

TECHNISCHE UNIVERSITÄT DRESDEN

FACULTY OF COMPUTER SCIENCE
INSTITUTE OF SOFTWARE AND MULTIMEDIA TECHNOLOGY
CHAIR OF COMPUTER GRAPHICS AND VISUALIZATION
PROF. DR. STEFAN GUMHOLD

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science

Interactive Registration of 3D Scans in VR

Daniel Bekele
(Born 15th January 1995 in Jena, Mat.-No.: 309845)

Tutor: Prof. Dr. Stefan Gumhold

Dresden, February 24, 2019

Aufgabenstellung

0.1 Motivation and Goals:

When 3D scanning real objects several scans from different views need to be acquired and combined. For high quality reconstruction of a surface model, the relative 3D scanner poses of the different views need to be determined accurately before combining the points from the individual 3D scans. This process is also called registration. The registration problem can be split into coarse and fine registration. In coarse registration a registration with low accuracy is determined which is further refined by a fine registration, for which fully automatic methods work efficiently. The goal of the thesis is to study the coarse registration problem in an interactive setting where a user adjusts the relative poses in a VR environment and to compare its efficiency with a WIMP based interface, e.g. the one implemented in MeshLab [1].

0.2 Tasks:

- Literature research on registration approach with a focus on interactive and semi-automatic approaches
- Collection of a representative set of 3D scan collections
- Design of a VR environment for the interactive registration realizing the concept of a workbench and a natural alignment of the 3D scan collection that needs to be registered in a way the selection of individual scans is efficient.
- Implementation of VR-based picking and positioning strategy of individual point clouds based on the interaction capabilities of a VIVE VR set.
- Development and implementation of a strategy to guide the user through the process of registering all scans of a dataset
- Integration of an automatic fine registration algorithm like SparseICP [2].
- Evaluation of the developed methods in a qualitative user study based on a ground truth dataset provided by the Chair of Computer Graphics and Visualization.

0.3 Optional Tasks:

- Integration of a surface reconstruction algorithm in a background process to preview the reconstructed surface based on the current intermediate registration result
- Design and implement specific interactions and visualizations that help in a faster or more accurate user based registration
- Improvement of the realism of the VR environment by integrating detailed and textured 3D models.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tag dem Prüfungsausschuss der Fakultät Informatik eingereichte Arbeit zum Thema:

Interactive Registration of 3D Scans in VR

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den February 24, 2019

Daniel Bekele

Kurzfassung

Abstract

abstract text english

Contents

| | | |
|----------|---|-----------|
| 0.1 | Motivation and Goals: | 2 |
| 0.2 | Tasks: | 2 |
| 0.3 | Optional Tasks: | 3 |
| 1 | 3D scan registration | 2 |
| 1.1 | Introduction to registration algorithms | 2 |
| 1.1.1 | Correspondences | 3 |
| 1.1.2 | KD-Trees | 3 |
| 1.1.3 | The iterative closest point algorithm | 4 |
| 1.1.4 | Subsampling | 5 |
| 2 | Interactive VR-Concepts | 6 |
| 2.1 | The virtual environment | 6 |
| 2.1.1 | The virtual workspace | 6 |
| 2.1.2 | Scaling | 7 |
| 2.2 | User centred Design for VR | 7 |
| 2.2.1 | Constraints and Affordances | 7 |
| 2.2.2 | Visual - physical conflict | 8 |
| 2.2.3 | Feedback design | 8 |
| 2.2.4 | Picking and selection process | 8 |
| 2.2.5 | Grouped scans | 9 |
| 2.2.6 | Scan positioning | 10 |
| 2.2.7 | Additional application concepts | 11 |
| 2.3 | eine Grafik | 11 |
| 2.3.1 | Etwas Mathe | 11 |
| 2.3.1.1 | Verweise auf Literatur | 11 |
| | Bibliography | 14 |

1 3D scan registration

The registration problem is defined as finding the matching transformation for one or more sets of data to match to a target set of data.(BELEG) This task can be splitted into the coarse and fine registration. In order to work properly, or to achieve faster and better results, most registration algorithms require a coarse registration that is done before the actual algorithm starts.

"Depending on the method used, a quite accurate initial guess is required because some methods have convergence problems due to the presence of local minima."[10]

Fully automated algorithms for the registration problem mostly need 2 different algorithm for coarse and fine alignment. As proposed in a paper by Ji [5] using an improved method of the ICP for fine alignment and a genetic algorithm for coarse alignment. Other than that Chao uses a semi automatic approach feature based methods. To reduce rotation errors the user decides manually the correct rotation [2]

The method proposed by this thesis is a mixture of a manually coarse alignment and simple ICP version that should be capable of running in real time, making it a subject of interest for gaming applications as well as real time CAD applications. The coarse registration problem was often solved manually by used a WIMP[explain WIMP] based interface like in Chaos paper [2]. To enhance the process further and in order to give the user more insight and controll over his 3D models, a virtual environment is used in our application instead.

The main goal of this thesis is to investigate if this coarse alignment problem can be solved more efficiently by introducing the user to a virtual environment capable of fully displaying the point clouds in all three dimensions, instead of using a WIMP based interface. The fine registration is then automatically performed by an ICP, in the best case this should execute real time with an animation.

1.1 Introduction to registration algorithms

To achieve this goal of fine registration several steps are necessary:

1. Find corresponding points between the roughly aligned sets of data
2. Find a formula to retrieve the transformation needed to align dataset A to dataset B using the

correspondences found in 1.

3. Calculate the actual transformation.
4. Apply the actual transformation

1.1.1 Correspondences

To find corresponding points between 2 sets of points a fast and efficient method is needed, hence the application should work in real time and not use more than 500ms as this is only the first step for the registration algorithm. The k-nearest neighbour search (knn-search) is commonly used for such problems, as it includes the distances between the nearest points and should deliver good results, given the scans are near together and approximately have the correct rotation. The circumstances of the application fulfill this requirements, since it needs the user to coarse align the scans before the automated fine alignment. The knn-search does calculate, as its name implies, the distance to the nearest points. To estimate the nearest neighbours, distances between the individual points need to be calculated and then sorted to find the closest points.

Given this situation a simple knn-search with the brute Force method fails. The method would calculate all distances between all points, the resulting complexity of $O(mnd)$ [3], with m as nubmer of all points from point cloