structures partition space to allow efficient access to the stored elements via positional queries. Most spatial data structures are hierarchical in nature, such as k-d trees [8] and octrees [9]. The grid file [10] is a rare flat non-hierarchical data structure. We explicitly exclude it from consideration here because of its prohibitive memory requirements. A special type of NNS employs the Morton order for arranging the point cloud. A Morton order is a space-filling curve (SFC), i.e. a curve that allows ordering the set of points along one dimension while preserving the locality thereof [11].

In addition to the well known k-d tree and octree we also consider the hierarchical data structure R-tree [12]. There are other data structures such as the range-tree [13] and vp-tree [14]. Unfortunately, to the authors' knowledge, there is no publicly available NNS library that employs either the vp- or the range-tree.

An octree [9] is the generalization of quadtrees, which store two dimensional data [15]. Each node in an octree represents the volume formed by a rectangular cuboid, often simplified to an axis-aligned cube. Consequently an octree node has up to eight children, each corresponding to one octant of the overlying cube/node. A node having no children usually implies that the corresponding volume can be uniformly represented, i.e., no further subdivision is necessary to disambiguate. This convention is not completely applicable when storing points, which are dimensionless, i.e., there is no volume associated with them. When storing a point cloud, a stopping rule must be defined for occupied volumes, like a maximal depth or a minimal number of points. Empty volumes will however not be further split up.

The k-d tree [8] is similar to the octree in that each node represents an axis-aligned rectangular cuboid and its children split up the volume to form smaller cuboids. Empty volumes are not subdivided further, and there must be a stopping rule for occupied

volumes. However, a k-d tree is a binary tree and the subdivision of the node's volume must not be regular, i.e. each node also defines an axis-aligned hyperplane that splits the volume in two. Unlike the octree, there is not a unique way of constructing a k-d tree since the splitting plane can be placed at any position in any node. There are several strategies for placing the splitting plane. The standard rule is splitting before the median point along the longest dimension of the node's volume. Another splitting rule is the so called midpoint rule, which merely splits the current volume in half along its longest dimension.

The R-tree [12] is non-binary and is primarily used for spatial data other than point clouds, i.e. for geographic information systems. Each node represents an axis-aligned bounding box of arbitrary dimension. The represented volumes are allowed to overlap, except for the requirement that a child's volume must be entirely within the parent's. There are several variants of the R-tree that differ only in their insertion algorithms. The linear, quadratic and exponential R-tree all insert a new element into the node that requires the least extension, but they use algorithms of different complexities (hence the name) for splitting the node, if necessary. The R*-tree insertion algorithm chooses which node to insert the new element into by a minimum overlap criteria. It also extends the splitting algorithm by the principle of forced reinsertion, i.e. elements that are already stored may be deleted and reinserted into another node [16].

Apart from the data structure, the NNS queries are of interest. There are different types of NNS query that deserve discussion: The first type of query that springs to mind in the context of NNS is the k-NN search. The result of this type of query are the k NN around a specified query point. Another is the fixed radius search, which computes all points withing a given radius of the query point. The combination of the two types yields the ranged search, i.e. the retrieval of the k NN with a given maximal distance. The latter query with $k = 1$ is the type of query needed for shape registration. In the large majority of realistic applications the presence of obstruction leads to only partially-overlapping point clouds. Thus, allowing point correspondences with a too large distance can only infuse the registration process with errors. The range search variant also allows for a large potential for efficient implementation. The maximal radius restricts the search region to only a small subspace of the entire data. This can be efficiently combined with the restriction on the number of points. Even the small subspace does not need to be explored in its entirety if it can be ensured that the k NN points have been located. It is surprising then, that many NNS libraries fail to provide this search variant (cf. Table I).

To the authors' knowledge there is little previous work exploring and comparing current NNS libraries in the context of shape registration. Blumenthal et al. [17] give a first evaluation of some available NNS libraries. We compare a wider range of algorithms on a wider range of data, both artificial and real. Pomerleau at al. [18] explore several parameters influencing registration performance, but only one parameter (an early-out approximation factor) is related to the NNS. There have been comparisons of different

data structures for NNS, although not specifically for shape registration. Dandamudi et al. [19] compare the binary decision (bd) tree to variants of the k-d tree. Nakamura et al. [20] propose a new data structure, the md-tree and compare it against the k-d tree and the bd-tree. They also find the k-d tree to perform worse than the other data structures in the tested scenarios. Judging from the large number of libraries employing the k-d tree it is clear that it is still the favored data structure in general NNS libraries. Greenspan et al. [21], [22] have proposed improved NNS algorithms for shape registration. While these are promising, they have not yet found their way into any of the NN libraries evaluated in this paper.

III. Design and Implementation

In this section we present our highly-optimized implementation of the octree and k-d tree. The octree implementation has specifically been designed for the efficient storage of large point clouds. It is implemented in the 3DTK [23] and supports NNS, point-cloud compression and fast visualization. The fast k-d tree implementation is called libnabo [5] and is not to be confused with the simple k-d tree implementation within 3DTK.

A. Octree

Our octree code implements a novel search algorithm as presented in this section. It is written in ISO/IEC C++ 2003 and is integrated into the 3DTK framework as a visualization, shape detection and NNS data structure. It does not rely on other libraries to store its nodes and geometry data as it is optimized for memory efficiency [24]. It is templated to allow for choosing the precision of the point data.

Principally, octrees should allow for extremely efficient implementations of NNS. Due to their regular partitioning of the search space and the high branching factor, coordinate queries are very fast [25]. Yet, most NN libraries are based on k-d trees.

The complication is due to the fact that during NNS, nodes near the query point must be visited. Thus an order of traversal must be found that is both efficient, in that it visits no more nodes than necessary, and easy to compute. The key to an efficient traversal is the order in which children are visited. The number of nodes that need to be visited is best reduced by the closest-child first criteria, i.e. the order of traversal is determined by the distance to the query point. This is trivial to do for the binary k-d tree, but more involved for an octree.

For any node with 8 children there are a total of 96 possible sequences in which to traverse the children. A query point may fall into or be closest to any one of the 8 octants. This is the first child to traverse. For each of those cases there are 12 possibilities in which to traverse further. This is determined by the proximity of the query point to the 3 split planes. The next two children to visit are, in order, the two closest of the three direct neighbors of the first child. The fourth child to visit is either the last remaining direct neighbor or the node that complements the cuboid of visited nodes. The sequence

Algorithm 1 FindClosest

Input: query point q, maximal allowed distance d
  lookup deepest node N containing bounding box of q
  convert q to octree coordinate i
  return FindClosestInNode(N, q, i, d)

Algorithm 2 FindClosestInNode

Input: query point q and its coordinate i
Input: maximal allowed distance d and the current node N
compute child index c from i
for $j = 0$ to 7 **do**
  get next closest child $C = N$.children[`sequence[c][j]`]
  if C is outside of bounding ball **then**
    return currently closest point p
  else
    if C is a leaf **then**
      FindClosestInLeaf(C, q, d)
      update currently closest point p
    else
      FindClosestInNode(C, q, i, d)
      update currently closest point p
    end if
  end if
end for
return currently closest point p

in which to visit the remaining four nodes is entirely dependent on the traversal before and can not change. Thus, there are $8 \cdot 3 \cdot 2 \cdot 2 = 96$ possible traversals in whole.

NNS in an octree has to make proximity calculations to 3 split planes, sort the distances and select the appropriate sequence of traversal for every traversed node. Compared to this, the order of traversal in a k-d tree is instantly determined by a single proximity check, thereby avoiding unnecessary computations if nodes need not be visited.

However, the regular subdivisions of an octree can still be leveraged for fast NNS. The biggest benefit is that fast indexing is possible in an octree. This allows us to directly traverse to the deepest octree node, which contains the bounding sphere of the query point. This is done with a constant number of floating-point operations and is considerably faster than the equivalent operation in a k-d tree, which is essentially a lookup of a point already in the tree. However, the speedup gained is clearly dependent on the maximal distance allowed to the query point. The smaller this distance is, the deeper the node enclosing the bounding sphere is on average. The closest child first NNS with backtracking is then performed on this node, i.e. the deeper the node, the fewer the number of steps needed in the NNS.

The number of floating-point operations needed to choose the correct order of traversal out of the 96 possibilities at each node is exceedingly high. By restricting ourselves to 8 possible traversals it is possible to eliminate the need for any floating point operations in the inner nodes. The order of traversal is decided by which octant the query point is closest to. This is done in accordance to Frisken and Perry [25] and requires no floating-point operations. Each of the 8 traversals are identical in nature, first the closest octant is visited, then the three direct neighbors, then the three direct neighbors of those and finally the most distant node. The approach is summarized in Algorithm 1. The function `FindClosestInLeaf` is the NNS inside a leaf, a floating-point proximity check of all stored points.

B. k-d tree

Our fast k-d tree implementation is called libnabo [5]. Its tree creation and search algorithm is similar to ANN [26], a well-established library [2], but it differs in the implementation details, which significantly improves performance (see Section V). libnabo is written in ISO/IEC C++ 2003 and provides a nearest-neighbor–search interface, allowing to easily add new search strategies through inheritance. The current version of libnabo provides a brute-force strategy for comparison purpose and the kd-tree strategy. Geometry data are stored in Eigen 3 [27] structures, as this library is quickly becoming a standard for linear algebra in robotics research[1]. Other data such as the neighbors heap and the tree nodes are stored in STL containers, such as `std::vector`.

For creating the tree, libnabo uses the *sliding-midpoint rule*, that is, when considering a cuboid, it tries to cut it by the middle on its dimension of maximal extent. If this results in a trivial split, that is, if there is one side with no points, libnabo moves the splitting plane such that there is at least one point on each side. The search algorithm is also similar to ANN, doing a recursive descent on the side of the cutting plane the closest to the search point; and exploring alternative branches while going back up until the k NN are closer than the cutting planes. Algorithm 3 gives an overview of this method. Initially, the minimum distances D to other points for every dimension and their norm r are 0. The heap H of size k holds the current candidates for the k NN and allows fast check and insertion. libnabo provides two implementations, a linear- and a tree-based heap. The linear heap is suitable for small k, as it has a complexity of $O(k)$ with a small constant while the tree heap is in $O(\log k)$ but with a larger constant. We have experimentally found that up to $k = 30$, the linear version is faster. Note that D and H are passed as mutable references. The other parameters are either passed as constant references or as mutable values.

The difference with ANN lies at the level of data structures. Where ANN employs a tree of objects based on pointers, libnabo uses a compact vector of nodes. To understand why this difference is significant, let us consider floating-point data on a 64-bit

[1]it is being used by ROS and MRPT for instance

Algorithm 3 LibnaboFindClosest

Input: query point q, node N, heap H, minimum distance r,
Input: minimum offsets D, maximal squared allowed distance l
  if N is a leaf node **then**
    search for closest points to q in bucket pointed by N
    update H
  else
    get cut dimension c and cut value v from N
    $o_{\text{old}} = D[c]$, $o_{\text{new}} = q[c] - v$
    if $o_{\text{new}} > 0$ **then**
      LibnaboFindClosest(q, rightChild(N), H, r, D, l)
      $r = r - o_{\text{old}}^2 + o_{\text{new}}^2$
      if $r \leq l$ **and** $r <$ head(H) **then**
        $D[c] = o_{\text{new}}$
        LibnaboFindClosest(q, leftChild(N), H, r, D, l)
        $D[c] = o_{\text{old}}$
      end if
    else
      LibnaboFindClosest(q, leftChild(N), H, r, D, l)
      $r = r - o_{\text{old}}^2 + o_{\text{new}}^2$
      if $r \leq l$ **and** $r <$ head(H) **then**
        $D[c] = o_{\text{new}}$
        LibnaboFindClosest(q, rightChild(N), H, r, D, l)
        $D[c] = o_{\text{old}}$
      end if
    end if
  end if

architecture and compute the memory footprint. For every split node, ANN holds the cutting dimension (4 bytes), the cutting value (4 bytes), the lower and the upper bounds (8 bytes), and pointers to left and right children (16 bytes). Moreover, the node being a virtual object, it holds at least a pointer to its vtable (8 bytes). Therefore, a single node consumes at least 40 bytes of memory. On the contrary, libnabo's node is a non-virtual class[2] that contains only the cutting dimension and index of right child (first 4-bytes word), and a union of the cutting value or the bucket index (second 4-bytes word). Indeed, if the cutting dimension (the least-significant bits of the first word) is smaller than the number of dimensions, the node is a split node and the second word contains the cutting value. In this case, the left child is the index of this node plus one, and the right child is encoded in the most-significant bits of the first word. On the contrary, if the cutting dimension is equal to the number of dimensions, this node is a leaf node and the second word contains the index of the bucket. This index points to a dense

[2]Note that the *node* class is a private inner class in a subclass of the nearest-neighbor–search interface. This interface is implemented as a superclass with virtual members and a static factory function. Thus libnabo is extensible through inheritance.

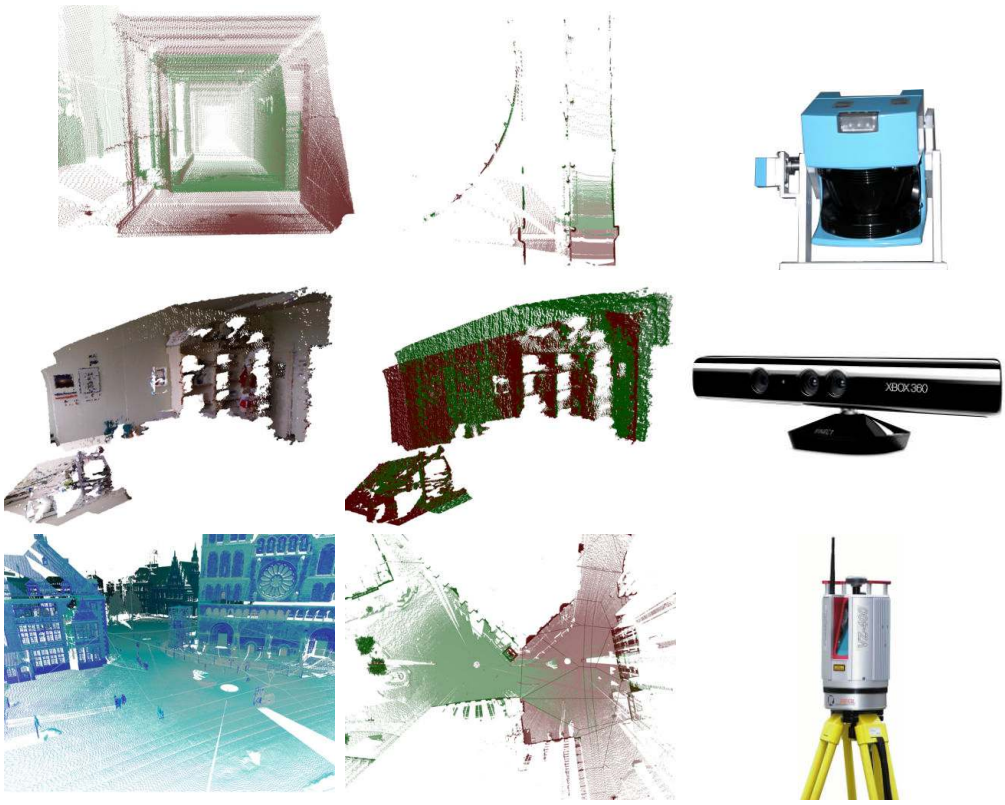


Fig. 1. Three pairs of real point clouds are used in our experiments. The top data set is a pair of 3D scans that was acquired by an actuated SICK LMS200 laser scanner in an office environment with ≈ 80,000 points each. The data set in the middle has been acquired by the Microsoft Kinect in an office environment. The clouds contain ≈ 290,000 points each and are relatively dense. Data courtesy of Jochen Sprickerhof of University of Osnabrück. The data set on the bottom is a high resolution scan taken in the historic city center of Bremen using the Riegl VZ-400 3D scanner. The point clouds contain ≈ 16,000,000 points each.

array of point indices, allowing the search algorithm to access the points. Therefore, a node in libnabo is only 8 bytes, a 5-fold spare in memory compared to ANN.

IV. Experimental methodology

Many factors influence the performance of a NNS query. One factor is the maximal distance parameter, which restricts the search space around the query point. The larger the maximal distance the more of the space the search routine must explore. A second factor is the type of data on which the NN query is being performed. Some data may result in optimally constructed data structures, while others may lead to unfavorable configurations. For this reason we carry out experiments on a multitude of data, both

artificial and real. The artificial point clouds are two geometrical primitives each in a hollow and filled variants. It should be noted, that for shape registration the hollow variants are more reminiscent of real data. Shape matching is almost exclusively done on surface data as most devices that capture 3D data like laser scanners and stereo camera acquire only such data. For each data set we have randomly generated 60,000 points in or on the geometrical primitive. We chose this number because it enables us to do a large number of tests while still maintaining experiments that are representative. We choose two shapes, a sphere with a radius of 1 and a cube of side length 2.

In addition to the tests on the artificial data sets, we run experiments under real conditions, i.e. data acquired by 3D range finders with parameters that produce the optimal registration. For this purpose we selected 3 pairs of point clouds that are representative of different types of sensors (see Fig. 1).

For the k-d trees a factor that influences average running time is the splitting strategy. Of the libraries used in this paper, only ANN supports changing the splitting rule. We use the rule suggested by the author's the sliding midpoint rule [28]. This rule is also used by libnabo. 3DTK's k-d tree as well as FLANN split the volume of a node along the largest side of the bounding box of the contained points. CGAL employs the standard median splitting rule along an axis dependent on the level of the tree.

Another notable parameter is the minimal number of points per leaf in the data structure, i.e. the bucket size. The smaller the bucket size is the larger is the overhead of the additional levels in the tree. The larger the bucket size the more points will have to be searched in a linear fashion. An evaluation on the SICK LMS200 data set (cf. Fig. 2), reveals that a bucket size between 5 and 20 is usually optimal. However, these results are not entirely representative and depend on the data sets used. In our experiments on artificial data, where trends are observed rather than absolute comparisons to be made, we use the default bucket size of the respective libraries, with two exceptions. In both FLANN and SpatialIndex we set the bucket size to be 10. This is also the default for most other libraries. In FLANN there is no default bucket size, so a choice had to be made. For SpatialIndex the default is 100 because it is intended that the data structure is cached to disk. However we run SpatialIndex in main memory and a bucket size of 100 gives significantly worse performance. For the experiments under real conditions the absolute running time instead of the trends in the runtime is the most important criteria. To provide clear results we set the bucket sizes to the optimal size as derived from Fig. 2, i.e. 5 for libnabo and 15 for both 3DTK data structures and FLANN. ANN does not allow for the changing of the bucket size.

Approximate search algorithms for the k-d tree, i.e. NNS with no guarantee of finding the exact NN, exist [29] but we opt to only test the exact NNS, since only this is implemented by all libraries. We use two variants of the R-tree for our experiments, the quadratic R-tree and the R*-tree. The exponential R-tree is not implemented in SpatialIndex, and the quadratic variant outperforms the linear one in all our experiments. Three libraries, ANN, FLANN and libnabo support more than a single NN query. Since

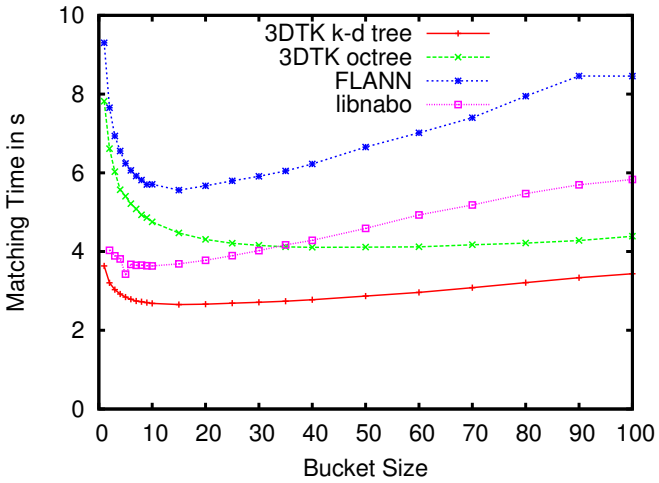


Fig. 2. ICP runtime in seconds on the SICK LMS200 data set.

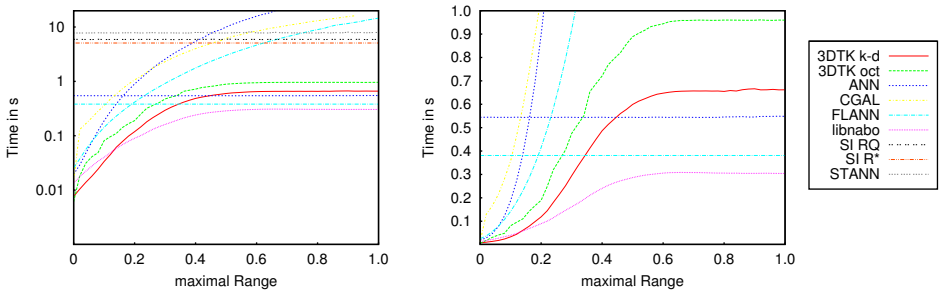


Fig. 3. The runtimes from all algorithms for the hollow cube/sphere combination. The query point set is the cube, whereas the sphere is the model data set. Left: Runtimes are plotted on a logarithmic scale to give an overview. Right: Runtimes are plotted on a linear scale in a smaller subsection.

ANN and FLANN do not support the ranged search, both queries are tested for both libraries. For libnabo, only the ranged search is tested as the k-NN search is implemented as a special case of the ranged search.

We compare a large set of NNS libraries: 3DTK, ANN, CGAL, FLANN, libnabo, SpatialIndex and STANN. Most of these rely on the k-d tree, in fact of these 3DTK is the only library implementing the octree, SpatialIndex the only one to rely on R-trees and STANN is the only one using SFCs. Only very few libraries feature multithreading capabilities, therefore all experiments were performed in single-threaded mode. The

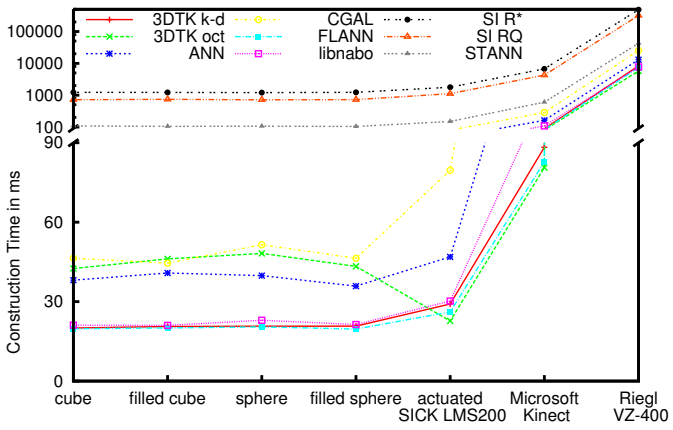


Fig. 4. The average times needed for creating different data structures. Note, that due to the large differences in the times the y-axis has been split into a linear and logarithmic scale.

results of every library on a handful of chosen testcases has been compared to each other to confirm their correctness.

All libraries were compiled with the gnu compiler collection version 4.4.3 on optimization level 3. The system running all experiments is a 64 Bit Xubuntu Linux (kernel 2.6.32-27) with an Intel(R) Xeon(R) E5520@2.27GHz and 12 GB of memory.

V. Results

The most extensive experiments were done on the aforementioned artificial data sets. For each library we varied the maximal range allowed for the NN from 0 to 1 in steps of 0.02. We repeated the NNS in 100 trials, and plotted the averaged running time.

An interesting combinations of query and model data set is presented in Fig. 3. It is a combination that resembles shape matching most closely, as both the query data set (the cube) and the model data set (the sphere) are surfaces. Clearly, the runtime of the ranged and fixed radius search depend strongly on the maximal distance. The larger the allowed distance, the more query points (near the cube's corners) have a NN. The more NN that need to be found, and the farther away these are, the longer the search will need to finish. Larger distances affect the fixed radius search algorithms significantly more than the ranged searches. The runtime for the fixed radius search continues to rise polynomially, whereas the ranged searches level off near $\sqrt{(3)} \approx 0.73$, which is the maximum possible distance in the data set. The runtime of the k-NN search is of course not influenced by the maximal distance. Most of them are significantly slower than the ranged searches. Due to the polynomial behavior of the runtime of the fixed radius

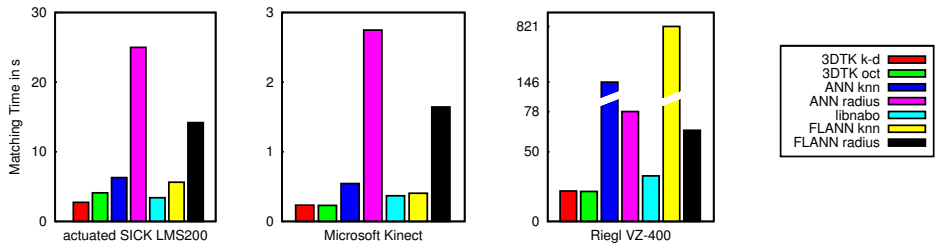


Fig. 5. The results from the 5 top performing libraries for the real data sets. Data structure construction time is not included in the figures.

algorithms all of the k-NN are faster for large enough maximal distances. However, only the ANN and FLANN library perform similar or better than the ranged searches for large enough maximal distances. Similar behavior is exhibited in other combinations. For an overview see Fig. 6. Note, that due to the large differences in running time, the time is on a logarithmic scale. For plots on a linear scale see Fig. 7. There is a large discrepancy in the running time for the search libraries. This discrepancy falls quite neatly along the lines of which type of algorithm is implemented. In almost all cases the best performing NN libraries were those implementing the ranged search, i.e. 3DTK and libnabo. The libraries only implementing k-NN search, i.e. STANN, CGAL and SpatialIndex can not compete with other algorithms. The remaining libraries, ANN and FLANN are in a few cases faster than some ranged search implementations. For a clearer look into this region, please see Fig. 7. In all non-trivial cases, i.e. when the query data set and the model data set are distinct, libnabo is faster than both FLANN and ANN. There are 2 exceptions to this, where libnabo is equal in runtime to FLANN and ANN. This occurs when the maximal range is at significant fractions of the point clouds size, i.e. $\approx \frac{1}{3}$ to $\frac{1}{2}$. This is also the range at which FLANN and ANN start to beat the other ranged search algorithms. Before this the ranged search implementations are faster than both the k-NN search implementations and the fixed radius search implementations.

For shape matching the region of smaller maximal distances are crucial. During the matching process the point clouds move closer towards each other. The average point to point distance is inherently reduced. The effect this has on the runtime of the NNS can be tracked on the rightmost column in Fig. 6 and Fig. 7. On the top, the query points (hollow cube) are farthest away from the model data set. The points are closer with the filled cube, even closer with the hollow sphere and finally identical with the filled sphere. The ranged searches as well as the k-NN searches speed up due to this progression, whereas the fixed radius search actually slow down.

However, absolute comparisons of the runtime may not be valid based on these results alone when the runtime difference is only several hundreds of milliseconds. Differences may be caused by non-optimal bucket sizes (cf. IV) or by the artificial nature of the data

sets. In addition to the artificial data sets, we therefore also compare the runtime of the 5 top performing libraries on real data sets. To ensure bucket sizes play no role in these results, we have chosen the optimal values as per Fig. 2. The results are presented in Fig. 5. The previous trend of the ranged search libraries performing better than FLANN and ANN continues. For the smaller data sets the k-NN search performs with similar speeds as the ranged search. For the large data set acquired by professional hardware, the performance is significantly worse. The fixed radius search can compete with the ranged search in the latter case but is significantly worse in the smaller data sets. In contrast to the experiments with artificial data, libnabo tends to perform slightly worse than the 3DTK algorithms on real data sets. This is likely due to the added overhead of converting the query data set into data usable by libnabo, i.e. the 3DTK NNS has a home field advantage.

All previous experiments were concerned with the runtime of the NNS and excluded the time needed to construct the data sets. Fig. 4 shows the average time for construction for all data structures and every point cloud used in the previous experiments as a model data set. On the whole the libraries perform similarly, except for STANN and SpatialIndex, the latter of which is especially slow. Compared to the time needed for the NNS (Fig. 7) and the ICP (Fig. 5) the construction time is usually negligible, especially considering that the data structure needs to be constructed only once for the entire registration process.

We see that libnabo is always faster than ANN, sometimes by a small amount, often by at least a factor of two, and sometimes much more. As both libraries implement the same algorithm, this discrepancy is solely due to the compactness of libnabo's data structures, that consume at least 5 times less memory than ANN's. This is important, because modern computer architectures have multiple levels of cache, and thus memory access is often the bottleneck [30]. The compactness provided by libnabo comes at the price of additional and repeated computations during the exploration of the tree. Yet these work on local variables that are easily contained in cache level 1. This shows that the NNS problem is clearly memory-bound in general and that the choice of compactness is sound.

VI. Conclusion

We have made a significant finding regarding the type of search algorithm to use for shape registration. Ranged search queries are ideally suited for shape matching and beat the alternatives in all relevant cases. The reason for this is that shape registration purposefully minimizes the distance to the NN. In the beginning of the registration process the NN range is at its largest. Most ICP iterations are done when the NN range is very small. In fact, the maximal range to begin with is usually only a small fraction of the point clouds' extent. For our artificial data sets a maximal range of $\approx \frac{1}{2}$ constitutes the largest reasonable range for shape matching.

There is yet another complication for the fastest alternatives to ranged search, i.e. FLANN and ANN in these regions. Choosing between the k-NN and the fixed search

is non-trivial. This would involve guessing which of the two is more efficient for a given data set and a given maximal range. Since the runtime of the fixed radius search explodes for large ranges, a shape registration library would have to default to k-NN search. The performance of k-NN search is reasonable on average, but bad in just the region that is important for shape matching.

Since most libraries implement only the k-d tree, it is hard to draw final conclusions as to what data structure is better suited for NNS. The R-tree library SpatialIndex performs about on par to the STANN library. Both were generally slower than the k-d tree implementations. The octree implementation was amongst the best performing algorithms. This effect may not be due to the data structures alone. As Table I shows, SpatialIndex was optimized for GIS systems, STANN for multithreaded application while the octree implementation was designed to be used in shape matching.

We have contributed our own novel open-source implementation of NNS and have shown these to perform well on realistic as well as artificial shape registration problems. We have shown that for similar algorithms, the compactness of data structures plays a critical role and carefully-designed structures can double performances.

SUPPLEMENTAL MATERIAL

We provide the code as well as the data used to perform all experiments in this paper. Everything is available as a subversion repository under http://slam6d.svn.sourceforge.net/viewvc/slam6d/branches/NNS/. The implemented search algorithms are freely available under http://slam6d.sourceforge.net/ and https://github.com/ethz-asl/libnabo.

ACKNOWLEDGMENTS

The authors would like to thank Dorit Borrmann for her support in the data analysis.

REFERENCES

[1] P. Besl 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, February 1992. I

[2] D. M. Mount and S. Arya, "ANN: A Library for Approximate Nearest Neighbor Searching," http://www.cs.umd.edu/~mount/ANN/, October 2011. I, III-B

[3] "CGAL, Computational Geometry Algorithms Library," http://www.cgal.org. I

[4] M. Muja, "FLANN - fast Library for Approximate Nearest Neighbors," http://people.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN, October 2011. I

[5] S. Magnenat, "libnabo," https://github.com/ethz-asl/libnabo, October 2011. I, III, III-B

[6] M. Hadjieleftheriou, "SpatialIndex," http://libspatialindex.github.com/, October 2011. I

[7] M. Connor, "STANN - The simple, Thread-safe Approximate Nearest Neighbor Library," http://sites.google.com/a/compgeom.com/stann/, October 2011. I

[8] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975. II

[9] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129 – 147, 1982. II

[10] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The Grid File: An Adaptable, Symmetric Multikey File Structure," *ACM Trans. Database Syst.*, vol. 9, no. 1, pp. 38–71, 1984. II

[11] M. Connor and P. Kumar, "Fast construction of k-nearest neighbor graphs for point clouds," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 4, pp. 599 –608, 2010. II
[12] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *International Conference on Management of Data*. ACM, 1984, pp. 47–57. II
[13] G. S. Lueker, "A data structure for orthogonal range queries," in *FOCS'78*, 1978, pp. 28–34. II
[14] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, ser. SODA '93. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1993, pp. 311–321. II
[15] R. A. Finkel and J. L. Bentley, "Quad Trees - a Data Structure for retrieval on Composite Keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974. II
[16] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '90. New York, NY, USA: ACM, 1990, pp. 322–331. II
[17] S. Blumenthal, E. Prassler, J. Fischer, and W. Nowak, "Towards identification of best practice algorithms in 3D perception and modeling," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 3554 –3561. II
[18] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart, "Tracking a depth camera: Parameter exploration for fast icp," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*. IEEE Press, 2011, pp. 3824–3829. II
[19] S. P. Dandamudi and P. G. Sorenson, "An empirical performance comparison of some variations of the kd tree and bd tree," *International Journal of Parallel Programming*, vol. 14, pp. 135–159, 1985. II
[20] Y. Nakamura, S. Abe, Y. Ohsawa, and M. Sakauchi, "A balanced hierarchical data structure for multidimensional data with highly efficient dynamic characteristics," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 5, no. 4, pp. 682–694, aug 1993. II
[21] M. A. Greenspan, G. Godin, and J. Talbot, "Acceleration of binning nearest neighbor methods," 2000. II
[22] M. Greenspan and G. Godin, "A nearest neighbor method for efficient icp," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, 2001, pp. 161–168. II
[23] Automation Group (Jacobs University Bremen) and Knowledge-Based Systems Group (University of Osnabrück), "3DTK – The 3D Toolkit," http://slam6d.sourceforge.net/, February 2011. III
[24] J. Elseberg, D. Borrmann, and A. A. Nüchter, "Efficient processing of large 3d point clouds," in *Proceedings of the XXIII International Symposium on Information, Communication and Automation Technologies*, ser. ICAT '11, 2011. III-A
[25] S. F-Frisken and R. N. Perry, "Simple and Efficient Traversal Methods for Quadtrees and Octrees," *Journal of Graphics Tools*, vol. 7, no. 3, 2002. III-A, III-A
[26] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. ACM*, vol. 45, pp. 891–923, November 1998. III-B
[27] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010. III-B
[28] S. Maneewongvatana and D. M. Mount, "It ' s okay to be skinny , if your friends are fat," *ReCALL*, pp. 1–8, October 1999. IV
[29] S. Arya and D. M. Mount, "Approximate nearest neighbor queries in fixed dimensions," in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, ser. SODA '93. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1993, pp. 271–280. IV
[30] D. Molka, D. Hackenberg, R. Schone, and M. S. Muller, "Memory performance and cache coherency effects on an intel nehalem multiprocessor system," *Parallel Architectures and Compilation Techniques, International Conference on*, vol. 0, pp. 261–270, 2009. V

Jan Elseberg is a research associate and Ph.D. student at the Jacobs University Bremen. Past affiliations were with the Temple University in Philadelphia and the University of Osnabrück, from which he received the Master degree in computer science in 2009 and a Bachelor degree in computer science and mathematics in 2006. His research interests include 3D environment mapping, 3D vision, laser scanning technologies and data structures. http://plum.eecs.jacobs-university.de/~jelseber/

Stéphane Magnenat is senior researcher at the Autonomous Systems Lab in ETH Zurich. He received the M.Sc degree in computer science (2003) and the Ph.D. degree (2010) from École Polytechnique Fédérale de Lausanne (EPFL). His research interests include software architecture, software integration and scalable artificial intelligence on mobile robots. Stéphane Magnenat is enthusiastic about open source software as a mean to advance mobile robotics and its adoption. He is a member of the IEEE. http://stephane.magnenat.net

Roland Siegwart is a full professor for Autonomous Systems and Vice President Research and Corporate Relations at ETH Zurich since 2006 and 2010 respectively. He has a Master in Mechanical Engineering (1983) and a PhD in Mechatronics (1989) from ETH Zurich. In 1989/90 he spent one year as postdoctoral fellow at Stanford University. After that he worked part time as R&D director of MECOS Traxler AG and lecturer at the Institute of Robotics, ETH Zürich. From 1996 to 2006 he was associate and later full professor for Autonomous Microsystems and Robots at the Ecole Polytechnique Fédérale de Lausanne (EPFL). Roland Siegwart is member of the Swiss Academy of Engineering Sciences, IEEE Fellow and officer of the International Federation of Robotics Research (IFRR). He served as Vice President for Technical Activities (2004/05) and was awarded Distinguished Lecturer (2006/07) and is currently an AdCom Member (2007-2010) of the IEEE Robotics and Automation Society. He leads a research group of around 30 people working in the fields of robotics, mechatronics and product design. Roland Siegwart was a general chair of several conferences in robotics including IROS 2002, AIM 2007, FSR 2007, ISRR 2009 and is a co-founder of multiple successful spin-off companies in robotics. http://www.asl.ethz.ch/

Andreas Nüchter holds an assistant professorship at Jacobs University Bremen. Before he was a research associate at University of Osnabrück. Further past affiliations were with the Fraunhofer Institute for Autonomous Intelligent Systems (AIS, Sankt Augustin) and University of Bonn, from which he received the diploma degree in computer science in 2002 (best paper award by the German society of informatics (GI) for his thesis). He holds a doctorate degree (Dr. rer. nat) from University of Bonn. His work focuses on robotics, cognitive systems and artificial intelligence. His main research interests include reliable robot control, 3D environment mapping, 3D vision, and laser scanning technologies, resulting in fast 3D scan matching algorithms that enable robots to map their environment in 3D using 6 degrees of freedom poses. The capabilities of these robotic SLAM approaches were demonstrated at RoboCup Rescue competitions, ELROB and several other events. He is a member of the GI and the IEEE. http://www.nuechti.de

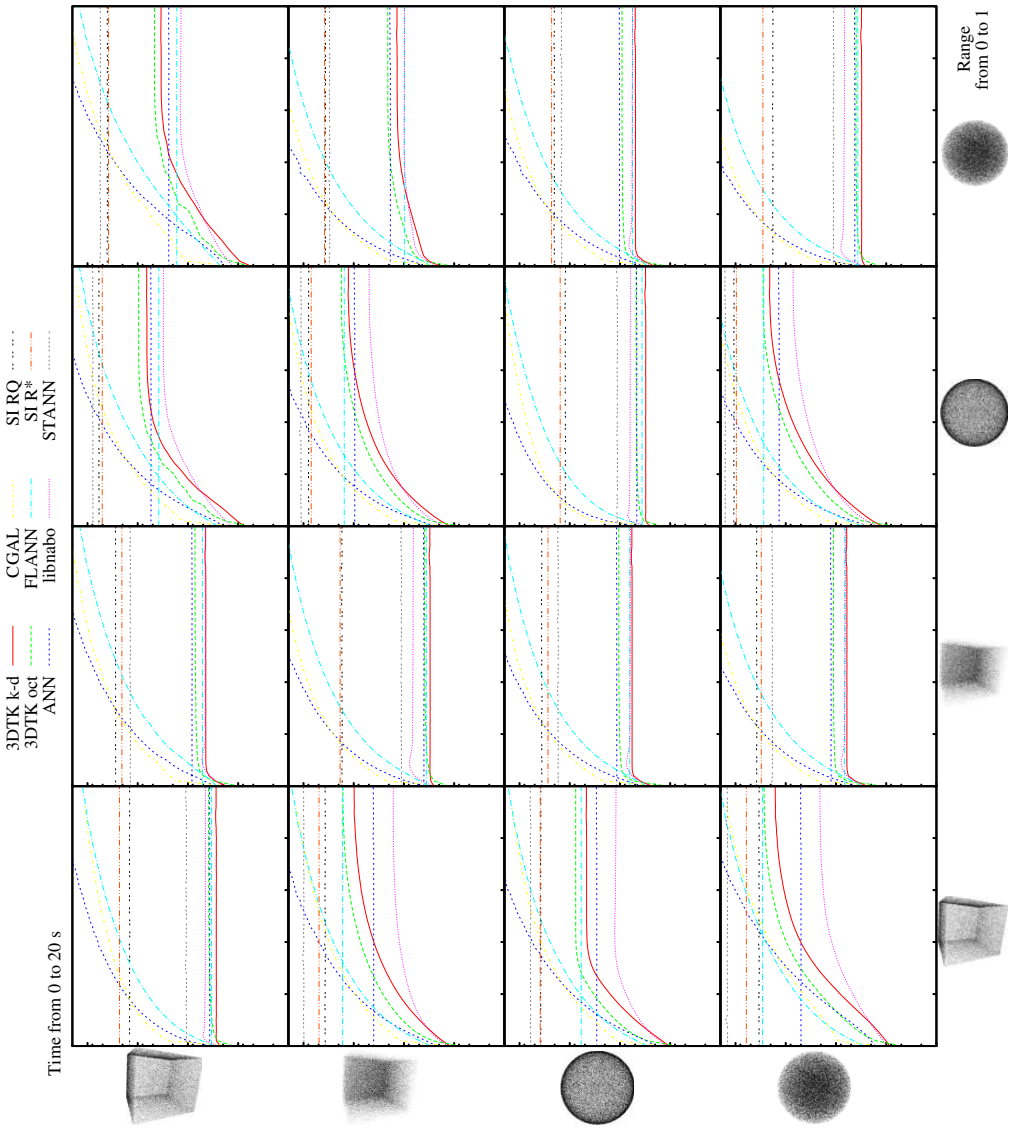


Fig. 6. The results from all algorithms for the artificial data sets on a logarithmic scale. The images on the x-axis indicate the data set that was stored in the data structure (model set). The images on the y-axis indicate the query point set. The query time in seconds required for the entire data set is plotted on a logarithmic scale for all settings of the maximal allowed distance. The maximal distance was altered between 0 and 1 in steps of 0.02. Construction time for the data structures is not included.

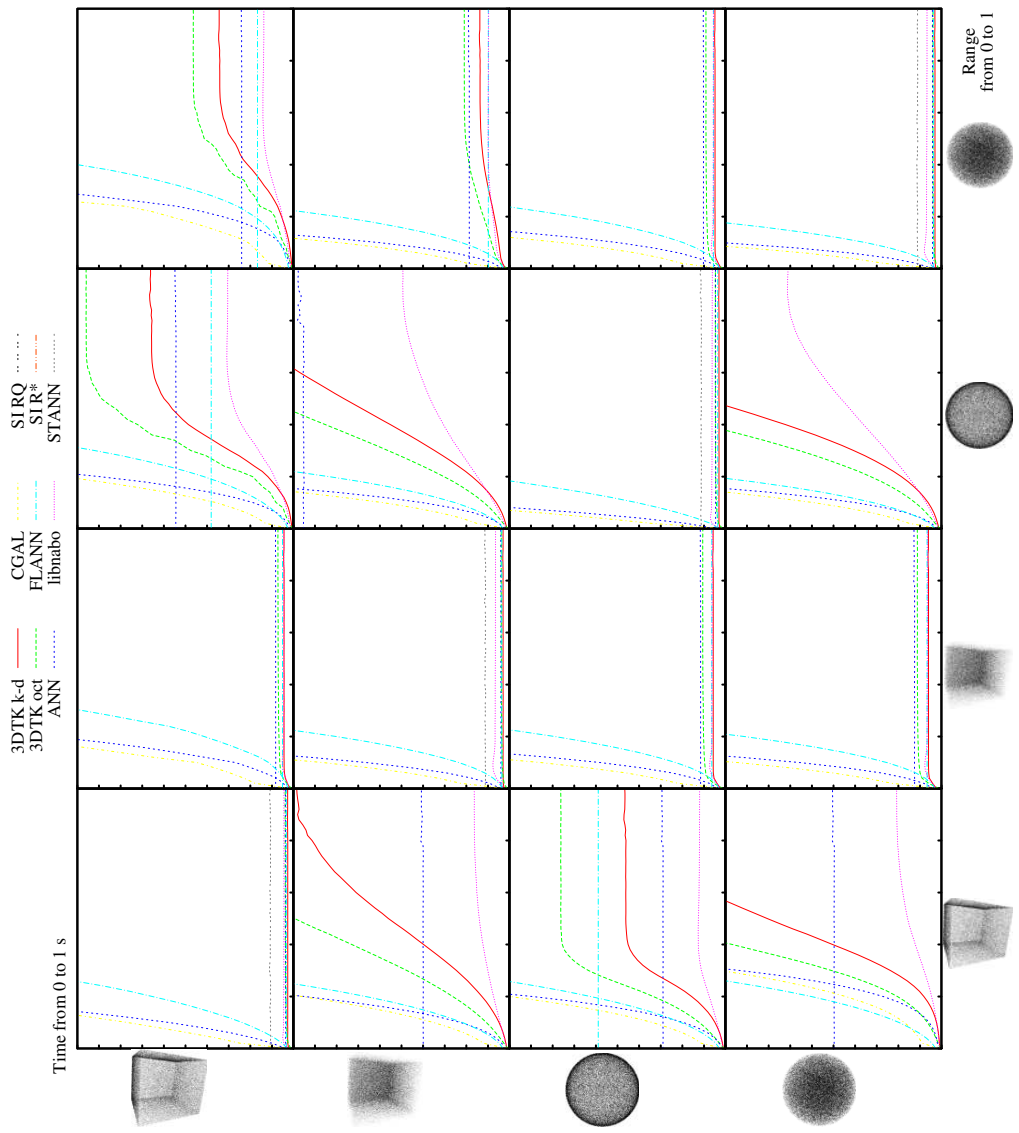


Fig. 7. The results from the best performing algorithms for the artificial data sets on linear scale. Note that the scale has also been changed from 0 to 1 s.

Chapter 3

Mobile Laser Scanning

> Bleibe nicht am Boden heften, Frisch
> gewagt und frisch hinaus!
> Keep not standing fix'd and rooted.
> Briskly venture, briskly roam!
>
> Johann Wolfgang von Goethe

Mobile laser scanning (MLS) refers to the practice of acquiring range measurements while in motion. This is in contrast to terrestrial laser scanning, where the laser scanner remains static during scan acquisition. Mobile laser scanning is similar to airborne laser scanning in this regards except that it is usually only applied to ground based mapping. Both MLS and airborne laser scanning are also sometimes called kinematic laser scanning. Although very similar there are some distinctions between the two beyond the obvious. Due to the different scenarios the laser range finders will have different distances to the environment. This requires that the laser scanners have a high range of several hundreds of meters for airborne laser scanning, whereas a maximum range of less than 100 meters is often enough for MLS. Similarly, measurement precision in airborne laser scanners is in the order of cm instead of mm as it is often the case in MLS. In addition to this, some scanning modes that are possible in mobile mapping are infeasible in an airborne scenario.

Still, the two laser scanning techniques have some things in common as well. The most important one is that the motion of the laser scanner during acquisition is highly uncertain. This makes accurate registration into a global coordinate system highly challenging. Both mobile and airborne laser scanning systems are therefore usually equipped with sensors that measure their position and orientation in the global coordinate system at all times. The sensor information is then combined with the range measurements to create a representation of the environment. This process is called motion compensation.

Although the term mobile laser scanning is relatively recent, and in the surveying community refers solely to ground based vehicles equipped with laser scanners, the general idea of mobile mapping is far from new. For example, the first close-up images of Jupiter were acquired with a single-pixel photodetector that was fixed to the satellite Pioneer 10 [70]. The satellite rotated around its axis while it passed the gas giant at a distance of more than 100.000 km [63]. The

Fig. 3.1: A composite of several images of Jupiter that were acquired by a single-pixel camera on board of the Pioneer 10 probe during the December of 1973 as taken from [71]. The composite of Jupiter was constructed from Pioneer 10 data and shows the growth in the apparent size of the giant planet as the spacecraft approached, and its receding crescent as the spacecraft passed behind it.

sensor was positioned off-axis to the satellite such that the opening moved in a helical motion. Pioneer 10 rotated at a speed of about 288 rotations per hour. The velocity of the satellite with respect to Jupiter was about 130.000 km/h, so that the movement of the space craft had to be taken into account to produce the undistorted images of the planet as seen in Fig. 3.1.

Classical mobile and airborne laser scanning works similar to this. In both fields the motion of the vehicle that is equipped with the laser scanner is crucial to acquiring the data. The laser scanner on its own would usually not be able to acquire fully three dimensional images of the environment. In airborne laser scanning a 2D laser scanner is mounted on the underside of an airplane. During flight the downward looking 2D laser scanner is moved orthogonally to its measuring plane thus creating a 3D point cloud. The setups for MLS are more varied. In this area 2D laser scanners are employed as well. However, the state of the art for commercial mobile laser scanners is to use two 2D laser scanners whose measuring planes are perpendicular to each other. These are then mounted on a vehicle, usually an automobile, although boats and other car-like vehicles are also a possibility. The scanners are oriented in a 45° angle towards the direction of movement of the vehicle and more or less orthogonally to the ground plane. This setup is depicted in Fig. 3.2 alongside other less popular setups. Sometimes, only one of the two 2D laser scanners is used in order to save costs. In this case the scanner is frequently oriented differently to the previous setup.

In addition to these two airborne-like setups there is one more setup that is only possible in MLS. Instead of one or multiple 2D scanners, a 3D laser scanner is mounted on a vehicle and is used in continuous mode, i.e., the sensor acquires multiple complete 3D scans without stopping in-between. This has advantages and disadvantages that will be more closely examined

Fig. 3.2: A line-up of different types of MLS systems. The conventional mobile scanner setup with up to two 2D laser scanners mounted symmetrically on cars is shown in the top. Top left: The VMX-450 by RIEGL Laser Measurement GmbH contains two 2D laser scanners of type VQ-450 and up to six cameras. The position and orientation of the car is estimated by an Applanix system that combines data from odometry, an IMU and a GPS unit. The VQ-450 scanners are pulsed laser scanners, that acquire 550.000 points per second and 200 scans per second. Top right: The Lynx Mobile Mapper by Optech has up to two 2D laser scanners and up to four cameras. The trajectory of the vehicle is computed by the Applanix POS L with integrated odometry, IMU and GPS unit. Optech is also using pulsed laser scanners. These collect data at 500.000 points per second and 200 scans per second. Bottom left: The Pegasus SM by Leica Geosystems is a combination of six cameras, one or two HDS6200 (as in Table 1.1) and a Novatel SPAN GPS, IMU and odometry system for trajectory estimation (image from Leica Geosystems AG). This is a rare case of a system to use phase-based laser scanners. Note, that even though the HDS6200 is a 3D terrestrial laser scanner it is purely used as a 2D laser scanner in the Pegasus SM. Bottom right: An experimental setup for a mobile laser scanner by the RIEGL Measurement GmbH using a single VZ-400 3D laser scanner in continuous mode. The trajectory of the car is estimated with the integrated inclination sensor and GPS unit of the scanner.

in the following chapter, where the robotic mobile laser scanner Irma3D is presented. This kind of setup for MLS systems is relatively new. In fact the surveying community only just began experimenting with such setups as a cheaper alternative to traditional mobile laser scanners. One such experimental system is shown in Fig. 3.2. In the robotics community, these designs are somewhat more common. Cole et al. [13] were on of the first to use a continuously pitching laser scanner to acquire 3D data while the platform is in motion. The range measurements are filtered into discrete point clouds that are rigidly registered.

Whether the mobile mapping system is a setup using one or two 2D laser scanners or a single 3D laser scanner, they come with additional challenges. The most obvious consequence of moving the range sensor during acquisition is that the problem of registration becomes significantly more challenging. No longer are there few discrete positions and orientations that need to be estimated for large point clouds. In mobile mapping, every range measurement is acquired at a different location such that every point could be moved independently of each other in the registration process. The previous rigid registration problem becomes a non-rigid registration problem in this scenario. The puzzle analogy from Chapter 2.2 is thus nearing its breaking point. Individual puzzle pieces are dissolved to individual dots of color that could be arranged anywhere. The current state of the art for solving this problem is via hardware: Very expensive equipment is used to measure the position and orientation of the vehicle at every instance of measurement. Most systems combine high quality odometry measurements with highly precise IMUs and GPS units to achieve this. In Section 3.2 a novel algorithmic approach to solving the non-rigid registration problem is presented. Although, some trajectory estimate via odometry, IMU, GPS or other sources is required to bootstrap the algorithm, highly expensive equipment is not necessary for achieving acceptable results. In the presence of high quality trajectory estimates, the algorithm is also capable of improving the trajectory further.

A second challenge that arises with MLS is that of calibration. Calibration is the problem of finding the numerical values for certain real world quantities that are necessary for correctly interpreting the sensors measurements. This problem exists as well for terrestrial laser scanning, where quantities such as the position and orientation of the laser emitter and receiver or the shape and orientation of the mirrors within the laser scanner must be known. These values remain relatively constant throughout the lifetime of the sensor. Their dependency on outside temperature and air pressure can also be measured, such that these devices are calibrated by the manufacturer and are usually recalibrated only once a year. For MLS systems several additional quantities must be known. As each mobile laser scanner is essentially a combination of several sensors not only must each sensor itself be calibrated, but the position and orientation offsets between them must also be known. The large number of quantities as well as the much more dynamic environment often necessitates the daily calibration of the mobile scanner. As such, fast methods for calibration are a more pressing concern for MLS than for static laser scanning. We also present a general solution to the calibration problem in Section 3.2.

3.1 Irma3D

The Intelligent Robot for Mapping Applications in 3D (Irma3D) is a robotic mobile laser scanner that was developed for the purpose of exploring issues like registration and calibration in a mobile

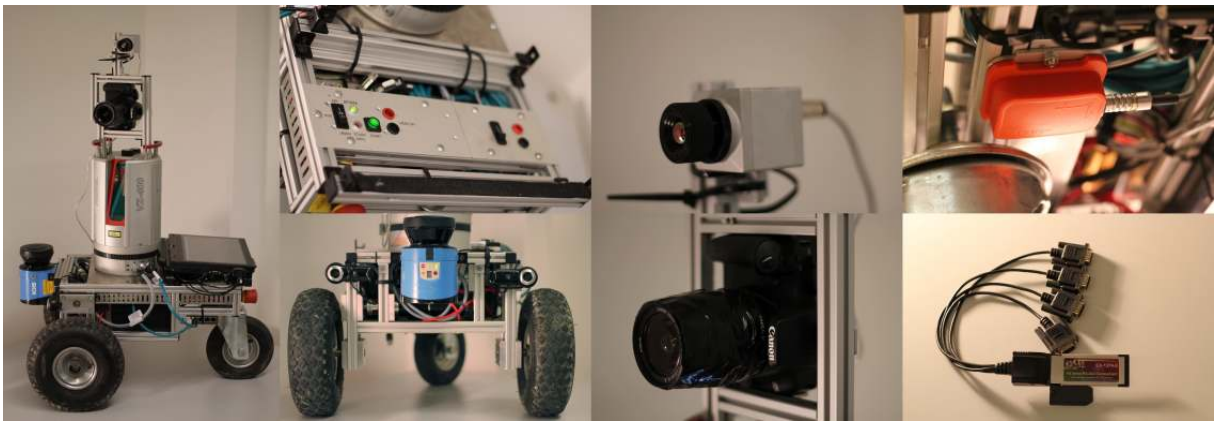


Fig. 3.3: Images of Irma3D. Left: Side view of Irma3D with all of its sensors and equipment. Top left: The control panel with switches for the two electric circuits, sockets for loading the batteries and the laptop mount. Bottom left: Front view of Irma3D. Two small digital cameras and a 2D laser scanner are attached to the front of the robot for easier navigation. Top middle: The thermal camera on top of the 3D laser scanner that is used to sense thermal properties of the environment. Bottom middle: A DSLR camera is mounted on the 3D laser scanner to provide color information in the point clouds. Top right: The IMU is mounted on the underside of the chassis next to the rear wheel to provide maximum shielding from the magnetic fields that are generated from the motors and the 3D laser scanner. Bottom right: An ExpressCard to RS-232 Adapter is used so that modern laptops without serial ports can interface with the Volksbot motor controller and the IMU device.

laser scanning scenario. Irma3D is a small, battery-powered, light weight, three wheeled vehicle. Irma3D and its components are depicted in Fig. 3.3. With a width of 52 cm it is small enough to pass through narrow doorways. The three wheel design allows for a high maneuverability such that it can rotate on the spot. These properties make Irma3D ideally suited for indoor environments. In addition, the high-powered electrical two-wheel drive powered by two 150 W DC motors by Maxon with a top speed of about 2.2 m/s combined with the 26 cm wide pneumatic wheels also make it capable of operating in moderately challenging outdoor environments. The robot can be remote-controlled, either via a W-LAN connection or through a Logitech Wireless Gamepad F710 or similar devices. Irma3D can also be used in a fully autonomous mode. Once activated, Irma3D will attempt to explore its surroundings, up to some preset limits, and create a 3D map of the environment [17].

As a laser scanner platform, it can be used to acquire range measurements while moving through the environment. Alternatively, the robot can remain stationary when a 3D point cloud is acquired. This type of static laser scanning is called stop-and-go scanning. It is possible to create 3D models of the environment as detailed as with MLS using such methods as presented in chapter 2. However, since the laser scanner is not operating while the robot is moving, more time is required in this mode to create equally large point clouds. This dual-use of Irma3D is made possible by the 3D terrestrial laser range finder that it is equipped with. Without a 3D scanner that is able to freely rotate, Irma3D could not acquire 3D range images of its environment without moving.

The robot Irma3D is a combination of several sensors, a mobile platform and a portable laptop for processing data and controlling the robot itself. The chassis of the robot is a modified Volksbot RT-3 [74]. The Volksbot RT-3 has two front wheels. Each is actuated by a single Maxon servo motor. Together, the two motors are powerful enough to move the robot at a total maximum velocity of 2.2 m/s. The third wheel is in the back of the chassis and is swivel-mounted and thus completely passive as it follows the directions of the front wheels. The platform is powered by four 12 V 7.2 Ah Panasonic lead-acid batteries. This is two more than in the original design. The chassis has been modified to provide two electric circuits. In addition to extra wiring the modifications include additional elements to the control panel in the back of the Volksbot RT-3. The platform has a variable laptop mount that can fit any reasonably sized laptop. Currently Irma3D operates on a Samsung Q45 Aura laptop with an Intel Core 2 Duo T7100 and 4 Gb of RAM. The laptop mount has been situated such that the laptop will rest above the control elements of the chassis (see Fig. 3.3). Therefore, the location of the kill switch has been changed to the rear of the platform to keep it accessible at all times. The physical dimensions of the Volksbot platform are 58 cm x 52 cm x 31.5 cm with a weight of about 25 kg. A large part of this weight is from the lead batteries, each contributing about 2.5 kg.

For navigation and obstacle avoidance, the robot is equipped with a SICK LMS 100 [66]. This 2D laser scanner is mounted at the front of the chassis and is facing forward, acquiring 2D range scans at a rate of 50 Hz. To fully exploit the 270° field of view of the SICK LMS 100, the sensor head is position slightly above the chassis. The SICK LMS 100 scans with a resolution of 0.5° and a maximum effective range of about 20 m. To support a human operator when the robot is remote controlled two small webcams of type QuickCam Pro 9000 by Logitech [46] are also attached to the front of the chassis. The motors of the Volksbot are equipped with rotary encoders to measure wheel rotations. This information is used to provide pose estimates of the robot via odometry. The pose estimates are improved using data from the Xsens MTi IMU device [9] that is also attached to the robotic platform. The IMU is susceptible to magnetic interference and must be positioned away from strong magnetic fields to reduce erroneous sensor readings. The motors as well as the laser scanners generate magnetic fields. Therefore, the IMU is fixed to the rear and bottom of the chassis. The central sensor of Irma3D is the 3D laser scanner VZ-400 by RIEGL Measurement GmbH [69]. The scanner is mounted on top of the Volksbot RT-3 chassis. Attached to the top of the scanner is a Canon 1000D DSLR camera [37]. After a 3D scan has been acquired the camera is used to acquire color information for the point cloud. A similar process is done using the Optris PI160 thermal camera [28] which is also mounted on top of the VZ-400 to acquire information about the thermal properties of structures in the point cloud. Currently, these two cameras can only be used when scanning in stop-and-go mode to avoid the data cables from coiling up. All cameras on Irma3D are USB devices. With the addition of the Logitech Wireless Gamepad F710 for remotely controlling the robot this adds up to a total of 5 USB plugs that need to be connected. To reduce requirements on the laptop, Irma3D is equipped with a USB hub. Both laser scanners transfer information via Ethernet. To enable the laptop to connect to both devices at the same time, Irma3D is also provided with a network switch. The Volksbot motor controller as well as the Xsens MTi communicate via RS232 serial ports. Since modern laptops are rarely equipped with an RS232 port, let alone two, a Delock PCMCIA to RS232 adapter [18] is used to allow for communication. The laptop supplies its own power via the laptop battery. All USB devices draw their power from the laptop.

The network switch as well as all other sensors with the exception of the VZ-400 share a power supply with the Volksbot chassis in the form of two of the four lead batteries. The remaining two lead batteries are dedicated to the VZ-400, as it draws a similar amount of power to the rest of the system.

The VZ-400 is able to freely rotate around its vertical axis to acquire 3D scans even when the robot is not in motion. The fastest it can do this is at 6 s per rotation. At this speed each point cloud will contain about 750.000 points. The minimum angular resolution of a range scan is 0.0024° in both directions. Given that the scanners vertical field of view is 100°, this equates to more than 6 billion points per scan. The scanner is capable of online Full Wave Transform, may record multiple distance measurements per laser beam and will return not only the range of each echo, but also their amplitude, deviation and a calibrated reflectance value. Amplitude and deviation refer to the parameters of the normal distribution that is fitted into each response. The deviation of a point is a measure for the certainty of the measurement. The amplitude is roughly equivalent to the intensity with which the laser beam was reflected. This value is similar to what other laser scanners return as reflectance, reflectivity or intensity. It should be noted that this is not a good measure of the reflection coefficient of a surface. The intensity of a signal is affected not only by the reflectance properties of the surface but also by the angle of incidence, the distance to the surface, the temperature and other atmospheric conditions. The VZ-400 also returns so-called calibrated relative reflectance values, which attempts to correct for the influence of the distance to the surface.

Irma3D is a mobile laser scanner that acquires range measurements in a manner decidedly distinct from conventional MLS systems. When Irma3D is in motion, its continually spinning 3D laser scanner will scan the environment in a spiral pattern instead of a regular grid. We will now examine what this different mode of MLS entails. One advantage has already been mentioned. An MLS system with a rotating 3D scanner can also be used in stop and go mode to acquire range scans. In fact, this is not a binary either-or decision. Both modes can be used in combination with each other. This helps registration, since scans acquired while the vehicle is stopped can be localized more easily. Range measurements taken while in movement can then be used to improve the density of the 3D model. The laser scanner is then always in use with no wasted idle time like in static scanning.

Other advantages and disadvantages can be more easily explained when visualizing the scan results that different types of MLS systems acquire in the same environment. For this purpose a simple scanning scenario in a tunnel has been simulated. The 20 m long tunnel is in the shape of a cylinder with a radius of 4 m that is cut in half precisely along its axis. Each mobile laser scanner traverses the tunnel along its axis. This scenario as well as the point clouds that result from this simulation are depicted in Fig. 3.4. It should be noted, that other designs for MLS systems are conceivable as well. For example, a design similar to the stationary 3D scanner by Ohno et al. [58] could theoretically be adapted to mobile purposes. In their design a 2D laser scanner is actuated by two servo motors. The two rotational degrees of freedom allow for a variety of more complex scanning patterns. Mobile laser scanning systems that employ rotating 3D laser scanners are easily identifiable by their characteristic spiral pattern. 2D laser scanners in motion always generate regular grid patterns. This is irrespective of how many scanners are used or how they are mounted on the mobile platform. Systems like Irma3D exhibit a more uneven distribution of points.

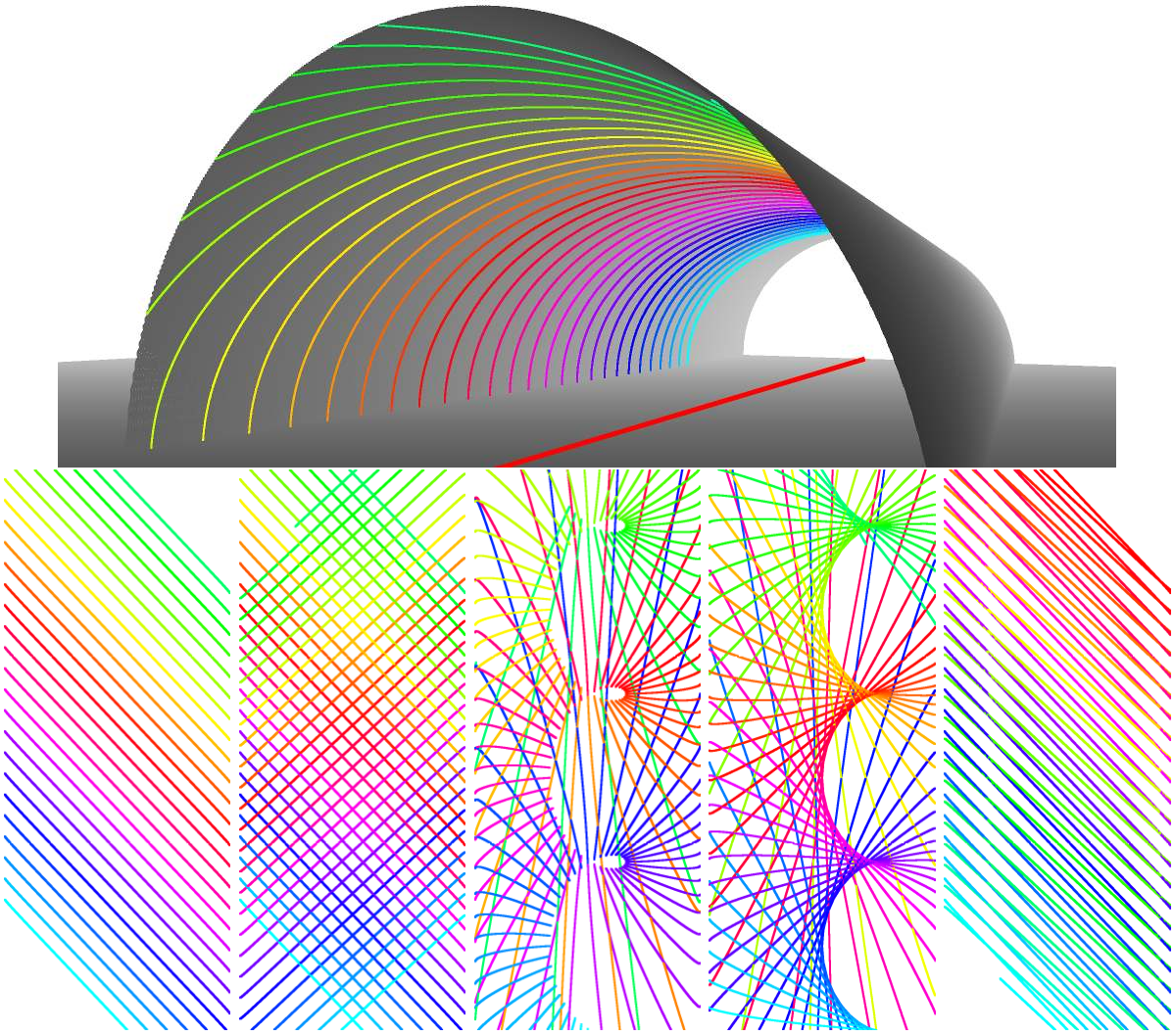


Fig. 3.4: Five different principles of scanning a tunnel were simulated. Top: The tunnel is half a cylinder, truncated with a plane that is collinear to the axis of the cylinder. The mobile laser scanner moves along the axis (red) and acquires range measurements. These are displayed in an orthographic projection as seen from exactly above the tunnel. The points are colored according to the time that they were acquired (from cyan to green). The results of five simulated mobile laser scanners from left to right: 1. A single 2D laser scanner rotated around the yaw axis, such that there is a 45° angle between the laser plane and the direction of movement. 2. Two 2D laser scanners rotated around the yaw axis, such that they form a right angle to each other and a 45° angle between each laser plane and the direction of movement. 3. A 2D laser scanner looking sideways, i.e., with a field of view of −90° to 90° is rotating continuously and in a clockwise fashion around the yaw axis of the mobile platform. 0° is defined as the angle parallel to the floor. 4. A 2D laser scanner looking up, i.e., with a field of view of 0° to 180° is also rotating in the same fashion. 5. The same mobile laser scanner as in 1. except it traverses the tunnel twice. Once from the bottom to the top and then from the top back to the bottom.

The point density on a planar surface is only dependant on the distance and angle of that surface in relation to the laser scanner. This holds true for mobile systems with 3D laser scanners as well. In contrast to systems with 2D scanners though, the angle of a surface to the laser scanner varies with time even when the orientation of the mobile platform does not change. Due to the ever changing orientation of the laser beam with respect to the mobile platform, some time will be spent measuring "ahead" and "behind" of the platform. This has the disadvantage of less measurements falling on the side of the platform. However, details in the environment that can only be measured from certain angles will more likely be picked up by a rotating laser scanner. This issue often arises in urban environments, where many planar surfaces like street signs, fences or even walls may be aligned in an unfortunate manner towards the laser scanner. A street sign that is coplanar with or close to coplanar with the laser plane of a conventional mobile scanner is near invisible. With a rotating 3D scanner this is not the case.

Another advantage for Irma3D is that the laser beam always "returns" to previously scanned objects in the environment. Note how much the timestamps of measurements near to each other vary for systems like Irma3D in Fig. 3.4 in contrast to conventional mobile scanners. At both ends of the tunnel this type of MLS system can acquire range measurements from the other end of the tunnel. This has implications for the registration of the point clouds that will become more obvious in the next sections. To be clear, only measurements taken at different times that are correlated to each other and can be identified to be correlated can be used to compute improved estimates for the position and orientation of the laser scanner at those times. Although all range measurements from a static environment naturally correlate to each other this correlation is not so simple to extract. Point measurements that stem from a single feature in the environment, or from surfaces that are near to each other, are much easier to identify even in noisy data. It is therefore important for registration to use a system that is capable of acquiring such seemingly redundant measurements. A conventional mobile scanner with multiple 2D scanners is capable of this, although to a very limited extent because only a very short period of time elapses between the pass of the first 2D scanner and the passes of the other 2D scanners. These systems only allow for a trajectory correction in these minuscule time steps.

Finally, there is one more difference in the distribution of range measurements between the two types of mobile laser scanners. The effect is clearly visible when varying the speed of the mobile platform in the simulation. This was done in Fig. 3.5. The faster the platform, the less time is spent in any given environment. Since the measuring rate remains constant, less points are acquired as a consequence. As with conventional mobile scanners the point density decreases the faster the platform moves. However, as can be seen in the right of Fig. 3.5 the point distribution to the left and to the right of the mobile laser scanner is not equal. This effect is due to the direction of rotation of the 3D laser scanner. In the simulation the 3D scanner was rotating in a clockwise fashion. Thus, the sweeping motion of the scanner on the left side of the platform coincides with the direction of movement of the platform. On the right hand side of the platform both directions are opposed to each other. Consequently, points on the left hand side are more evenly distributed than on the right hand side.

Mathematically, this can be explained as follows. The apparent velocity $v_l(t)$ of a laser beam at time t on a wall that is parallel to the direction of motion of the platform and at a distance

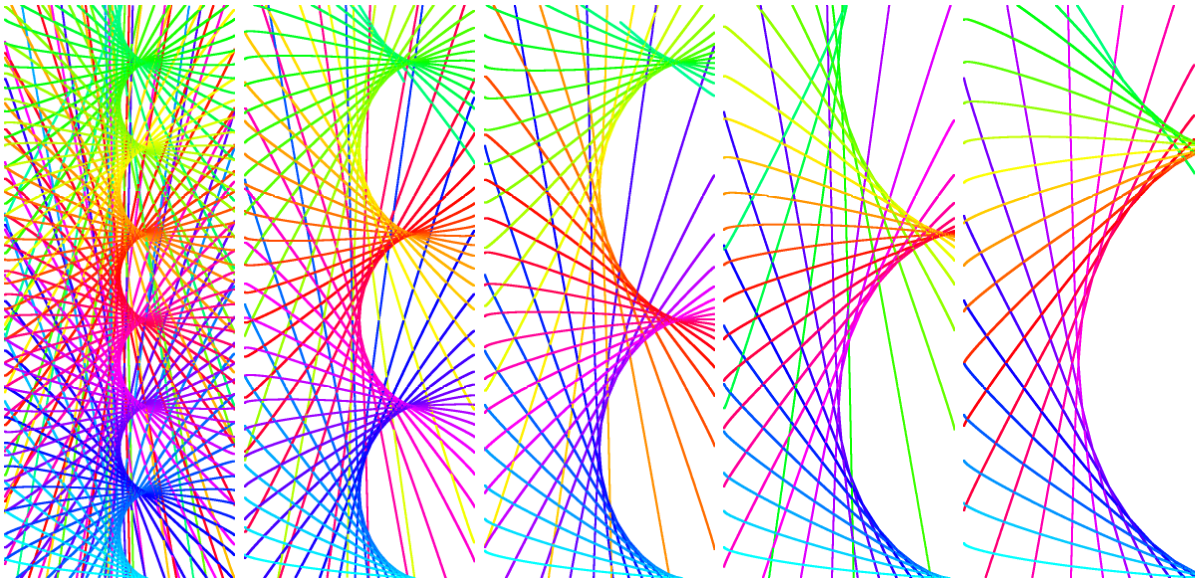


Fig. 3.5: Comparison of the effect of movement speed on the distribution of points for mobile laser scanners of type 4 (see Fig. 3.4). The scenario is the same cylindrical tunnel as in Fig. 3.4. The tunnel has a length of 20m, a radius of 4m. The 2D scanner requires 6 seconds per rotation and takes 40 2D scans per rotation. This last value is unrealistically low and was chosen for the purpose of illustration only. From left to right, the mobile platform moved at a speed of, 1 m/s, 2 m/s, 3 m/s, 4 m/s and 5 m/s.

of D to the laser scanner on its left hand side is given by:

$$v_l(t) = v_m + v_s D \left(1 + tan^2(v_s t)\right). \tag{3.1}$$

This assumes that the laser scanner is rotating in a clockwise fashion. Here v_m describes the velocity of the platform and v_s describes the rotation velocity of the laser scanner. The equivalent apparent velocity $v_r(t)$ of the laser beam on the right hand side is given by:

$$v_r(t) = v_m - v_s D \left(1 + tan^2(v_s t)\right). \tag{3.2}$$

It is clear that $v_l(t)$ will always remain positive, whereas $v_r(t)$ can be both negative and positive when the forward velocity of the platform is too low, the distance to the surface is too short or the rotation velocity of the scanner is too high. See Fig. 3.6 for a plot of both functions, with D set to 1 m, v_m to 1 m/s and v_s set to $\frac{1}{6}$ s. This represents a challenging indoor scenario for a relatively swiftly moving Irma3D. The interesting range of angles is around 0°, where the laser is perpendicular to the surface. Absolute angles greater than 80° can be considered unimportant for objects to the side of the mobile laser scanner. The dominating part of both functions is the squared tangent. However, the addition of v_m means that the average velocity around 0° is high for the left side and quite low for the right side. Since the tangent tends towards infinity the average velocity for large absolute angles is similarly high for both sides. The result is that on the right hand side the laser beam quickly approaches a position, remains in that general location for a while and then quickly leaves it again.

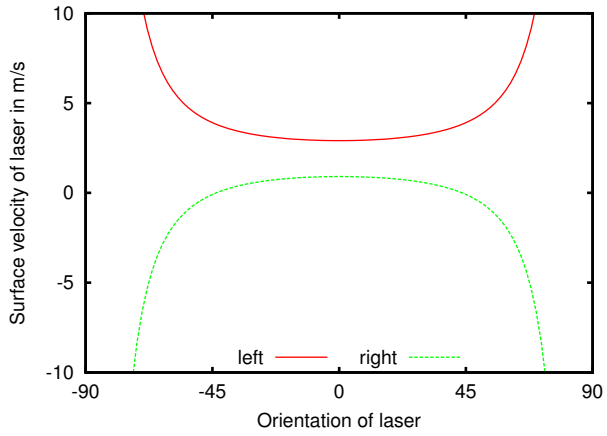


Fig. 3.6: The velocity of the laser beam on surfaces parallel to the motion of the platform and at a distance of 1 m. The platform moves at a speed of 1 m/s, whereas the laser scanner requires 6 s for a single rotation in a clockwise fashion.

While this effect should be kept in mind when choosing the movement speed of Irma3D for specific environments it should be noted that the previous analysis assumes a highly simplified scenario. Furthermore the effect is much less pronounced in reality than what Fig. 3.5 suggests due to the much higher scan rate of the terrestrial laser scanner.

3.2 Algorithmic Solutions for computing accurate maximum likelihood 3D Point Clouds from Mobile Laser Scanning Platforms

The next article deals with the aforementioned issues that mobile laser scanners face, namely that of calibration and registration. The paper presents and evaluates novel algorithmic solutions to these problems. It has recently been accepted for publication in the Journal of Remote Sensing (`http://www.mdpi.com/journal/remotesensing`).

OPEN ACCESS
remote sensing
ISSN 2072-4292
www.mdpi.com/journal/remotesensing

Algorithmic solutions for computing precise maximum likelihood 3D point clouds from mobile laser scanning platforms

Jan Elseberg [1,*], Dorit Borrmann [2] and Andreas Nüchter [2]

[1] Automation Group, School of Engineering and Science, Jacobs University Bremen gGmbH, Campus Ring 1, 28759 Bremen, Germany

* Author to whom correspondence should be addressed; j.elseberg@jacobs-university.de

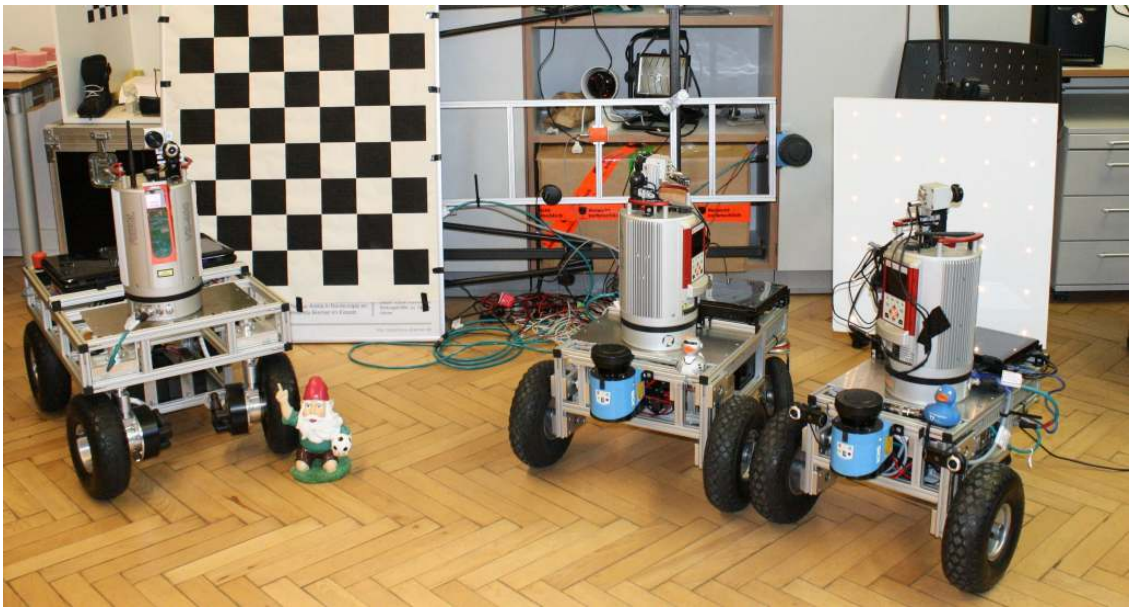
Abstract: Mobile laser scanning puts high requirements on the accuracy of the positioning systems and the calibration of the measurement system. We present a novel algorithmic approach for calibration with the goal of improving the measurement accuracy of mobile laser scanners. We describe a general framework for calibrating mobile sensor platforms that estimates all configuration parameters for any arrangement of positioning sensors including odometry. In addition, we present a novel semi-rigid Simultaneous Localization and Mapping (SLAM) algorithm that corrects the vehicle position at every point in time along its trajectory, while simultaneously improving the quality and precision of the entire acquired point cloud. Using this algorithm the temporary failure of accurate external positioning systems or the lack thereof can be compensated for. We demonstrate the capabilities of the two newly proposed algorithms on a wide variety of data sets.

Keywords: mobile laser scanning; non-rigid registration; calibration; mapping; algorithms

1. Introduction

Laser scanning provides an efficient way to actively acquire accurate and dense 3D point clouds of object surfaces or environments. Mobile scanning is currently used for modeling in architecture and agriculture as well as urban and regional planning. Modern systems like the Riegl VMX-450 and the Optech Lynx Mobile Mapper work along the same basic principle. They combine a highly accurate Global Navigation Satellite System (GNSS), a high precision Inertial Measurement Unit (IMU) and the odometry of the vehicle to compute fully timestamped trajectories. Using a process called motion compensation this trajectory is then used to georeference the laser range measurements that were

Figure 1. A variety of mobile laser scanners exists.



(a) Custom built robots on a volksbot chassis using the Riegl VZ-400 as its main sensor. From left to right the robots are Achim3D, Lars3D and Irma3D.

(b) 3D mobile mapper based on a continuously rotating VZ-400, a low-cost inertial measurement unit (IMU), global positioning system (GPS), and an on-board diagnosis device (OBD-II).

(c) The TopScan GmbH Mobile laser scanner using the Optech Lynx Mobile Mapper. The Lynx mobile mapper employs twin 2D laser scanners and pose estimation by the Applanix POS L with integrated odometry, IMU and a GPS unit.

acquired by the 2D laser scanner also mounted on the vehicle. The quality of the resulting point cloud depends on several factors:

- The calibration of the entire system, i.e., the accuracy to which the position and orientation of each individual sensor in relation to the vehicle has been determined.
- The accuracy of the external positioning sensors, i.e., the GNSS, IMU and odometry.
- The availability of GNSS, as it may suffer temporary blackouts under bridges, in tunnels and urban canyons.
- The accuracy of the laser scanner itself.

The non-availability of the GNSS in constricted spaces is an exceptionally limiting factor for mobile scanning. Without the requirement for a GNSS device, applications such as construction or maintenance of tunnels and mines and facility management would be open to the use of mobile scanning systems.

In this paper we propose a novel algorithm for the calibration of a mobile scanning system that estimates the calibration parameters for *all* system components simultaneously without relying on additional hardware. We also deal with the problem of erroneous positioning of the vehicle in a novel fashion. We present an algorithm for computing a corrected trajectory that produces point clouds that are a more precise representation of the measured environment. To this purpose we constructed a mobile platform with a modified approach to mobile laser scanning. Instead of a 2D laser scanner common for mobile scanning systems, we employ a 3D laser scanner as depicted in Fig. 1 (top and bottom right). The laser scanner obtains 3D data by rotating around its vertical axis during the scanning process (see http://youtu.be/kIBReVytbJw) at 6 s per rotation. This design retains the high degree of automation and speed of common mobile laser scanning systems. Furthermore, only a single laser range sensor is required to produce models with minimal data shadowing. The additional rotation with respect to the platform during the scanning process provides increased observability of positioning errors with respect to 2D mobile laser scanning systems. We evaluate the algorithms on data sets acquired by the mobile platform Irma3D and an automobile as well as on data acquired by a commercial mobile laser scanning system (see Fig. 1.)

2. State of the Art

2.1. Calibration

In order for the mobile laser scanner to acquire high quality 3D models the position and orientation of every individual sensors must be known. This paper is concerned with algorithmic calibration of these systems, i.e., algorithms to establish the parameters that best describe sensor displacements based on the sensor data itself. In this process calibration parameters that are only roughly measured with external instruments are fine-tuned automatically.

The most basic sensor of the vehicle is its odometry, which estimates the 3D pose, i.e., the 2D position and 1D orientation of the vehicle by counting and extrapolating the wheel rotations. Martinelli

et al. present a method to calibrate the odometry readings using an augmented Kalman filter [1]. Their algorithm estimates the odometry parameters on the basis of pose information acquired by the Simultaneous Localization and Mapping (SLAM) system. As odometry is the least accurate estimator of a vehicles pose, mobile laser scanners are usually also equipped with more exact positioning sensors, e.g., an IMU or a GNSS device. Traditionally, these are calibrated against other positioning devices whose pose in relation to the vehicle is already known [2]. In [3] Nebot and Durrant-Whyte present a method for calibrating a low cost six degrees-of-freedom IMU on a land vehicle. Initially the vehicle is at rest to estimate the gravitational and other biases of the accelerometers and gyros of the IMU. Then the redundancies in the measurements are exploited to estimate the calibration parameters of the IMU during a test drive.

Finally, the position and orientation of the laser measurement device also requires calibration. This is often done using a process called boresight alignment. Boresight calibration is the technique of finding the rotational parameters of the range sensor with respect to the already calibrated IMU/GNSS unit. Skaloud and Schaer describe a calibration method where an airplane equipped with a laser scanner flies several times over a residential area. Planes are extracted from the roofs in every flyover [4]. Then, the planes are matched against each other to minimize the calibration error. A similar method was developed by Rieger et al. for non-airborne kinematic laser scanning [5]. Here the vehicle drives past the same house several times. Again the planar surfaces of the buildings are exploited to estimate the calibration parameters of the laser scanner. The calibration parameters of the laser scanner can also be estimated when the vehicle itself is stationary. Talaya et al. present a calibration method for estimating the boresight parameters of a laser scanner by registering several scans of the environment at different positions [6]. The position and orientation of the vehicle are known at any one point and the scans are registered using landmarks. Recently Underwood et al. presented an approach for calibrating several range scanners to each other with no information about the pose of the vehicle [7]. The vehicle scans a previously known environment from several positions. The range data is then manually labeled, so that the ground truth for each data point is known and an error metric can be constructed. Minimizing the error yields optimal calibration parameters for the range sensors.

Our calibration system is similar to that in [7] in that multiple components are calibrated using a quality metric of the reconstructed point cloud. However, our approach also calibrates the odometry and we require no manual selection of points or any special environment for the calibration. Instead, we employ a quality metric that is similar to the one used by [8].They calibrate a terrestrial 3D laser scanner of their own design by computing the minimum of the quality metric with respect to the internal calibration parameters. We also improve upon the metric by reducing the time complexity that is involved in its evaluation. Furthermore, we estimate the calibration parameters for multiple sensors simultaneously. This allows us to forego a separate calibration process for every subsystem.

2.2. Simultaneous Localization and Mapping

Aside from sensor misalignment a second source of errors are timing related issues. On a mobile platform all subsystems need to be synchronized to a common time frame. This can be achieved with pure hardware via triggering or with a mix of hard and software like pulse per second (PPS) or the network time protocol [9]. Good online synchronization is not always available for all sensors. Olson developed a solution for the synchronization of clocks that can be applied after data acquisition [10].

An even more significant source of errors is the incorrect positioning of the vehicle. Solving this problem requires approaches other than classical rigid SLAM algorithms. An area that may provide a solution is the area of non-rigid registration, which is largely unexplored except in the medical imaging community where it is widespread due to the need to align data with multiple modalities [11,12].

Williams et al. describe an extension of a rigid registration algorithm that includes point estimation to compensate for noisy sensor data [13]. This technically constitutes a non-rigid registration algorithm designed for low scale high frequency deformations. Similarly, Pitzer and Stiller provide a probabilistic mapping algorithm for 2D range scans, where also optimized point measurements are estimated [14].

Chui and Rangarajan proposed a point matching algorithm that is capable of aligning point clouds with each other [15]. These approaches are usually time expensive due to the enlarged state space. Aiming to reduce noise Unnikrishnan and Hebert locally fit high-order polynomials to 3D data [16]. However, large scale deformations will not be corrected by such a local approach. Mitra et al. rely on a high number of scans which contain a slowly deforming object [17]. Instead of computing point correspondences they estimate the deformation by the local space-time neighborhood. Consequently, higher point density is not a difficulty but a benefit for the algorithm. On the other hand the algorithm is incapable of dealing with large deformations and movements between individual scans.

Brown and Rusinkiewicz developed a global non-rigid registration procedure [18]. They introduced a novel variant of the Iterative Closest Point (ICP) algorithm to find very accurate correspondences. The original ICP algorithm [19] aligns laser scans acquired at different positions based on the distances between points from the two point clouds. Even though the registration requires extreme subsampling the deformation can be successfully generalized to the entire scan. Unfortunately, this technique is not fully applicable to laser scans acquired by mobile robots [20].

Stoyanov and Lilienthal present a non rigid optimization for a mobile laser scanning system [21]. They optimize point cloud quality by matching the beginning and the end of a single scanner rotation using ICP. The estimate of the 3D pose difference between the two points in time is then used to optimize the robot trajectory in between. A similar approach Bosse a uses a modified ICP algorithm with a custom correspondence search to optimize the pose of six discrete points in time of the trajectory of a robot during a single scanner rotation [22]. The trajectory in between is modified by distributing the errors with a cubic spline. This approach is extended to the Zebedee, a spring mounted 2D laser scanner in combination with an IMU [23]. Their algorithm sequentially registers a short window of 2D slices onto the previously estimated 3D point cloud by surface correspondences.

The approach presented in this paper optimizes the point cloud using full 6D poses and is not restricted to a single scanner rotation or time window. Instead we improve scan quality globally in all 6 degrees

of freedom for the entire trajectory. Finally, a rotating 3D scanner is not necessary for the algorithm to improve scan quality. This is demonstrated on state of the art mobile laser scanners.

3. Calibration

Calibration is the process of estimating the parameters of a system. For a sensor system this means that the closer these values are to the true physical quantities, the more accurate the final measurements of the entire system will be. We have designed a calibration process for mobile laser scanners that is capable of estimating all those quantities that influence the internal precision of the acquired point cloud. We assume some rough estimate is available for each parameter. The first step is to acquire a data set with the mobile scanning system. Although it is not necessary for the geometry of the environment to be known, the system should exhibit linear motion, left and right turns and ideally even pitching and yawing motions. A specially designed measure of the internal precision of the point cloud is then used to find the calibration parameters via Powell's method.

In the context of mobile laser scanners, there are several types of parameters of note. First, the geometrical alignment of each subsystem with respect to the vehicle. There are many frames of reference on a mobile platform. The challenge of establishing the transformation between the vehicle and the global reference system is subject to the measurements of the positioning systems and is discussed in the next chapter. For proper data acquisition, the full six degrees of freedom (DOF) pose $\mathbf{W}_s = (t_{x,s}, t_{y,s}, t_{z,s}, \theta_{x,s}, \theta_{y,s}, \theta_{z,s})$ of each sensor s with respect to the vehicle frame is essential. Here $t_{x,s}, t_{y,s}$ and $t_{z,s}$ are the positional parameters specifying the translation along the x, y and z axis whereas the $\theta_{x,s}, \theta_{y,s}$ and $\theta_{z,s}$ parameters are Euler angles specifying the orientation around the respective coordinate axes of sensor s. Incorrect geometrical calibration leads to incorrect trajectory estimation and systematic errors in the final point cloud.

Aside from the general alignment parameters, there are internal parameters that are specific to certain sensors. These quantities directly influence the measurement accuracy of the respective sensor and could in principle be calibrated independently of the rest of the system. An example of this can be found in [24]. The laser measurements are dependent on the internal alignment of the emitter and the mirror geometry of the laser scanner whereas the odometry is dependent on wheel circumference w and axis length a of the vehicle. In practice, for many sensors the modification of the calibration parameters is unnecessary or infeasible, as the internal calibration is often performed by the manufacturers themselves.

Finally, systematic timing errors due to latencies can be counteracted by offset parameters o_s. We assume all sensor measurements are timestamped. Time frames are synchronized by an offset that represents the minimal inherent delay between a measurement and its reception in the system. In principle, the proposed algorithm is capable of adjusting every calibration parameter including timing related parameters. However, systematical synchronization errors are minor for mobile laser scanning systems and do not contribute to the quality of the final point cloud in a significant way. Therefore, we exclude timing related parameters from the following formulation.

3.1. Algorithm description

The principle behind the approach is to find the parameters $\mathbf{C}$ that produce the most consistent point cloud possible. The calibration parameters for all sensors s are concatenated to construct the calibration vector

$$\mathbf{C} := (a, w, \mathbf{W}_0, o_0, \ldots, \mathbf{W}_n, o_n). \tag{1}$$

In the definition (1) a and w represent odometry parameters, while $\mathbf{W}_s$ represent the rigid transformation of sensor s to the vehicle coordinate system. Then the process of extracting the point cloud $P = \mathbf{p}_0, \ldots, \mathbf{p}_n$ consisting of n points with $\mathbf{p}_i = (x_i, y_i, z_i)$ from M, the entirety of measurements of every sensor, can be said to be a function $f(M, \mathbf{C}) = P$. To find the optimal calibration we must define an appropriate quality measure on P.

We employ a general quality measure that is similar to the one used by [8]. We model the points $\mathbf{p}_i$ as drawn from a probability distribution function (*pdf*) $d(\mathbf{l})$ which represent the probability that a specific location $\mathbf{l}$ has been measured. The pdf can be approximated as

$$d(\mathbf{l}) = \frac{1}{n} \sum_{j}^{n} G(\mathbf{l} - \mathbf{p}_j, \sigma^2 \mathbf{I}) \tag{2}$$

where $G(\mu, \sigma^2 \mathbf{I})$ is a Gaussian with mean μ and covariance $\sigma^2 \mathbf{I}$. A Gaussian distribution does not model errors such as those that are introduced by uncertainties in the trajectory or that are dependent on the range, incidence angle or reflectance of the surface that was measured. However, it is more than sufficient to capture the notion of point cloud consistency. Calibration errors lead to surfaces appearing at multiple positions in the point cloud. The entropy of $d(\mathbf{l})$ increases with these errors and decreases the more compact the point cloud is. Thus, an entropy measure on $d(\mathbf{l})$ is also a quality measure for P. [8] derive the following simplified entropy measure, which depends only on the pairwise distance of every possible pair of sample points:

$$E'(\mathbf{P}) = -\sum_{i}^{n} \sum_{j}^{n} G(\mathbf{p}_i - \mathbf{p}_j, 2\sigma^2 \mathbf{I}) \tag{3}$$

[8] use the Jacobian of E' with respect to the calibration parameters of their system to apply Newton's method for optimization. This is not possible in our case for several reasons. First, the calibration is supposed to be general, i.e., no definitive system to calibrate for is given. Second, the inclusion of parameters for the positioning systems makes the derivation of the Jacobian infeasible. This is due to the fact that in order to compute a global measurement $\mathbf{p}_i$ at time t_i the pose estimate $\bar{\mathbf{V}}_i$ of the vehicle must be known. However, to compute $\mathbf{W}_i$ all sensor measurements prior to t_i may be taken into account. Furthermore, the presence of multiple positioning sensors requires sensor fusion, thereby increasing the non-linearity and complexity of the entropy measure. In addition, we acquire a large number of sample points, usually in the order of several millions for properly calibrating an entire mobile platform. The quality measure $E'(P)$ is infeasible for the calibration using large point clouds. One way of dealing

with this problem is to reduce the number of points. We propose to uniformly subsample the entire point cloud. Furthermore, we propose to simplify the measure by using only a subsample of all pairs of points. For every point $\mathbf{p}_i$ that remains from the initial sub sampling we determine its closest point $\mathbf{q}_i \in P$ such that $|t_i - t_j| > \delta$. Here, δ is the minimal amount of time that must have elapsed for the laser scanner to have measured the same point on the surface again. Temporally close measurements are usually spatially close as well, so they must be excluded to prevent them from dominating the quality measure. δ is easily derived from $\delta = \frac{2\pi}{v_s + v_r}$, where v_s is the rotational speed of the laser scanner and v_r the maximal rotational speed of the vehicle. At most n pairs of closest points are thus used in the evaluation of the error metric instead of n^2. We seek to find

$$\hat{\mathbf{C}} = \underset{\mathbf{C}}{\operatorname{argmax}}\, E(f(M, \mathbf{C})) \tag{4}$$

where

$$E(f(M, \mathbf{C})) = -\sum_{i}^{n} G(\mathbf{p}_i - \mathbf{q}_i, 2\sigma^2 \mathbf{I}). \tag{5}$$

Standard minimization algorithms compute the derivative of the error function with respect to the calibration parameters. This is infeasible for the function $E(f(M, \mathbf{C}))$ as it involves complex algorithms for filtering data and fusing multiple modalities of measurements. We employ Powell's method for optimizing E as it does not require the derivative [25]. Instead, we must merely provide an initial estimate of $\mathbf{C}$, which is readily obtainable by manual estimation.

3.2. Strategies for increasing efficiency

To deal with the massive amount of data in a reasonable time, we employ several strategies. First, we uniformly and randomly reduce the point cloud by using only a constant number of points per volume, typically to 1 point per 3 cm^3. An octree data structure is ideally suited for this type of subsampling. An additional advantage of the subsampling uniformity is that surfaces closer to the scanner do not unfairly influence the quality measure more than surfaces that are farther away. We also employ an octree data structure as a fast and compact representation of the entire point cloud. This allows us to compute the nearest neighbor for a point at comparable speed to state of the art k-d tree search libraries [26]. The octree also compresses the point cloud so it can be easily stored and processed.

We employ the nearest neighbor search algorithm as described in [27] that has been modified to enforce the time constraint. The modified algorithm is summarized in Algorithm 1.

Closest points are accepted only if $|\mathbf{p}_i - \mathbf{q}_i| \leq d_{\max}$. Although $d_{\max}$ is an input parameter that has to be manually set for the algorithm, it massively reduces the computation time needed to evaluate the quality measure. Furthermore, it is linked to the second input parameter σ. Point pairs with the maximal distance $d_{\max}$ should contribute to the quality metric with a fixed ratio r to that of the "perfect" point pair with zero distance, i.e. we set $\sigma = \sqrt{-\frac{{d_{\max}}^2}{2 \ln r}}$. In the experiments the ratio has always been set to $r = 0.01$.

Algorithm 1 FindClosest

Input: query point q, maximal allowed distance d

```
1: function FindClosest(q, d)
2:     lookup deepest node N containing bounding box of q
3:     convert q to octree coordinate i
4:     return FindClosestInNode(N, q, i, d, 0)
5: end function

1: function FindClosestInNode(N, q, i, d, p)          ▷ p is the currently closest point
2:     compute child index c from i
3:     for j = 0 to 7 do
4:         get next closest child C = N.children[sequence[c][j]]
5:         if C is outside of bounding ball then
6:             return currently closest point p
7:         else
8:             if C is a leaf then
9:                 FindClosestInLeaf(C, q, d, p)
10:            else
11:                FindClosestInNode(C, q, i, d, p)
12:            end if
13:        end if
14:    end for
15:    return currently closest point p
16: end function

1: function FindClosestInLeaf(C, q, d, p)
2:     T(q) is timestamp of q
3:     for all points o do
4:         T(o) is timestamp of point o
5:         if |T(q) − T(o)| > δ and |o − q| < d then
6:             p ← o
7:             d ← |o − q|
8:         end if
9:     end for
10:    return closest point p
11: end function
```

4. Semi-Rigid Optimization for SLAM

In addition to the calibration algorithm, we also developed an algorithm that improves the entire trajectory of the vehicle simultaneously. Unlike previous algorithms, e.g., by [21] and [22], it is not restricted to purely local improvements. We make no rigidity assumptions, except for the computation of the point correspondences. We require no explicit vehicle motion model, although such information may be incorporated at no additional cost. The algorithm requires no high-level feature computation, i.e., we require only the points themselves.

The movement of the mobile laser scanner between time t_0 and t_n creates a trajectory $T = \{\mathbf{V}_0, \ldots, \mathbf{V}_n\}$, where $\mathbf{V}_i = (t_{x,i}, t_{y,i}, t_{z,i}, \theta_{x,i}, \theta_{y,i}, \theta_{z,i})$ is the 6DOF pose of the vehicle at time t_i with $t_0 \leq t_i \leq t_n$. Using the trajectory of the vehicle a 3D representation of the environment can be obtained by "unwinding" the laser measurements M to create the final map P. However, sensor

errors in odometry, IMU and GNSS as well as systematic calibration errors and the accumulation of pose errors during temporary GNSS outages degrade the accuracy of the trajectory and therefore the point cloud quality.

The current state of the art developed by [22] for improving overall map quality of mobile mappers in the robotics community is to coarsely discretize the time. This results in a partition of the trajectory into subscans that are treated rigidly. Then rigid registration algorithms like the ICP and other solutions to the SLAM problem are employed. Obviously, trajectory errors within a subscan cannot be improved in this fashion. Applying rigid pose estimation to this non-rigid problem is also problematic since rigid transformations can only approximate the underlying ground truth. Consequently, overall map quality suffers as a result.

We employ a much finer discretization of the time, at the level of individual 2D scan slices or individual points. This results in the set of measurements $M = \{\mathbf{m}_0, \ldots, \mathbf{m}_n\}$ where $\mathbf{m}_i = (m_{x,i}, m_{y,i}, m_{z,i})$ is a point acquired at time t_i in the local coordinate system of $\mathbf{V}_i$. In case $\mathbf{V}_i$ represents more than a single point, $\mathbf{V}_i$ is the local coordinate of the first point. All represented points are motion compensated with the best known interpolated trajectory. As modern laser scanners typically acquire 2D scan slices at a frequency of 100–200 Hz time is discretized to 5–10 ms. Applying the pose transformation $\oplus$ we derive the point $\mathbf{p}_i = \mathbf{V}_i \oplus \mathbf{m}_i = \mathbf{R}_{\theta_{x,i},\theta_{y,i},\theta_{z,i}}\mathbf{m}_i + (t_{x,s}, t_{y,s}, t_{z,s})^T$ in the global coordinate frame and thereby also the map $P = \{\mathbf{p}_0, \ldots, \mathbf{p}_n\}$. Here, $\mathbf{R}_{\theta_{x,i},\theta_{y,i},\theta_{z,i}}$ is the rotation matrix that is defined by:

$$\begin{pmatrix} \cos\theta_y \cos\theta_{z,i} & -\cos\theta_{y,i}\sin\theta_{z,i} & \sin\theta_{y,i} \\ \cos\theta_{z,i}\sin\theta_{x,i}\sin\theta_{y,i} + \cos\theta_{x,i}\sin\theta_{z,i} & \cos\theta_{x,i}\cos\theta_{z,i} - \sin\theta_{x,i}\sin\theta_{y,i}\sin\theta_{z,i} & -\cos\theta_{y,i}\sin\theta_{x,i} \\ \sin\theta_{x,i}\sin\theta_{z,i} - \cos\theta_{x,i}\cos\theta_{z,i}\sin\theta_{y,i} & \cos\theta_{z,i}\sin\theta_{x,i} + \cos\theta_{x,i}\sin\theta_{y,i}\sin\theta_{z,i} & \cos\theta_{x,i}\cos\theta_{y,i} \end{pmatrix} \tag{6}$$

Given M and T we find a new trajectory $T' = \{\mathbf{V}'_1, \ldots, \mathbf{V}'_n\}$ with modified poses so that P generated via T' more closely resembles the real environment.

The complete semi-rigid registration algorithm proceeds as follows. Given a trajectory estimate, we compute the point cloud P in the global coordinate system and use nearest neighbor search to establish correspondences. Then, after computing the estimates of pose differences and their respective covariances we optimize the trajectory T. This process is iterated until convergence, i.e., until the change in the trajectory falls below a threshold.

To deal with the massive amount of data in reasonable time, we employ the same strategies as for the calibration. Using an octree data structure we uniformly and randomly reduce the point cloud by using a constant number of points per volume, typically to 1 point per 3 cm^3. Furthermore, in initial stages of the algorithm, estimates $\bar{\mathbf{V}}_{i,j}$ are only computed for a subset of poses $\mathbf{V}_0, \mathbf{V}_k, \mathbf{V}_{2k}, \ldots$, with k in the order of hundreds of milliseconds. In every iteration k is decreased so that the trajectory is optimized with a finer scale.

4.1. Pose Estimation

Our algorithm incorporates pose estimates from many sources, such as odometry, IMU, and GNSS. For this paper we use the formulation of pose estimates $\bar{\mathbf{V}}_{i,i+1}$ that are equivalent to pose differences:

$$\bar{\mathbf{V}}_{i,i+1} = \bar{\mathbf{V}}_i \ominus \bar{\mathbf{V}}_j. \tag{7}$$

The operator $\ominus$ is the inverse of the pose compounding operator $\oplus$, such that $\mathbf{V}_j = \mathbf{V}_i \oplus (\mathbf{V}_i \ominus \mathbf{V}_j)$. In addition to the default pose estimates that may also be enhanced by separating all pose sensors into their own estimates with proper covariances, we estimate differences between poses via the point cloud P.

For each measurement $\mathbf{p}_i$, we find a closest measurement $\mathbf{p}_j$ via nearest neighbor search with $|t_i - t_j| > \delta$, where δ is again the minimal amount of time that must have elapsed for the laser scanner to have measured the same point again. Points are stored in the global coordinate frame as defined by the estimated trajectory T. The positional error of two poses $\mathbf{V}_i$ and $\mathbf{V}_j$ is then given by

$$E_{i,j} = \sum_{k=i-N}^{i+N} \left\| \mathbf{V}_i \oplus \mathbf{m}_k - \mathbf{V}_j \oplus \mathbf{m}'_k \right\|^2 = \sum_{i=1}^{m} \left\| \mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j) \right\|^2. \tag{8}$$

Here, the $\mathbf{m}_k, \mathbf{m}'_k$ is the pair of closest points in their respective local coordinate system, and N defines a small neighborhood of points taken in the order of hundreds of milliseconds that is assumed to have negligible pose error. Since we will require a linear approximation of this function and $\oplus$ is decidedly non-linear we apply a Taylor expansion of $\mathbf{Z}_k$:

$$\mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j) \approx \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) - \left(\nabla_{\mathbf{V}_i} \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) \Delta \mathbf{V}_i - \nabla_{\mathbf{V}_j} \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) \Delta \mathbf{V}_j \right).$$

Here, $\nabla_{\mathbf{V}_i}$ refers to the derivative with respect to $\mathbf{V}_i$. We create a matrix decomposition $\nabla_{\mathbf{V}_i} \mathbf{Z}_k = \mathbf{M}_k \mathbf{H}$ such that the matrix $\mathbf{M}_k$ is independent of $\mathbf{V}_i$. Then, $\mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j)$ is approximated as:

$$\mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j) \approx \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) - \mathbf{M}_k(\mathbf{H}_i \Delta \mathbf{V}_i - \mathbf{H}_j \Delta \mathbf{V}_j).$$

The minimum of this new linearized formulation of the error metric and its covariance is given by

$$\begin{aligned} \bar{\mathbf{E}}_{i,j} &= (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{Z} \\ \mathbf{C}_{i,j} &= s^2 (\mathbf{M}^T \mathbf{M}). \end{aligned}$$

Here $\mathbf{Z}$ is the concatenated vector consisting of all $\mathbf{Z}_k = \bar{\mathbf{V}}_i \oplus \mathbf{m}_k - \bar{\mathbf{V}}_j \oplus \mathbf{m}'_k$. s is the unbiased estimate of the covariance of the independent and identically distributed errors of the error metric and is computed as

$$s^2 = (\mathbf{Z} - \mathbf{M}\bar{\mathbf{V}}_{i,j})^T (\mathbf{Z} - \mathbf{M}\bar{\mathbf{V}}_{i,j}) / (2M - 3). \tag{9}$$

The matrix decomposition $\mathbf{M}_k\mathbf{H}$ depends on the parametrization of the pose compounding operation $\oplus$. Since we employ the Euler parametrization as given in Eq. 6 the matrices $\mathbf{M}_k$ and $\mathbf{H}_i$ are given by:

$$\mathbf{M}_k = \begin{pmatrix} 1 & 0 & 0 & 0 & -m_{y,k} & -m_{z,k} \\ 0 & 1 & 0 & m_{z,k} & m_{x,k} & 0 \\ 0 & 0 & 1 & -m_{y,k} & 0 & m_{x,k} \end{pmatrix} \tag{10}$$

$$\mathbf{H}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & \bar{t}_{z,i}\cos(\bar{\theta}_{x,i}) + \bar{t}_{y,i}\sin(\bar{\theta}_{x,i}) & \bar{t}_{y,i}\cos(\bar{\theta}_{x,i})\cos(\bar{\theta}_{y,i}) - \bar{t}_{z,i}\cos(\bar{\theta}_{y,i})\sin(\bar{\theta}_{x,i}) \\ 0 & 1 & 0 & -\bar{t}_{z,i} & -\bar{t}_{x,i}\sin(\bar{\theta}_{x,i}) & -\bar{t}_{x,i}\cos(\bar{\theta}_{x,i})\cos(\bar{\theta}_{y,i}) - \bar{t}_{z,i}\sin(\bar{\theta}_{y,i}) \\ 0 & 0 & 1 & \bar{t}_{y,i} & -\bar{t}_{x,i}\cos(\bar{\theta}_{x,i}) & \bar{t}_{x,i}\cos(\bar{\theta}_{y,i})\sin(\bar{\theta}_{x,i}) + \bar{t}_{y,i}\sin(\bar{\theta}_{y,i}) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_{y,i}) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_{x,i}) & \cos(\bar{\theta}_{x,i})\cos(\bar{\theta}_{y,i}) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_{x,i}) & -\cos(\bar{\theta}_{y,i})\sin(\bar{\theta}_{x,i}) \end{pmatrix}. \tag{11}$$

4.2. Pose Optimization

We maximize the likelihood of all pose estimates $\mathbf{V}_{i,j}$ and their respective covariances $\mathbf{C}_{i,j}$ via the Mahalanobis distance

$$W = \sum_i \sum_j (\bar{\mathbf{V}}_{i,j} - (\mathbf{V}'_i - \mathbf{V}'_j))\mathbf{C}^{-1}_{i,j}(\bar{\mathbf{V}}_{i,j} - (\mathbf{V}'_i - \mathbf{V}'_j)),$$

or, with the incidence matrix $\mathbf{H}$ in matrix notation:

$$W = (\bar{\mathbf{V}} - \mathbf{H}\mathbf{V})^T\mathbf{C}^{-1}(\bar{\mathbf{V}} - \mathbf{H}\mathbf{V}). \tag{12}$$

This linear formulation of the problem has been made possible by the linearization of the pose difference equation in the previous section. The minimization of W is accomplished via solving the following linear equation system:

$$(\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H})\mathbf{V} = \mathbf{H}^T\mathbf{C}^{-1}\bar{\mathbf{V}}. \tag{13}$$

Computing the optimized trajectory is then reduced to inverting a positive definite matrix [28]. Since a significant portion of correspondences i, j are not considered due to the lack of point correspondences the matrix is sparse so that we make use of sparse Cholesky decomposition, [29]. A consequence of this formulation is that an arbitrary pose $\mathbf{V}_i$ must remain fixed, otherwise Eq. 13 would be under determined. This fixed pose also incidentally defines the global coordinate system. Thus, for the algorithm the optimization problem is decoupled from determination of global frame coordinates of the point cloud.

Table 1. Overview of data sets used in the experiments

Environment	Sensors	Data	Experiment
Basement 1	Irma3D - Riegl VZ-400 laser scanner, Xsens MTi IMU, odometry of Volksbot RT3	4 short runs; 0.1 m/s; 3 to 7 million points each	evaluation of calibration procedure
Basement 2	Irma3D - Riegl VZ-400 laser scanner, Xsens MTi IMU, odometry of Volksbot RT3	1 short run; 0.2 m/2; 3 to 7 million points each	quantitative evaluation of semi-rigid optimization
Campus	VW Touran - Riegl VZ-400 laser scanner, Xsens MTi IMU, OBD device	500 m; 19 km/h; 12 million points	qualitative evaluation of semi-rigid optimization with very bad input data
Salzburg	Riegl - Riegl VZ-400, GPS RTK		qualitative evaluation of semi-rigid optimization algorithm
Dortmund 1	Optech Lynx Mobile Mapper; twin 2D laser scanners; Applanix POS L with integrated odometry, IMU and GNSS	circle around a park; unbroken trajectory; 150 m	qualitative evaluation of semi-rigid optimization
Dortmund 2	Optech Lynx Mobile Mapper; twin 2D laser scanners; Applanix POS L with integrated odometry, IMU and GNSS	two runs along the same road; 400 m	extensive quantitative analysis of semi-rigid optimization
Vienna	Riegl - Riegl VMX-450; Applanix POS L with integrated odometry, IMU and GNSS	two runs around a street corner; 300 m	comparison of calibration to state of the art method; qualitative evaluation of semi-rigid optimization

5. Experiments

To evaluate the newly proposed system for improving the scan quality we have acquired a variety of data using multiple mobile laser scanners. An overview of the data sets is given in Table 1. The first scanner is the autonomous robot platform Irma3D as seen in Fig. 2. The main sensor of Irma3D is a Riegl VZ-400 laser scanner. Irma3D is built with a Volksbot RT-3 chassis and uses the Xsens MTi IMU as well as odometry to estimate its position. Several data sets using this robot were acquired in an empty basement room (see Fig. 2). All data sets were acquired in continuous mode, i.e., the laser scanner rotates around its vertical axis while the robot moves simultaneously. The robot moved in "serpentine" trajectories, i.e., taking left and right turns as well as segments where the heading remains unchanged. In this paper we present five data sets recorded by Irma3D with three to seven million points each. Four of those data sets were acquired to demonstrate the calibration procedure as presented in this paper. For these data sets, the robot drove in a a relatively slow manner (approx. 0.1 m/s), to ensure minimal wheel slippage. The fifth data set is used in the evaluation of the optimization algorithm. In this case, the robot was deliberately moved faster (approx. 0.2 m/s) to produce errors in the pose estimation. For all five data sets the scanner needed 6 s per rotation. 2D scans with a resolution of 0.1° were acquired at a rate of 120 Hz, which equates to a vertical resolution of 0.5° for the 3D point cloud.

The second mobile scanner uses the same sensors and scanning principle. However, as seen in Fig. 3 the platform used was a VW Touran automobile. The vehicle is not equipped with GNSS or accurate odometry. The current forward velocity is acquired with an accuracy of only 1 km/h via an on-board

Figure 2. Left: The robot Irma3D in the room where the experiments were carried out. Right: The model of the basement (colored by reflectance) as acquired by standard terrestrial laser scanning.

Figure 3. *Campus* data set acquired at Jacobs University Bremen. Left: The mobile scanner that was used to acquire the campus outdoor data set. Top right: Digital orthophoto of the campus that has been mapped by the mobile scanner. The red line approximates the trajectory driven by the vehicle. Image courtesy of AeroWest, DigitalGlobe, GeoBasis-DE/BKG, GeoContent, GeoEye and Google. Bottom right: For comparison purposes, the environment was also modeled using standard terrestrial laser scanning techniques.

Figure 4. Left: The TopScan GmbH mobile laser scanner using the Lynx Mobile Mapper by Optech was used to acquire two data sets in the city of Dortmund. Right: One smaller data set consists of the mobile scanner making a U-turn in a relatively confined space. The trajectory of this data set is indicated in red. A second larger data set along the length of an entire street has also been acquired. The vehicle traversed the road twice. The trajectories are depicted in green and yellow. Image courtesy of AeroWest, DigitalGlobe, GeoBasis DE/BKG, GeoContent, GeoEye and Google.

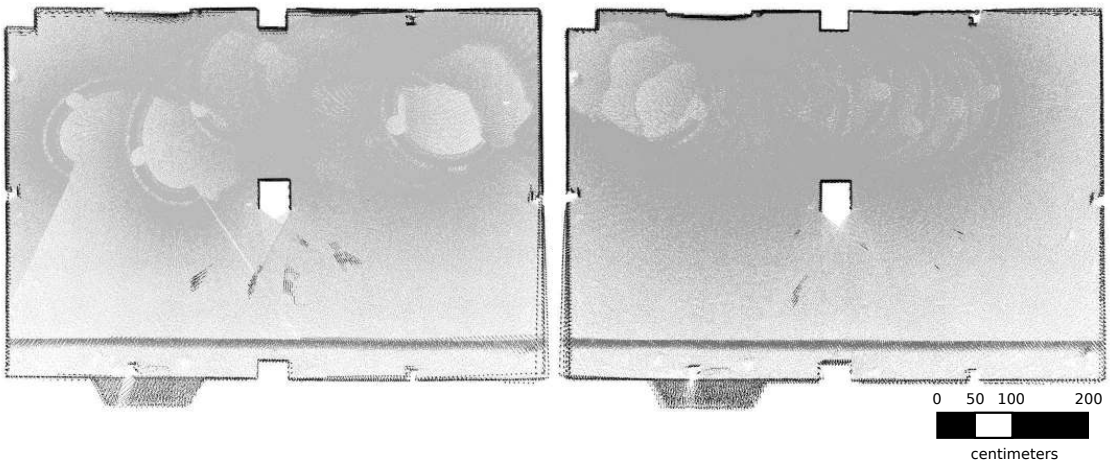
diagnostics (OBD) device that is directly connected to the entertainment CAN bus of the car. The vehicle moved alongside the campus buildings, made a U-turn, moved straight back on the other side of the road and made another U-turn to return to the origin. The car drove a distance of about 500 m and acquired a total of 12 million points with an average velocity of about 19 km/h. The scanner rotation and resolution settings were the same as in the Irma3D data sets. Due to the large inaccuracies in the initial data set, a quantitative ground truth comparison is not applicable. Nonetheless we acquired a ground truth data set of the campus using a terrestrial laser scanner at twelve positions to provide a point of reference.

We used data collected with an experimental platform constructed by RIEGL Laser Measurement Systems GmbH that is similar to the previous mobile scanners. This platform also employs the Riegl VZ-400 laser scanner in continuous mode. However, unlike the previous platforms the vehicle has no odometry at all. Since there is also no IMU installed, only a GPS RTK system was used for the estimation of the vehicle trajectory. This platform acquired a data set in Salzburg that we optimized using the semi-rigid matching algorithm.

In addition to these data sets, we evaluated the optimization algorithm on two data sets acquired by TopScan GmbH using the Optech Lynx Mobile Mapper (see Fig. 4). The Lynx mobile mapper employs twin 2D laser scanners and pose estimation by an Applanix POS L with integrated odometry, IMU and GNSS. The mobile mapping system traversed two streets in the city of Dortmund. In the first data set the vehicle encircled a small park in an unbroken trajectory with a length of about 150 m. For the second larger data set the vehicle traversed the same road twice. Both trajectories were stitched together on their eastern end to create a larger data set with a trajectory of a length of 400 m.

Lastly, we also show the suitability of the calibration algorithm as well as the optimization algorithm to the Riegl VMX-450. This system also uses an Applanix system for pose estimates with integrated odometry, IMU and GNSS. The data used for the experiments is a subset of a much larger data set and consists of a single street corner that was traversed twice by the mobile laser scanner with a trajectory

Figure 5. Topview of two point clouds acquired by the robot. The robot was calibrated with the proposed calibration algorithm. Although the scan quality is good, some non-calibration errors remain due to slipping wheels and erroneous position estimation.



of approximately 300 m length. However, these two strips are sufficient to perform calibration of the mobile laser scanner using our algorithm.

5.1. Calibration Results

The calibration algorithm is evaluated with an indoor data set where a quantitative analysis is carried out as well as on an outdoor data set where it is compared to a state of the art calibration.

5.1.1. Calibrating the Mobile Robot Irma3D

We test the calibration algorithm on 4 indoor data sets acquired by Irma3D. A manual estimation of the calibration parameters had been performed before the experiments. We applied the automatic calibration technique to each data set. A top view of two of the point clouds obtained with the automatically determined calibration parameters are shown in Fig. 5. To evaluate the quality of the resulting point clouds we compare them with a highly accurate model of the room as acquired by the Riegl VZ-400 laser scanner. The accuracy of the scanner is 5 mm [30] and thus the model should have a similar accuracy. Although the same scanner is used on the mobile robot, it is used in continuous mode, i.e., the laser scanner rotates around its vertical axis while the robot moves simultaneously. We compare the acquired point clouds with the model in the following fashion. After calibration, the entire mobile point cloud is matched to the model using ICP [19] from the 3D Toolkit (3DTK [31]). We compute point to plane distances on each of the 4 walls as well as the ceiling and floor of the room. The deviations for single scans from the point cloud are plotted as color coded images, i.e., green for absolute errors less than 1 cm, yellow to red for large positive errors and cyan to blue for large negative errors. The full color scale is given in Fig.6. White areas indicate that no point was measured at the corresponding location.

The results of the direct comparison between representative scans from each of the four runs after automatic and manual calibration and the model of the room are shown in Fig. 6. On the whole, the

Figure 6. Comparison of the acquired laser scans with the model using the manual (left) and the automatic calibration (right) for representative excerpts of the four *Basement 1* data sets acquired with Irma3D. Deviations in cm are color coded as indicated on the bottom. Best viewed in color.

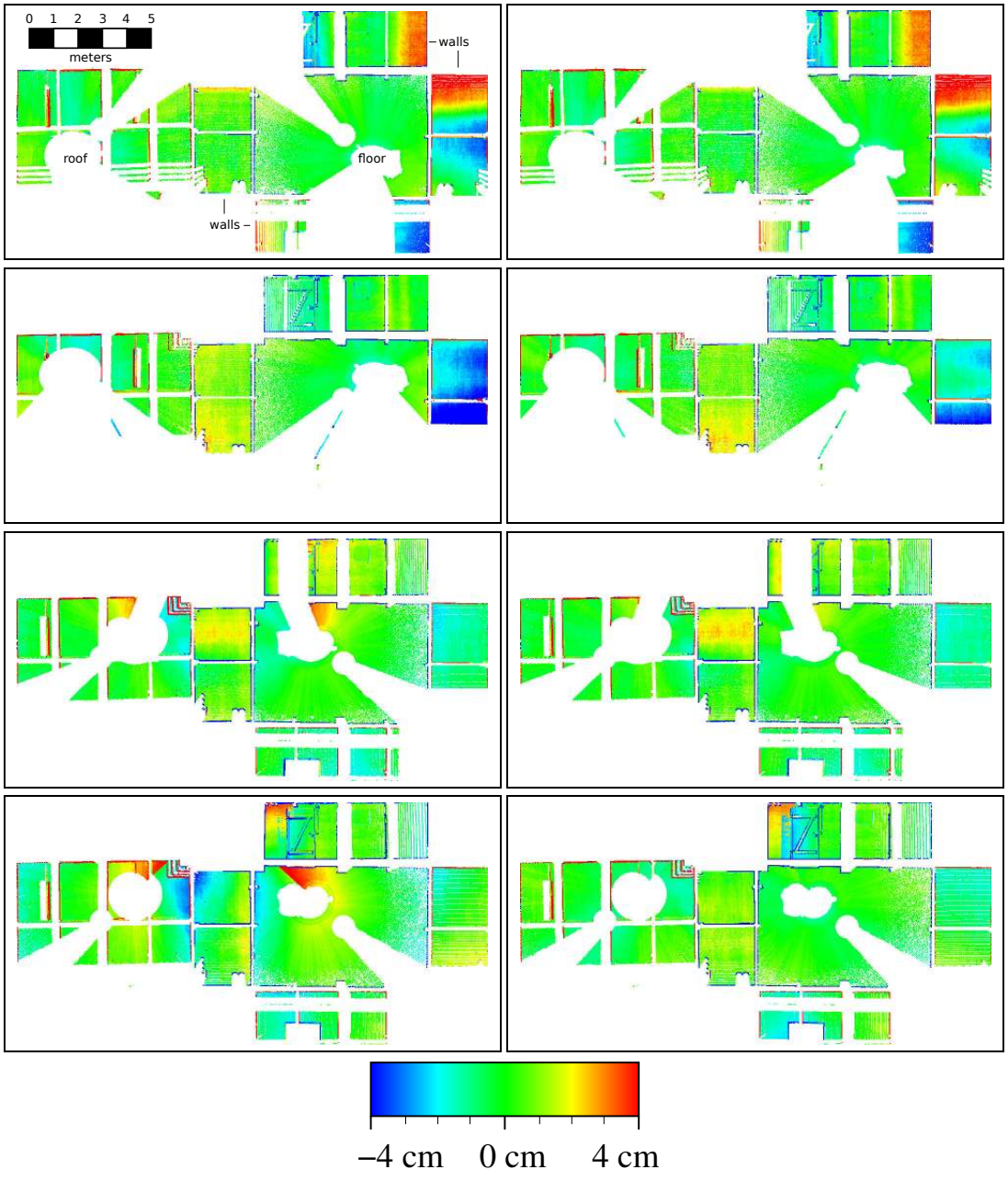
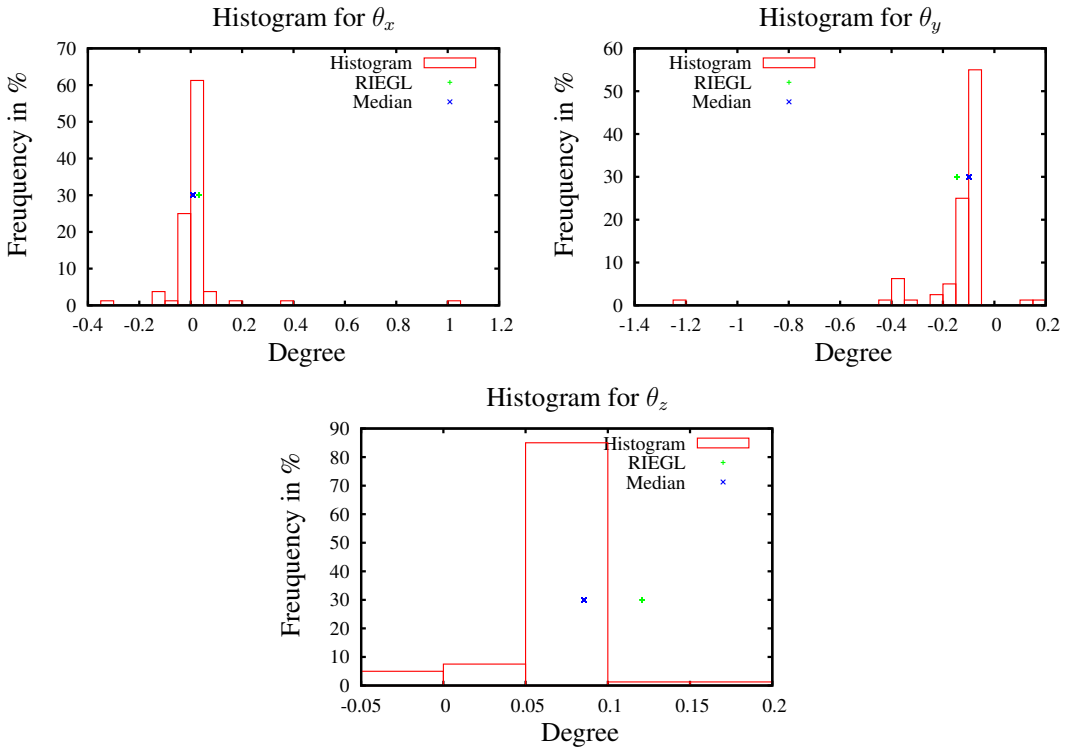


Figure 7. Histograms of the results of the boresight calibration. Bucket size is $0.05°$.



quality of the scans improves with the automatically determined parameters when compared to the manual estimation. Absolute errors are generally within 1-2 cm. Occasionally the deviations exceed that boundary. Very rarely they are above 3-4 cm. These large scale errors are explained by odometry related uncertainties in the estimation of the robot pose. Slipping wheels and shaking of the robot base are not detectable by any of the on-board sensors. Thus errors in the pose estimation carry over to erroneous point clouds. The combined mean error for all data sets was improved by the automatic calibration from 10.3 mm to 8.3 mm.

5.1.2. Riegl VMX-450

We also demonstrate the proposed calibration algorithm on an outdoor data set that was acquired using the Riegl VMX-450 in the city of Vienna. Fig. 8 (bottom right) shows the point cloud as it was calibrated by RIEGL Laser Measurement Systems using state of the art methods [5]. For this experiment we consider these calibration parameters as the ground truth. We applied the calibration algorithm a total of 100 times to this data set using ground truth parameters with additional random noise as the starting estimates. Noise was generated with a uniform distribution between $-10°$ and $10°$ for each of the euler angles that determine the orientation of the laser scanner. Other parameters like the wheel circumference of the vehicle could not be calibrated as the necessary raw measurements were unfortunately not available for this data set. The resulting calibration parameters are summarized in Fig. 7 and Table 2. Clearly,

Figure 8. Top view on intentionally perturbed *Vienna* dataset (top right), calibrated using the proposed algorithm (bottom left) and the original, as calibrated by RIEGL Laser Measurement Systems (bottom right) data set. An eerial view on the region is shown in the top left. Image courtesy of DigitalGlobe, European Space Imaging and Google.

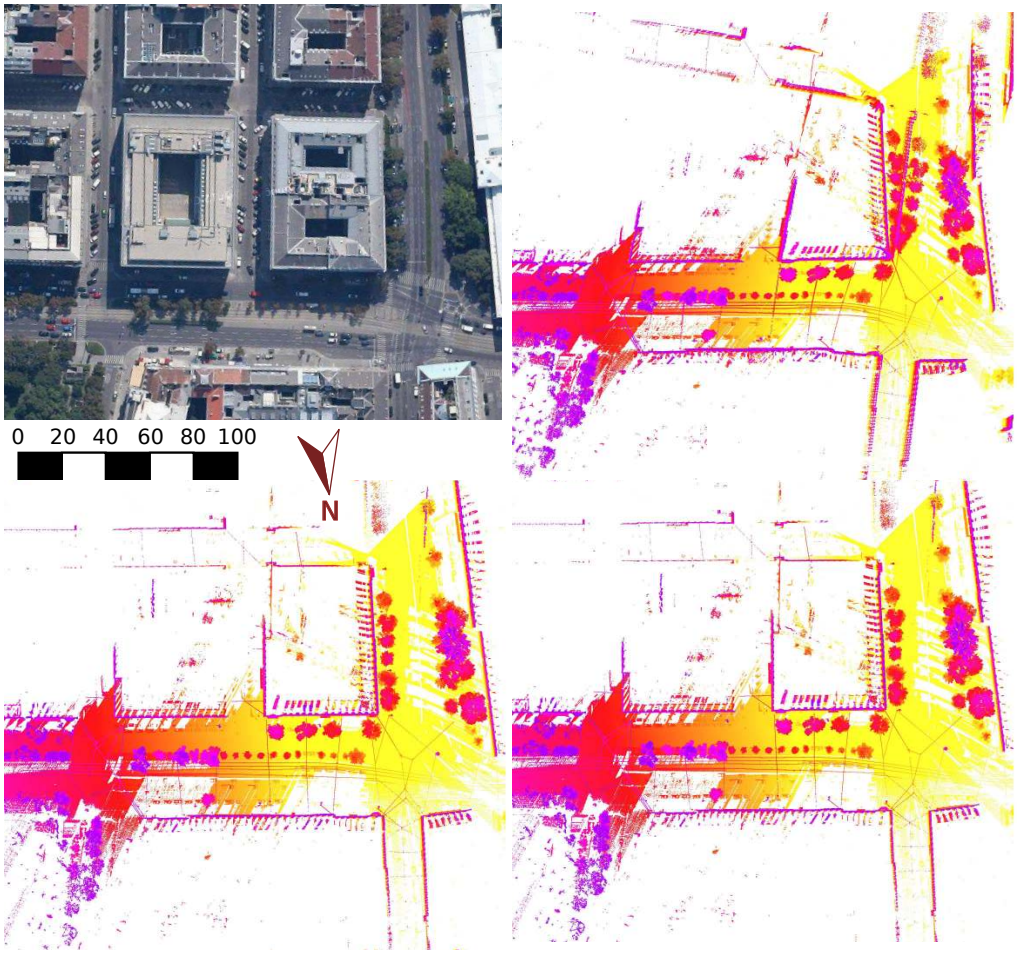


Figure 9. Close-ups on erroneous (top), calibrated using the proposed algorithm (middle) and the original, as calibrated by RIEGL Laser Measurement Systems (bottom), *Vienna* data set. Left: Corner of a building. Right: Top of church.

the proposed algorithm is capable of finding parameters very similar to those of the state of the art algorithm with a high degree of certainty. Fig. 8 and Fig. 9 show a representative starting estimate, the corresponding calibrated result as well as the original ground truth data set.

Table 2. Result of the outdoor boresight calibration experiment in degree.

	Riegl in-house calibration	calibration average	calibration median
θ_x	0.03286	0.01898	0.00878
θ_y	-0.14444	-0.13449	-0.09825
θ_z	0.12100	0.07695	0.08544

5.2. Optimization Results

We evaluate the algorithm using both indoor and outdoor data sets where ground truth is available and a quantitative analysis of the produced point clouds is possible. We also provide other outdoor data sets, where no quantitative analysis is possible. Nonetheless, qualitative analysis on a variety of data sets shows the suitability of the optimization algorithm.

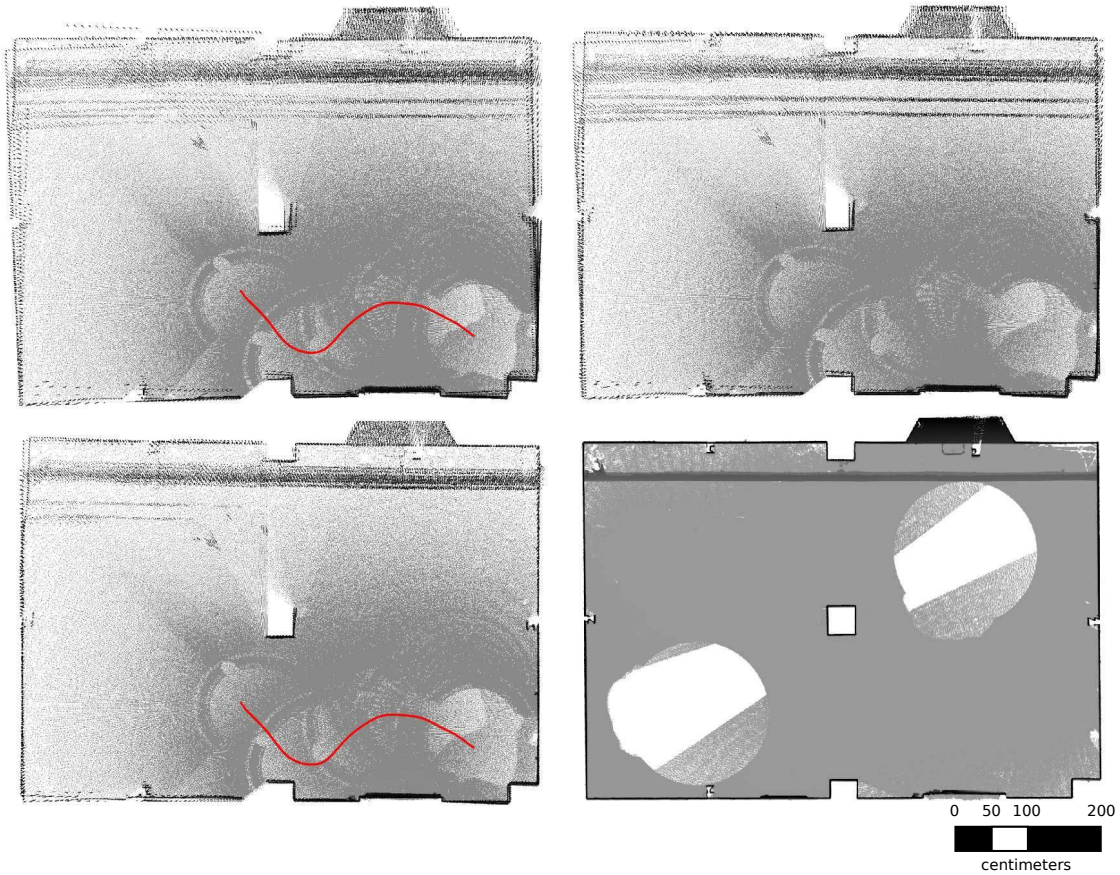
5.2.1. Trajectory Optimization of Spinning Scanners on Mobile Platforms

For a direct evaluation of the algorithm with comparison to ground truth data we acquired another data set (*Basement 2*) using the robot Irma3D in the empty basement room (see Fig. 2). Ground truth data of the room was obtained using static terrestial scanning with a Riegl VZ-400, with an estimated model accuracy of about 5 mm. To evaluate the quality of the resulting point cloud we compare it to the high accuracy ground truth model of the room in the same fashion as in section 5.1.1. .

A qualitative comparison of the results of the algorithm is presented in Fig. 10. The point clouds obtained with Irma3D along the full trajectory of data set *Basement 2* in the enclosed room are shown. The figure presents the initial point cloud (registered using robot IMU and odometry only), the rigid registration obtained with ICP, the semi-rigid registration computed by the novel algorithm and the ground truth data. The results of the direct comparison between the five individual point clouds, i.e., single rotations, from *Basement 2* before and after automatic semi-rigid registration and the model of the room are shown in Fig. 11. Again, the resulting images are color coded, i.e., green for absolute errors less than 1 cm, yellow to red for larger errors away from the observer and cyan to blue for larger errors towards the observer. White areas indicate that no point was measured at the corresponding location.

The proposed semi-rigid registration produces point clouds that more closely resemble the ground truth model. Although rigid registration procedures improve map quality to a degree, it is clear that the error within one scanner rotation is too large to allow for a good rigid match. The reduction of the inner error is even more obvious in the quantitative evaluation as seen in Fig. 11. The left side shows the point-to-model error of each of the individual scans before semi-rigid registration. The right column displays the error after the data set has been corrected. Apart from a small region in the top scan, all errors have been reduced significantly by the semi-rigid registration.

Figure 10. An overview of the *Basement 2* data set that is used for quantitative evaluation (cf. Fig. 6) of the semi-rigid registration. The red curve indicates the trajectory of the robot. Top: The initial data set with no registration (left) and with rigid registration via ICP and SLAM (right). Bottom left: The result of the novel semi rigid registration procedure. Bottom right: The model of the room acquired with an absolute accuracy of 5 mm.



In addition to the robots, we have data from two more mobile laser scanners that employ the VZ-400 in a continuously spinning mode. One is the automobile data set *Campus* acquired on the campus of the Jacobs University Bremen. The initial and the semi-rigidly corrected point cloud as well as the ground truth model of the campus are shown in Fig. 12. Due to the limited sensor capabilities of the mobile scanner, the initial trajectory is so erroneous that any attempts to improve upon the map quality by rigid registration methods failed. The semi-rigid registration shows a drastic improvement of the input data. The resulting point cloud exhibits no sign of error accumulation and reflects the ground truth quite well. It is remarkable that the algorithm also produces a straight trajectory when the only input is the first curved half of the trajectory. The scanner sweeps the same environment multiple times in 6 s intervals such that local improvements in the trajectory are possible. The algorithm is capable of computing a valid trajectory with an input that is erroneous all the time.

Figure 11. Comparison of the 5 acquired laser scans from *Basement 2* with the model using the initial (left) and the automatic semi-rigid registration (right). Deviations in cm are color coded as indicated in Fig. 6. Best viewed in color.

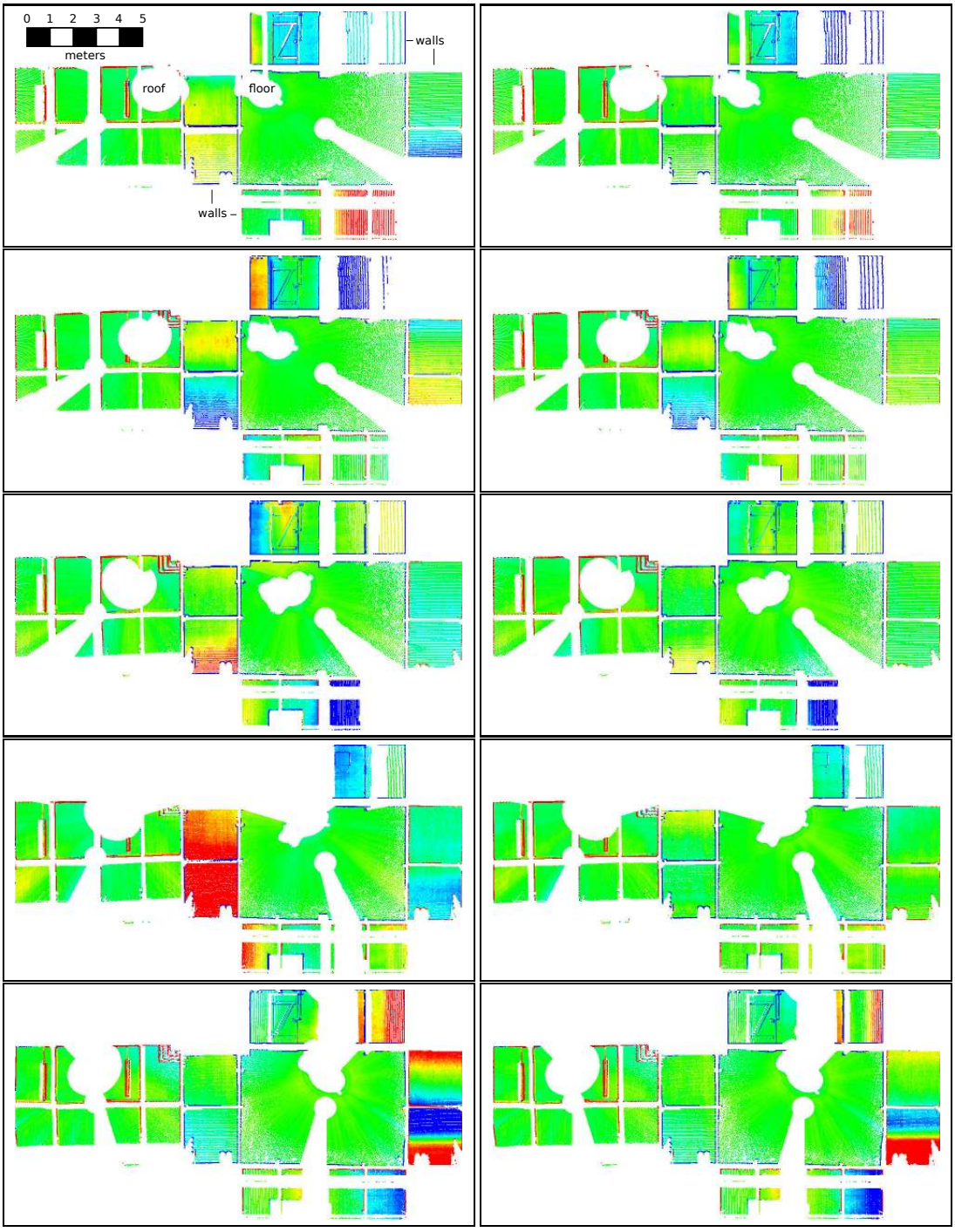
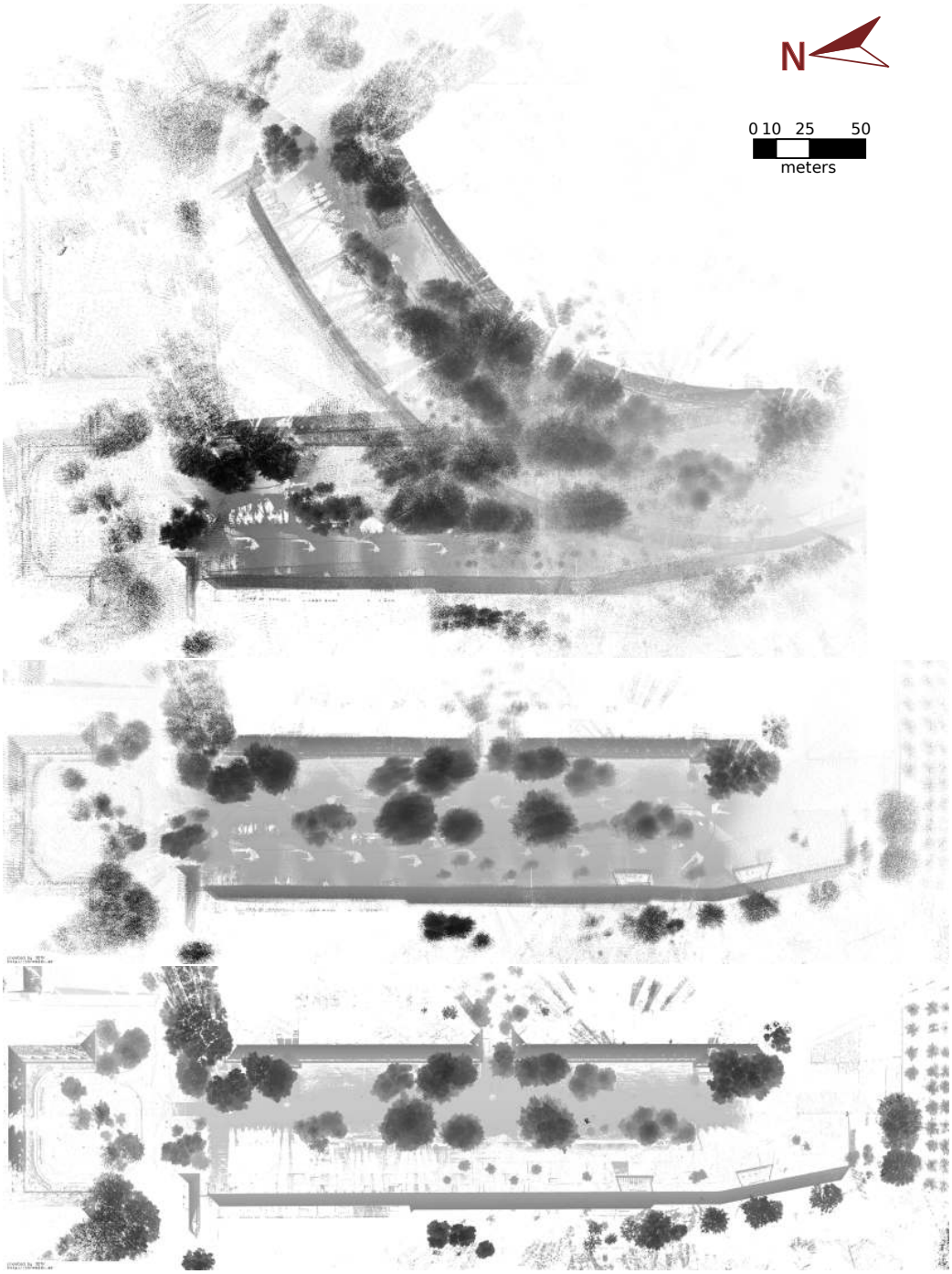


Figure 12. A comparison of the initial and optimized *Campus* point cloud acquired by the mobile laser scanner with the ground truth data acquired by a terrestrial laser scanner. Top: The initial point cloud of the car data set with about 12 million points. Middle: The result of the novel semi rigid registration algorithm. Bottom: The ground truth data set acquired with the Riegl VZ-400 mounted on a tripod at twelve positions.



Finally, the VZ-400 is also used on an experimental platform by the RIEGL Laser Measurement Systems GmbH equipped only with a GPS unit for trajectory estimation. An overview on the data set is given in Fig. 13. It is clear that the semi-rigid optimization algorithm improves on the quality of the point cloud. Fig. 14 gives a more detailed view of the scene. Although the final registration is not perfect, the errors in the initial point cloud are reduced. The semi-rigid registration does not further improve this data set due to being trapped in a local minimum. This is an inherent limitation of the proposed algorithm, similar to the behaviour of classical rigid registration using the ICP algorithm.

5.2.2. Lynx Mobile Mapper data

The optimization algorithm was applied to the Dortmund data sets acquired by TopScan GmbH using the Optech Lynx Mobile Mapper. These data sets differ significantly from previous ones in that the Lynx Mobile Mapper employs two 2D laser scanners and the 3D point cloud is created purely by the movement of the vehicle. Data sets like this are produced by current state of the art mobile scanning systems. Standard rigid optimizations generally do not apply, because there is only a small overlap between subsequent "subscans". For these data sets no information about the measurement certainty of the vehicle pose was given. The covariances $\mathbf{C}_{i,i+1}$ were set to a manually estimated diagonal matrix that reflect the estimated certainty of the pose estimation in general. However, the semi-rigid optimization proposed in this paper can deal with these challenging data sets as well.

We first evaluate the algorithm on the smaller loop around the small park (*Dortmund 1*). Here, no ground truth data is available. However in a direct comparison between the acquired data and the optimized data one can see many improvements in scan quality; see Fig. 15 for an overview. The measured scene was static for the obversational period, i.e., the parked cars did not move. The most noticeable improvements are in features like tree trunks and advertising columns which should have a circular shape in the orthogonal projection in Fig. 15. Comparing building facades before and after optimization also reveals an improvement in scan quality. The video supplement to this paper is an animated fly-through of the data set. It can be seen in high quality at http://youtu.be/L28C2YmUPWA.

For the *Dortmund 2* data set the mobile scanner traversed the same scene twice in a straight trajectory. This minimizes errors due to rotational uncertainties and is a more realistic use of the mobile scanning system. An overview and closeup of the data set before and after semi rigid optimization is given in Fig. 16. For this data set a quantitative analysis is available. Planes were extracted for the facades of the buildings. This was done for both of the two 2D scanners as well as both of the passes. Thus, every facade is represented up to four times in the data set. Comparing two plane parameters of the same facade yields an error vector with a direction and length and gives a measure of relative accuracy. With four planes there can be a total of six non-redundant comparisons. From these, we exclude the two comparisons between planes extracted from different sensors but in the same pass. Error vectors between these reflect only how good the system is calibrated. Compared to the other evaluations, the errors are very small and are not improved by the semi-rigid optimization. This analysis has been conducted by TopScan GmbH. The result of this analysis for the initial data set is given in Fig. 18. The average distance error between the facades is 10.6 cm. Averages and deviations for each of the four comparisons is given in Fig. 17. The same analysis was carried out on the optimized data set. The results are given in Fig. 19. From a direct comparison of the results it is clear that the optimization succeeded in improving the quality

Figure 13. The *Salzburg* data set provided by RIEGL Laser Measurement Systems GmbH. Top: Aerial view on the region. Image courtesy of Aerodata International Surveys, Geoimage Austria and Google. Middle: The initial point cloud, with the trajectory estimated purely by an GPS RTK system on-board the experimental mobile laser scanner. The points are colored by their timestamps to provide contrast. Bottom: The optimized point cloud after processing with the semi-rigid registration algorithm.

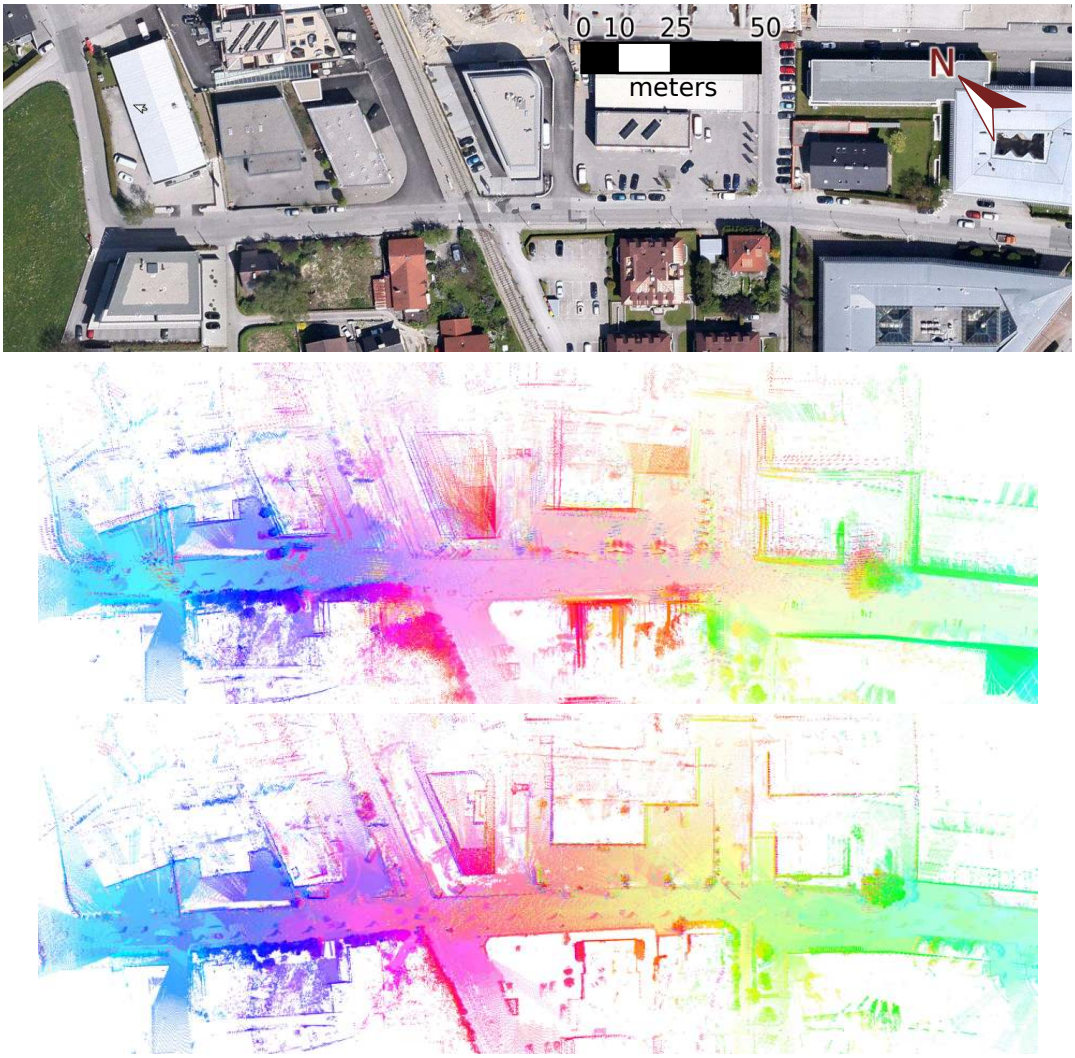


Figure 14. A more detailed view of the *Salzburg* data set from Fig. 13. The points are colored by their timestamps to provide contrast. The crane arm is approximately 42 m away from the trajectory. Top: The initial point cloud. Bottom: The optimized point cloud after processing with the semi-rigid registration algorithm.

Figure 15. A small loop in *Dortmund 1* (Germany) as acquired by the Lynx Mobile Mapper (top) and after the semi-rigid optimization has been applied (center). In the bottom, a more detailed view on the park with an advertising column and two trees initially (left) and after optimization (right).

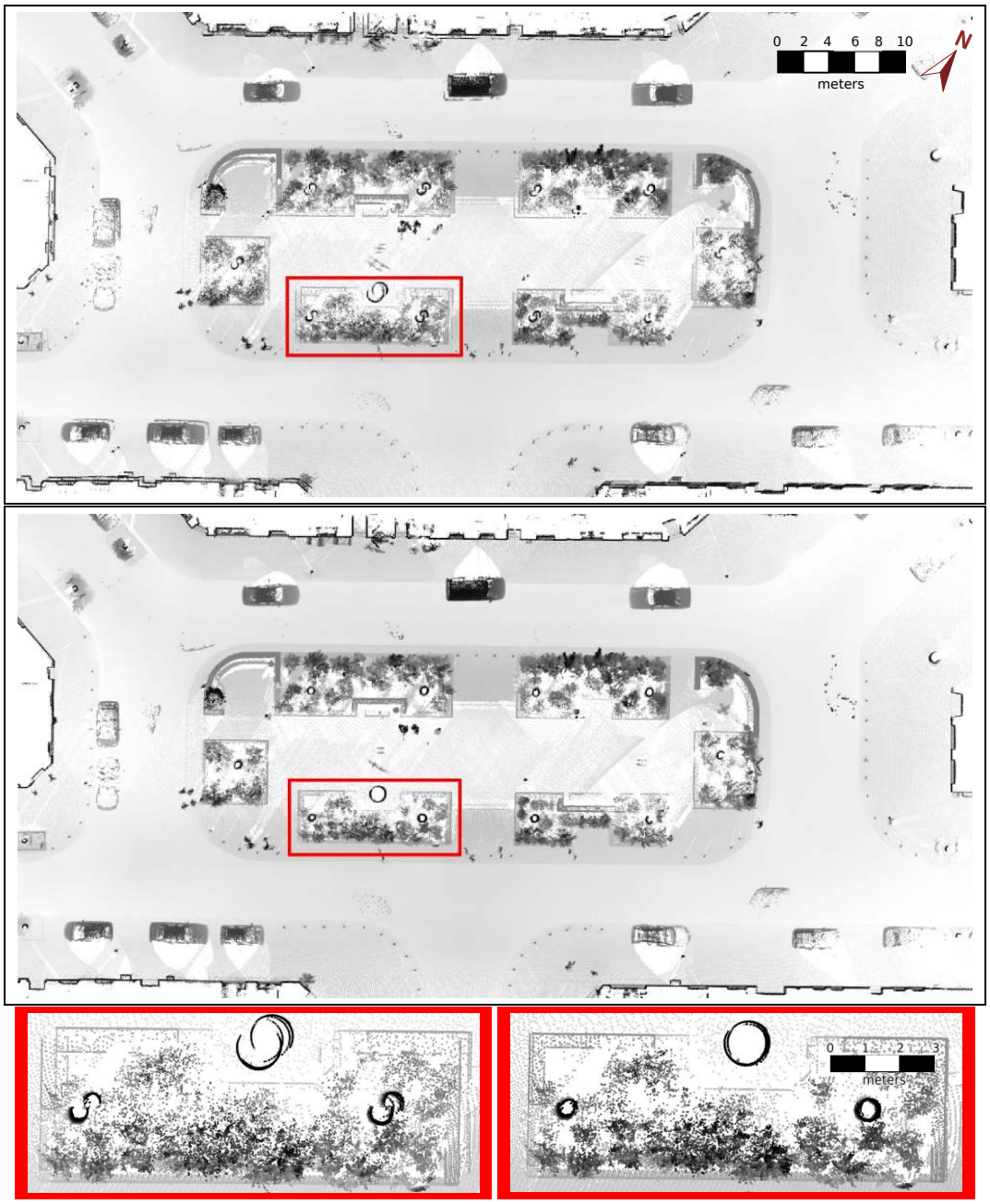


Figure 16. The larger *Dortmund 2* set data set acquired by TopScan GmbH. The state of the art original data set is shown in the top and the bottom left. The data set after optimization is shown in the center and bottom right.

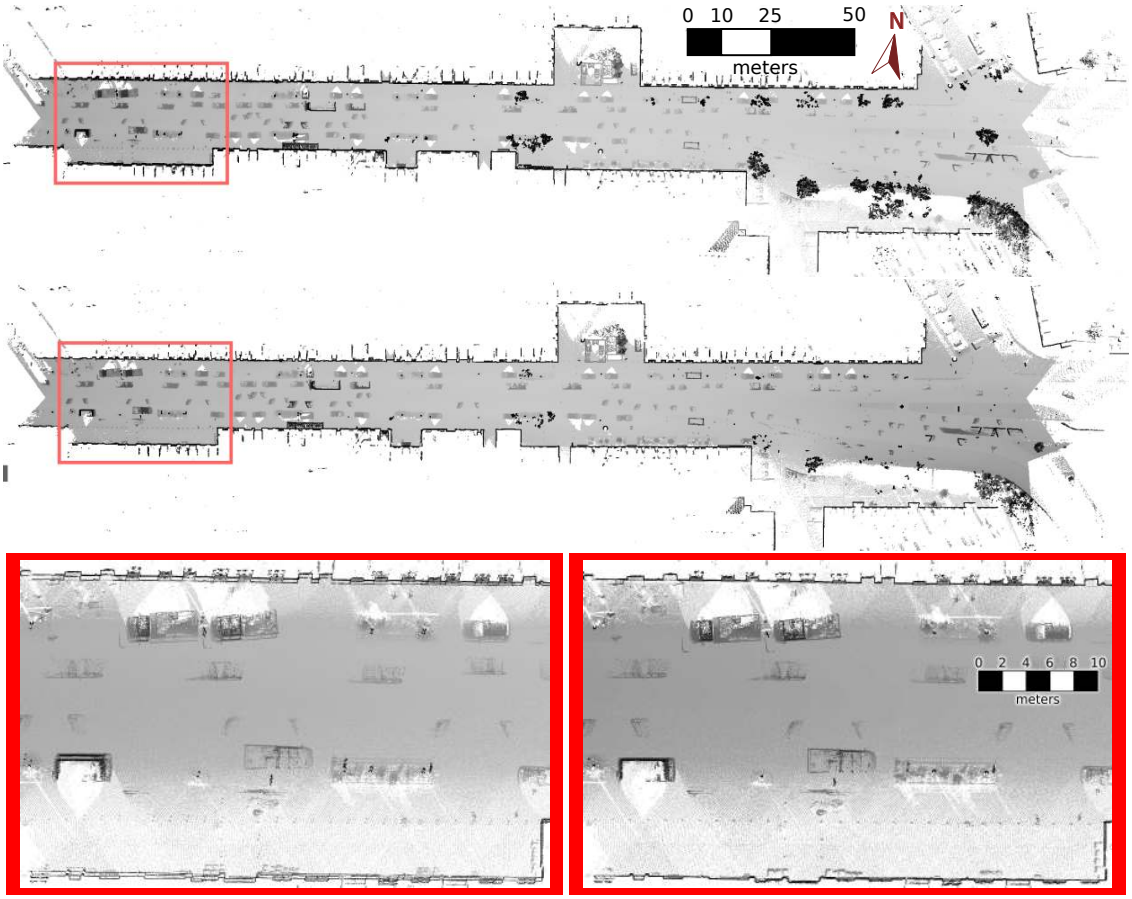
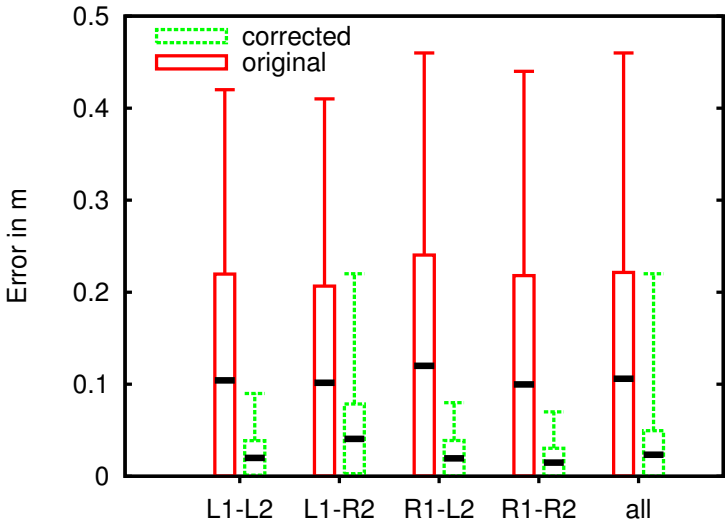


Figure 17. A comparison of the initial and optimized facade errors in the *Dortmund 2* data set. The box plots show the average (black line), the maximal error (whisker) and one standard deviation above and below the datum (box). We summarize each combination of sensor/pass in the same sequence as in Fig. 18, i.e. left sensors for the first and second pass (L1-L2), left sensor for the first pass and right sensor for the second pass (L1-R2) and so on. In addition, the overall errors (all) are plotted as well.



of the point cloud. Averages and variances are also given in Fig. 17. The average error is reduced to only 2.3 cm. The standard deviation is reduced from 11.5 cm to 2.6 cm. Even the maximum errors are reduced significantly. Similar to the indoor data sets we have also plotted the deviation between the first and second pass of the facades of the nearby buildings. The deviations both before and after optimization are color coded and displayed in Fig. 20.

In addition to the evaluation of the inner errors of the point clouds we also evaluated their external accuracy. This was done by way of five control points that were distributed in the scene. Each of the control points were located within a global reference with an absolute accuracy of approx. 1 cm by a professionel surveyor on site. The control points were also manually located in the point clouds of both 2D scanners for both passes. This is with the exception of a single control point that was not seen by one scanner in one of the passes. Thus, a total of 19 distance errors for the original data set as well as the optimized data set were determined. The result of this evaluation is given in Table 3. The difference between the original and the optimized data set has a minimum of less than 1 cm while the variances are 5.6 cm or 4.9 cm respectively. Given the small number of independent sample points we can conclude that the global error did not change to a statistically significant degree. This behavior is encouraging, since the optimization algorithm did not currently take into account the certainty with which the pose of the vehicle was measured. Furthermore, the coordinate system was fixed with respect to an arbitrary pose estimate, in our case the beginning of the trajectory of the first pass. We expect that the algorithm will also improve on the external accuracy of a given data set when GNSS measurements are used to define the global coordinate system and GNSS measurement certainty is taken into account.

Figure 18. Facade errors with current state of the art techniques in the *Dortmund 2* data set. From top to bottom: Left sensor in the first pass compared to the left sensor in the second pass. Left sensor in the first pass compared to the right sensor in the second pass. Right sensor in the first pass compared to the left sensor in the second pass. Right sensor in the first pass compared to the right sensor in the second pass.

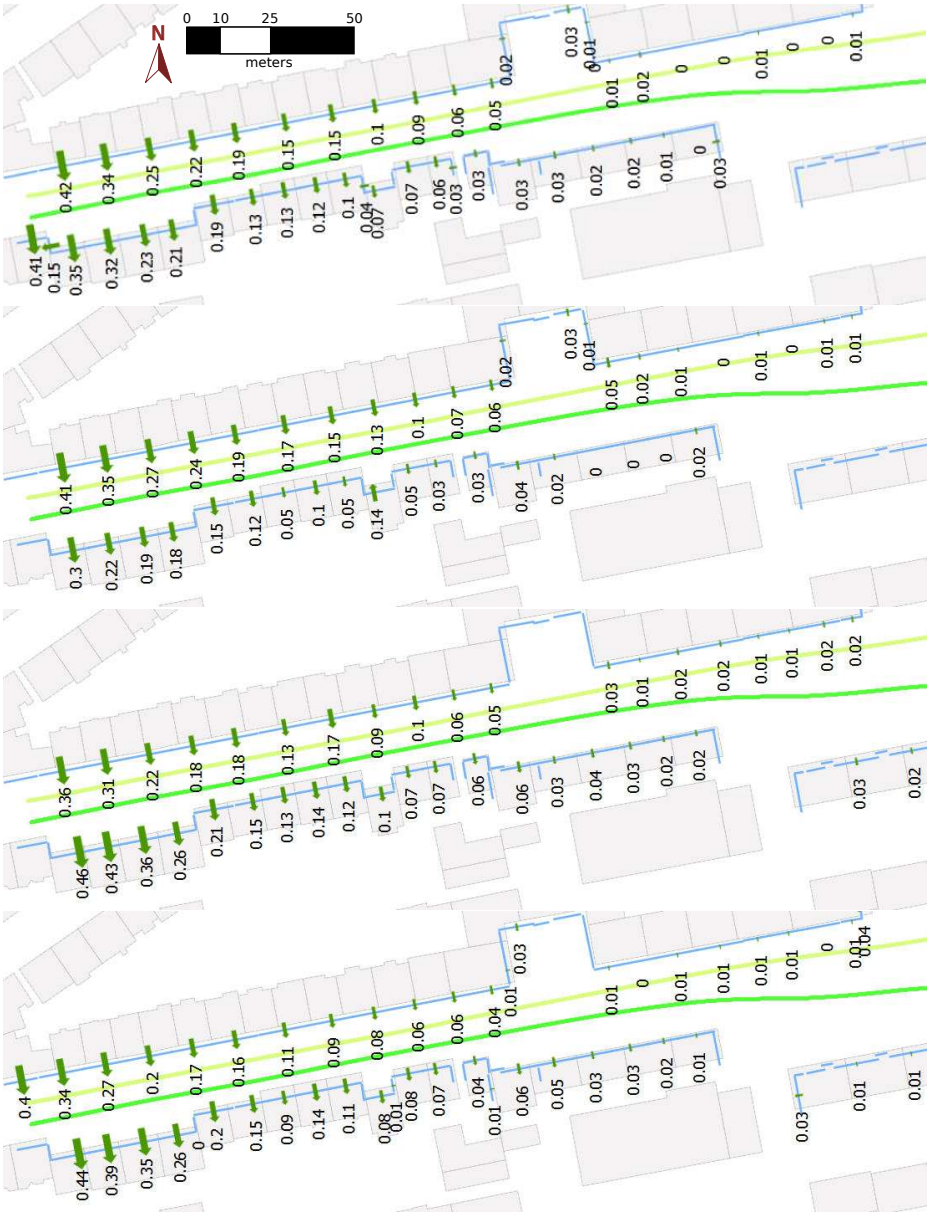


Figure 19. Facade errors after the proposed optimization in the *Dortmund 2* data set. From top to bottom: Left sensor in the first pass compared to the left sensor in the second pass. Left sensor in the first pass compared to the right sensor in the second pass. Right sensor in the first pass compared to the left sensor in the second pass. Right sensor in the first pass compared to the right sensor in the second pass.

Figure 20. Facade deviations in the *Dortmund 2* data set. Refer to the legend in the bottom for deviation in m. Facades marked with "a" are those in the original Lynx Mobile Mapper data. Facades marked "b" show the deviations after semi-rigid optimization.

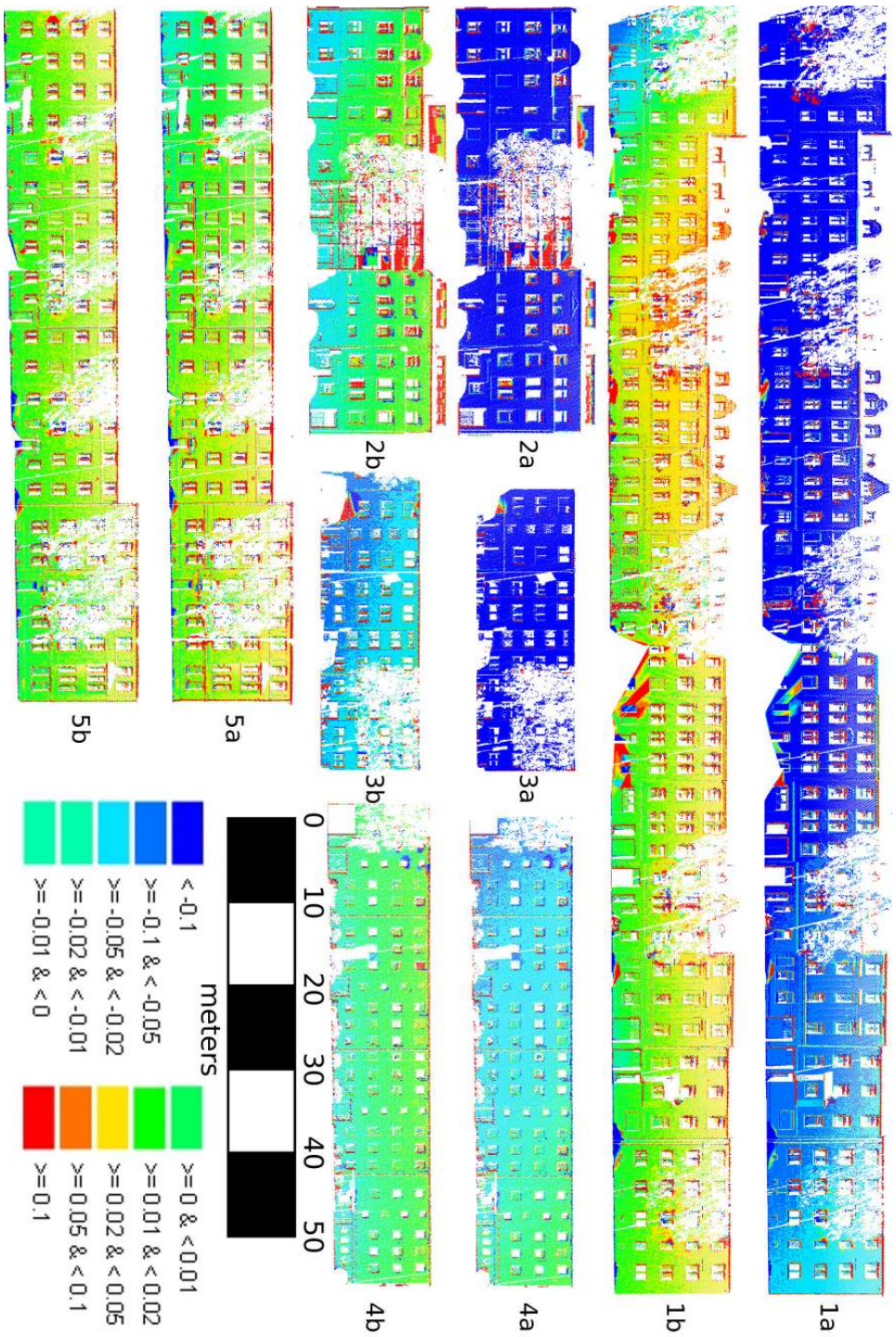


Table 3. Differences between the position of the control points on site and the position in the point cloud, both for the original Lynx Mobile Mapper data (left) and the optimized point cloud (right).

	original	optimized
L1-L2	0.132	0.178
	0.121	0.113
	0.118	0.171
	0.238	0.267
	0.245	0.171
L1-R2	0.121	0.171
	0.130	0.115
	0.105	0.156
	0.241	0.267
	0.282	0.202
R1-L2	0.099	0.124
	0.139	0.123
	0.151	0.169
	0.214	0.254
	0.156	0.158
R1-R2	0.099	0.129
	0.147	0.130
	0.226	0.242
	0.157	0.167
average	0.164	0.174
standard deviation	0.055	0.049

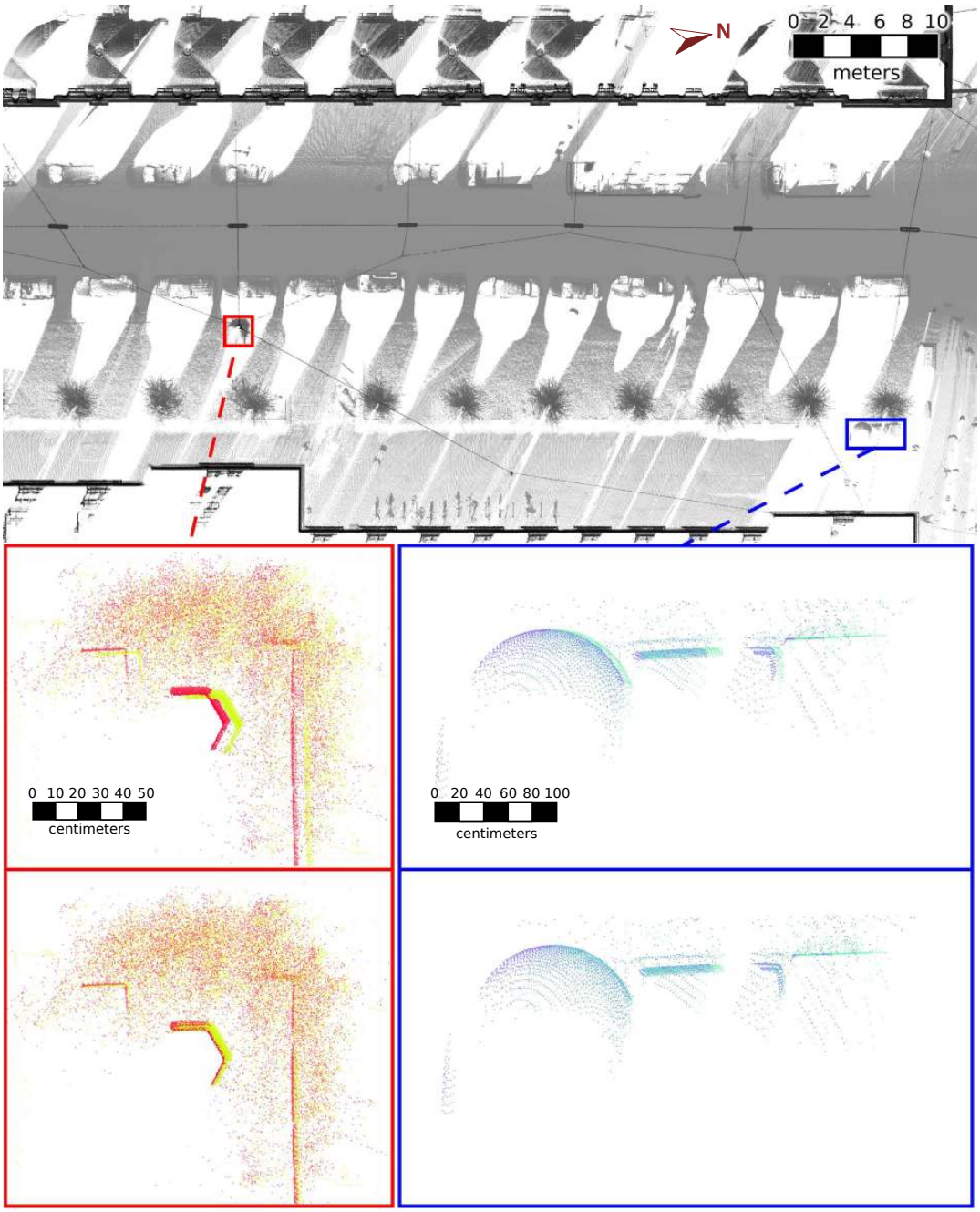
5.2.3. Riegl VMX-450

We also applied the semi-rigid optimization to data that was acquired by the Riegl VMX-450 laser scanner system. This *Vienna* data set is depicted in Fig. 21. Unfortunately, no quantitative evaluation of this data set is available. However, as in previous experiments we can visually inspect the results of the semi-rigid registration. This particular data set is initially of relative high quality but some errors are still present. Applying the optimization algorithm reduces the occurrence of surfaces appearing multiple times and leads to a more appealing point cloud.

6. Conclusions

The fully automatic calibration algorithm for mobile laser scanners as presented in this paper has proven to increase point cloud quality. It is capable of finding more precise calibration parameters than manual estimation. The algorithm is robust against non-systematic errors that are necessarily present in any mobile laser scanning system, such as laser scanner noise and erroneous pose estimations due to slipping wheels. A minor issue of the algorithm is the dependency on the environment. In our experiments the robot primarily moved on a level surface. This means some calibration parameters, i.e.,

Figure 21. The *Vienna* data set acquired by the Riegl VMX-450 mobile laser scanning system. Since the initial data set is of so high a quality that no difference is visible between the initial and optimized point cloud at the scale in the top, some details have been enlarged. The points are colored by their timestamps to provide contrast. In the detailed views, the top image shows the initial point cloud while the bottom one shows the optimized point cloud.



the upwards translation of the laser scanner in relation to the vehicle frame is less constrained by the quality measure as other parameters.

The formulation and implementation of the algorithm is generalized to allow for a wide range of sensors to be calibrated. In future work we will study how well the algorithm deals with multiple sensors and additional sensors of different modality.

The proposed semi rigid registration algorithms has shown that it is capable of processing and significantly improving upon a variety of data sets. It exceeds current state of the art rigid registration techniques not only when comparing the final maps produced, but also when comparing the inner deformation of subsections of the point cloud with ground truth. In future work, we will incorporate information about pose measurement certainty, especially GNSS error estimates to improve the overall global accuracy. Since the semi-rigid registration may converge only to a local minimum the conditions for the convergence must be inspected in more detail in the future. Furthermore, we plan to compare the precise maximum likelihood 3D point clouds with some reference independent points using a global coordinate system and aim at developing algorithms for improving the global accuracy of the point clouds.

Acknowledgements

The authors would like to thank our Master's students Vaibhav Bajpai, Vitali Bashko, Mohammad Faisal, Markus Harz, Bojan Kocev, and Vladislav Perelman for their contribution for setting up the car, TopScan GmbH for providing the Dortmund data set, and RIEGL Laser Measurement Systems GmbH for providing the Salzburg and Vienna data sets. This work was partially supported by the AUTOSCAN project (BMWi KF2470003DF0).

Conflict of Interest

The authors declare no conflict of interest.

References

1. Martinelli, A.; Tomatis, N.; Tapus, A.; Siegwart, R. Simultaneous Localization and Odometry Calibration for Mobile Robot. International Conference on Intelligent Robots and Systems, Las Vegas. Blub, 2003, pp. 75 – 85.
2. Baziw, J.; Leondes, C. In-Flight Alignment and Calibration of Inertial Measurement Units. Part II: Experimental Results. *IEEE Transactions on Aerospace and Electronic Systems* **1972**, *8*, 450 – 465.
3. Nebot, E.; Durrant-Whyte, H. Initial Calibration and Alignment of Low-cost Inertial Navigation Units for Land Vehicle Applications. *Journal of Robotic Systems* **1999**, *16*, 81– 92.
4. Skaloud, J.; Schaer, P. Towards Automated LiDAR Boresight Self-calibration. Proceedings of the 5th International Symposium on Mobile Mapping Technology, 2007.
5. Rieger, P.; Studnicka, N.; Pfennigbauer, M. Boresight Alignment Method for Mobile Laser Scanning Systems. *Journal of Applied Geodesy* **2010**, *4*, 13 – 21.

6. Talaya, J.; Alamus, R.; Bisch, E.; Serra, A.; Kornus, W.; Baron, A. Integration of a Terrestial Laser Scanner with GPS/IMU Orientation Sensors. *International Archives of Photogrammetry and Remote Sensing* **2004**, *5*.
7. Underwood, J.P.; Hill, A.; Peynot, T.; Scheding, S.J. Error Modeling and Calibration of Exteroceptive Sensors for Accurate Mapping Applications. *Journal of Field Robotics* **2009**, *27*, 2 – 20.
8. Sheehan, M.; Harrison, A.; Newman, P. Self-calibration for a 3D Laser. *The International Journal of Robotics Research* **2011**, *31*, 675 – 687.
9. Mills, D.L. Internet Time Synchronization: the Network Time Protocol. *IEEE Transactions on Communications* **1991**, *39*, 1482–1493.
10. Olson, E. A Passive Solution to the Sensor Synchronization Problem. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010, pp. 1059 – 1064.
11. Maes, F.; Collignon, A.; Vandermeulen, D.; Marchal, G.; Suetens, P. Multimodality Image Registration by Maximization of Mutual Information. *IEEE Transactions on Medical Imaging* **1997**, *16*, 187 – 198.
12. Cohen, L.D.; Cohen, I. Deformable Models for 3D Medical Images using Finite Elements and Balloons. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992, pp. 592 – 598.
13. Williams, J.A.; Bennamoun, M.; Latham, S. Multiple View 3D Registration: A Review and a New Technique. Proceedings of the International Conference on Systems, Man, and Cybernetics, 1999, Vol. 3, pp. 497 – 502.
14. Pitzer, B.; Stiller, C. Probabilistic Mapping for Mobile Robots using Spatial Correlation Models. IEEE International Conference on Robotics and Automation (ICRA '10), 2010, pp. 5402–5409.
15. Chui, H.; Rangarajan, A. A New Point Matching Algorithm for Non-Rigid Registration. *Computer Vision and Image Understanding* **2003**, *89*, 114–141.
16. Unnikrishnan, R.; Hebert, M. Denoising Manifold and Non-Manifold Point Clouds. 18th British Machine Vision Conference (BMVC), 2007, pp. 96.1–96.10.
17. Mitra, N.J.; Flöry, S.; Ovsjanikov, M.; Gelfand, N.; Guibas, L.; Pottmann, H. Dynamic Geometry Registration. Proceedings of the fifth Eurographics symposium on Geometry processing; , 2007; Vol. 257, pp. 173 – 182.
18. Brown, B.J.; Rusinkiewicz, S. Global Non-Rigid Alignment of 3-D Scans. *ACM Transactions on Graphics* **2007**, *26*, 21.
19. Besl, P.; McKay, N. A Method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* **1992**, *14*, 239 – 256.
20. Elseberg, J.; Borrmann, D.; Lingemann, K.; Nüchter, A. Non-rigid Registration and Rectification of 3D Laser Scans. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10), 2010, pp. 1546–1552.
21. Stoyanov, T.; Lilienthal, A.J. Maximum Likelihood Point Cloud Acquisition from a Mobile Platform. Proceedings of the IEEE International Conference on Advanced Robotics (ICAR '09), 2009, pp. 1 – 6.

22. Bosse, M.; Zlot, R. Continuous 3D Scan-Matching with a Spinning 2D Laser. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09), 2009, pp. 4312–4319.
23. Bosse, M.; Zlot, R.; Flick, P. Zebedee: Design of a Spring-Mounted 3-D Range Sensor with Application to Mobile Mapping. *Robotics, IEEE Transactions on* **2012**, *28*, 1104–1119.
24. Glennie, C.L. Calibration and Kinematic Analysis of the Velodyne HDL-64E S2 Lidar Sensor. *Photogrammetric Engineering and Remote Sensing* **2012**, *78*, 339 – 347.
25. Powell, M.J.D. An Efficient Method for finding the Minimum of a Function of Several Variables without calculating Derivatives. *The Computer Journal* **1964**, *7*, 155–162.
26. Elseberg, J.; Magnenat, S.; Siegwart, R.; Nüchter, A. Comparison of Nearest-Neighbor-Search Strategies and Implementations for Efficient Shape Registration. *Journal of Software Engineering for Robotics (JOSER)* **2012**, *3*, 2 – 12.
27. Elseberg, J.; Borrmann, D.; Nüchter, A. Efficient Processing of Large 3D Point Clouds. 2011 XXIII International Symposium on Information, Communication and Automation Technologies (ICAT '11), 2011, pp. 1–7.
28. Nüchter, A.; Elseberg, J.; Schneider, P.; Paulus, D. Study of Parameterizations for the Rigid Body Transformations of The Scan Registration Problem. *Journal Computer Vision and Image Understanding (CVIU)* **2010**, *114*, 963–980.
29. Davis, T.A. Algorithm 849: A Concise Sparse Cholesky Factorization Package. *ACM Trans. Math. Softw.* **2005**, *31*, 587 – 591.
30. Meschelke, K. Prüfverfahren für terrestrische Laserscanner an der HCU Hamburg. Terrestrisches Laserscanning (TLS 2011) Beiträge zum DVW-Seminar in Fulda; , 2011; pp. 69–92.
31. Andreas Nüchter et al.. 3DTK – The 3D Toolkit. http://slam6d.sourceforge.net/, 2011.

Chapter 4

Conclusion

> An Fortschritt glauben heißt nicht glauben daß ein Fortschritt schon geschehen ist. Das wäre kein Glauben.
> Believing in progress does not mean believing that any progress has yet been made. That would not be believing.
>
> Franz Kafka

This dissertation focuses on the issue of producing accurate 3D models from range data acquired by MLS systems. To arrive at accurate representations of our world many obstacles have to be overcome. These obstacles multiply and grow when the source of the data stems from moving platforms instead of tripods. Nevertheless, methods developed for stop-and-go laser scanning give valuable insights: Efficiently handling of 3D point cloud data is very similar in both contexts. In addition to the octree data structure we can also employ some strategies from 3D rigid registration. The SLAM back-end that we created for this end serves as an important part in the novel semi-rigid registration algorithm presented in this work. The issue of calibration is somewhat unique to MLS. However, solving this problem is vital for creating high quality 3D point clouds. We have developed a solution that is capable of calibrating MLS systems in any configuration using normal production data, requiring no special calibration data acquisition phase. The back-end for this algorithm is a well-known general function minimizer called Powell's method. Combined with a well-chosen metric for scan quality the presented algorithm is an efficient and effective solution to the calibration problem.

Together, all of these parts create a system that is ideally suited to create higher quality 3D models. This was demonstrated on a large series of data sets. We ran a number of experiments under controlled conditions where the ground truth was known. Furthermore, data from a variety of mobile laser scanners was used to test the presented algorithms. Most importantly, the two market leaders in the area of MLS, RIEGL [69] and Optech [38] provided us with point clouds to compare the performance of our calibration algorithm as well as our semi-rigid registration algorithm to the state of the art techniques. We also constructed the robotic platform Irma3D, primarily for acquiring our own mobile scanning data. Experiments on a combination of this data and state-of-the art data rounds up the analysis of the presented algorithms.

In addition to providing tools for optimizing the quality of 3D point clouds acquired by mobile scanners this work also demonstrates the benefit of changing the classical way of acquiring this data. Any modification that allow for multiple range measurements of the same locality from different perspectives opens up the possibility for quality improvements by algorithmic means. This can be achieved by using a rotating 3D scanner as a primary scanning device, as we use on Irma3D, instead of a 2D scanner. Alternatively, classical MLS systems with one or more 2D laser scanners may traverse the same environment multiple times. These seemingly redundant measurements allow for an overall reduction of errors via registration algorithms for mobile laser scanners.

Bibliography

[1] Heron Alexandrinus. *Vermessungslehre und Dioptra : griechisch und deutsch*, volume 3 of *Heronis Alexandrini opera quae supersunt omnia*. B. G. Teubner, 1903.

[2] Jon Bedford, Trevor Pearson, and Bernard Thomason. *Traversing the Past*. English Heritage, 2011.

[3] PL Bender, DG Currie, RH Dicke, DH Eckhardt, J Ek Faller, WM Kaula, JD Mulholland, HH Plotkin, SK Poultney, EC Silverberg, et al. The Lunar Laser Ranging Experiment. *Science*, 182(4109):229–238, 1973.

[4] D. J. Harding G. S. Berghoff. Fault Scarp Detection Beneath Dense Vegetation Cover: Airborne Lidar Mapping of the Seattle Fault Zone, Bainbridge Island, Washington State. Technical Report 0023, National Aeronautics and Space Administration, 2000.

[5] P. Besl and N. McKay. A Method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[6] Peter Biber. The Normal Distributions Transform: A New Approach to Laser Scan Matching. WSI-Report 3, University of Tübingen, WSI/GRIS, 2003.

[7] Claus Brenner. *Dreidimensionale Gebäuderekonstruktion aus digitalen Oberflächenmodellen und Grundrissen*. PhD thesis, Universität Stuttgart, Fakultät für Bauingenieur– und Vermessungswesen, 2000.

[8] Heiko Bülow and Andreas Birk. Spectral 6-DOF Registration of Noisy 3D Range Data with Partial Overlap. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35:954–969, 2013.

[9] Xsens Technologies B.V. `http://www.xsens.com`.

[10] Dane Carlson. 3D Laser Scanning Accident Sites. `http://www.business-opportunities.biz/2011/07/18/3d-laser-scanning-accident-sites/`. Accessed: 2013-6-6.

[11] A. Censi and S. Carpin. HSM3D: Feature-Less Global 6DOF Scan-Matching in the Hough/Radon Domain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3899–3906, 2009.

[12] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Image and Vision Computing*, 10(3):145–155, 1992.

[13] D. M. Cole and P. M. Newman. Using Laser Range Data for 3D SLAM in Outdoor Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)*, pages 1556–1563, Florida, USA, 2006.

[14] M. Connor and P. Kumar. Fast Construction of k-Nearest Neighbor Graphs for Point Clouds. *Visualization and Computer Graphics, IEEE Transactions on*, 16(4):599–608, 2010.

[15] Microsoft Corporation. `http://www.microsoft.com`.

[16] TOPCON CORPORATION. `http://global.topcon.com`.

[17] D. Borrmann and A. Nüchter and M. Đakulović and I. Maurović and I. Petrović and D. Osmanković and J. Velagić. The Project ThermalMapper – Thermal 3D Mapping of Indoor Environments for Saving Energy. In *Proceedings of the 10th Symposium on Robot Control (SYROCO)* , pages 1–8, Dubrovnik, Croatia, September 2012.

[18] Delock. `http://www.delock.de/`.

[19] James Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Technical report, Stanford University, Palo Alto, USA, 2006.

[20] Leonhard Euler. Formulae Generales pro Translatione Quacunque Corporum Rigidorum. *Novi Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 20(E478):189–207, 1776.

[21] Andrea Faccioli and Marco Fumanti. Le Ultime Tecnologie Hanno Aiutato i Primi Soccorsi alla Costa Concordia. *Geomedia*, 16(1):22–24, 2012.

[22] Inc. FARO Technologies. `http://www.faro.com`.

[23] Marting A. Fischler and Robert. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24:381–395, 1981.

[24] David A. Forsyth and Jean Ponce. *Computer Vision, A Modern Approach*. Prentice Hall, 2003.

[25] Björn Van Genechten. *Theory and Practice on Terrestrial Laser Scanning: Training Material based on Practical Applications*. Universidad Politecnica de Valencia Editorial, 2008.

[26] Todor Georgiev, Zhan Yu, Andrew Lumsdaine, and Sergio Goma. Lytro Camera Technology: Theory, Algorithms, Performance Analysis. In *Proceedings SPIE 8667*, 2013.

[27] Leica Geosystems. `http://www.leica-geosystems.com`.

[28] Optris GmbH. `http://www.optris.de/`.

[29] Raytrix GmbH. http://www.raytrix.de.

[30] Zoller + Fröhlich GmbH. http://www.zofre.de.

[31] Bianca Gordon. *Zur Bestimmung von Messunsicherheiten terrestrischer Laserscanner*. PhD thesis, Technische Universität Darmstadt, September 2008.

[32] N. D. Haasbroek. *Gemma Frisius, Tycho Brahe and Snellius and their triangulations*. Rijkscommissie voor Geodesie, 1968.

[33] P.R.N. Hobbs, B. Humphreys, J.G. Rees, D.G. Tragheim, L.D. Jones, A. Gibson, K. Rowlands, G. Hunter, and R. Airey. Monitoring the Role of Landslides in Soft Cliff Coastal Recession. In *Instability - Planning and Management*, pages 589–600, 2002.

[34] M. Hofer and H. Pottmann. Orientierung von Laserscanner-Punktwolken. *Vermessung & Geoinformation*, 91:297–306, 2003.

[35] LTD. HOKUYO AUTOMATIC CO. http://www.hokuyo-aut.jp.

[36] MESA Imaging. http://www.mesa-imaging.ch.

[37] Canon Inc. http://www.canon.com/.

[38] Optech Inc. http://www.optech.ca/.

[39] Heribert Kahmen. *Angewandte Geodäsie: Vermessungskunde*. de Gruyter, 2005.

[40] Freddie Kern, Siegrist Bettina, Uwe Huxhagen, and Stefan Mehlig. Genauigkeitsvergleich verschiedener Zielmarkendesigns. In *Photogrammetrie, Laserscanning, Optische 3D-Messtechnik: Beiträge der Oldenburger 3D-Tage 2010*, pages 96–105, 2010.

[41] Henry C. King. *The History of the Telescope*. Dover Books on Astronomy. Dover Publications, 1955.

[42] Michael Kirchhof, Boris Jutzia, and Uwe Stilla. Iterative Processing of Laser Scanning Data by Full Waveform Analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):99–104, 2007.

[43] Mark Lehner. *The Complete Pyramids: Solving the Ancient Mysteries*. Thames & Hudson, 2008.

[44] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anerson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proceedings of the ACM SIGGRAPH*, pages 131–144, New Orleans, USA, July 2000.

[45] Maren Lindstaedt, Thomas Kersten, Klaus Mechelke, and Tanja Graeger. Prüfverfahren für terrestrische Laserscanner – Gemeinsame geometrische Genauigkeitsuntersuchungen verschiedener Laserscanner an der HCU Hamburg. In *Photogrammetrie, Laserscanning, Optische 3D-Messtechnik: Beiträge der Oldenburger 3D-Tage 2012*, pages 264–275, 2012.

[46] Logitech. http://www.logitech.com/.

[47] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4:333–349, April 1997.

[48] Dorothea Ludwig, Sandra Lanig, and Martina Klärle. Location analysis for solar panels by LiDAR-Data with Geoprocessing - SUN-AREA. In Kristina Voigt Volker Wohlgemuth, Bernd Page, editor, *Shaker Verlag*, pages 83–89. Shaker Verlag, 2009.

[49] L Matejicek. Spatial Modelling of Air Pollution in Urban Areas with GIS: A Case Study on Integrated Database Development. *Advances in Geosciences*, 4:63–68, 2005.

[50] Donald Meagher. Geometric Modeling using Octree Encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

[51] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[52] G. M. Morton. A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. Technical Report 3, IBM Ltd., 1996.

[53] Fleet News. Laser scanner aids crash investigation. http://www.fleetnews.co.uk/news/2012/10/19/laser-scanner-aids-crash-investigation/45175/. Accessed: 2013-6-6.

[54] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light Field Photography with a Hand-Held Plenoptic Camera. Technical Report CSTR 2, Stanford University Computer Science, 2005.

[55] Anette Nielsen. Visual Representations, Usability and Urban Planning in Real-time 3D Geovisualization. In *8th AGILE Conference Estoril, Portugal*, 2005.

[56] Wolfgang Niemeier. *Ausgleichungsrechnung.* de Gruyter, 2002.

[57] Andreas Nüchter and Joachim Hertzberg. Towards Semantic Maps for Mobile Robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

[58] K. Ohno, S. Tadokoro, K. Nagatani, E. Koyanagi, and T. Yoshida. Trials of 3-D Map Construction using the Tele-operated tracked Vehicle Kenaf at Disaster City. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2864–2870, 2010.

[59] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, and Jann Poppinga. Fast Registration Based on Noisy Planes with Unknown Correspondences for 3D Mapping. *IEEE Transactions on Robotics*, 26(3):424–441, 2010.

[60] A. Pesci and G. Teza. Terrestrial Laser Scanner and Retroreflective Targets: an Experiment for Anomalous Effects Investigation. *International Journal of Remote Sensing*, 29(19):5749–5765, 2008.

[61] A. Posluschny. Wer wird denn gleich in die Luft gehen - archäologische Prospektion mittels Laserscanning. *hessenARCHÄOLOGIE*, pages 69–71, 2007.

[62] H. Pottmann, S. Leopoldseder, and M. Hofer. Simultaneous Registration of Multiple Views of a 3D Object. *ISPRS Archives*, 34(3A):265–270, 2002.

[63] John H. Rogers. *The Giant Planet Jupiter (Practical Astronomy Handbooks)*. Cambridge University Press, 2009.

[64] Szymon Rusinkiewicz and Marc Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 343–352, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[65] R.B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009.

[66] SICK Laser Range Finder. `http://www.sick.de`.

[67] B. Sittler. Revealing Historical Landscapes by Using Airborne Laser-Scanning - A 3D-Modell of Ridge and Furrow in Forests near Rastatt. *ISPRS Archives*, 36(8):258–261, 2004.

[68] D. Sobel. *Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time*. Bloomsbury Publishing, 2010.

[69] RIEGL Laser Measurement Systems. `http://www.riegl.com/`.

[70] Tom Gehrels. NASA Imaging Photopolarimeter (IPP). `http://nssdc.gsfc.nasa.gov/nmc/experimentDisplay.do?id=1972-012A-07`.

[71] Viktor T. Toth and Slava G. Turyshev. Finding the Source of the Pioneer Anomaly. *IEEE Spectrum*, 49(12):38–62, 2013.

[72] Christopher Urmson, Joshua Anhalt, J. Andrew (Drew) Bagnell, Christopher R. Baker , Robert E Bittner, John M Dolan, David Duggins, David Ferguson, Tugrul Galatali, Hartmut Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Alonzo Kelly, David Kohanbash, Maxim Likhachev, Nick Miller, Kevin Peterson, Ragunathan Rajkumar, Paul Rybski, Bryan Salesky, Sebastian Scherer, Young-Woo Seo, Reid Simmons, Sanjiv Singh, Jarrod M Snider, Anthony (Tony) Stentz, William (Red) L. Whittaker, and Jason Ziglar. Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge. Technical Report CMU-RI-TR, Robotics Institute, Pittsburgh, PA, April 2007.

[73] Inc. Velodyne Lidar. `http://velodynelidar.com`.

[74] The Fraunhofer VolksBot. `http://volksbot.de`.

[75] W. Wagner, C. Eberhöfer, M. Hollaus, and G. Summer. Robust Filtering of Airborne Laser Scanner Data for Vegetation Analysis. In *ISPRS Archives Volume - XXXVI-8/W2*, pages 56–61, 2004.

[76] Z. Zhang. Iterative Point Matching for Registration of Free–form Curves. Technical Report RR-1658, INRIA–Sophia Antipolis, Valbonne Cedex, France, 1992.

Previous Work

[77] A. Nüchter and J. Elseberg and P. Schneider and D. Paulus. Study of Parameterizations for the Rigid Body Transformations of The Scan Registration Problem. *Computer Vision and Image Understanding (CVIU)*, 2010.

[78] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally Consistent 3D Mapping with Scan Matching. *Journal of Robotics and Autonomous Systems (JRAS)*, 56(2):130–142, February 2008.

[79] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. The Efficient Extension of Globally Consistent Scan Matching to 6 DOF. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '08)*, Atlanta, GA, USA, June 2008.

[80] J. Elseberg, D. Borrmann, and A. A. Nüchter. Efficient Processing of Large 3D Point Clouds. In *Proceedings of the XXIII International Symposium on Information, Communication and Automation Technologies*, ICAT '11, pages 1–7, October 2011.

[81] J. Elseberg, D. Borrmann, K. Lingemann, and A. Nüchter. Non-rigid Registration and Rectification of 3D Laser Scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1546–1552, October 2010.

[82] J. Elseberg, R. Creed, and R. Lakaemper. A Line Segment Based System for 2D Global Mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[83] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter. Comparison of Nearest-Neighbor-Search Strategies and Implementations for Efficient Shape Registration. *Journal of Software Engineering for Robotics (JOSER)*, 3(1):2–12, 2012.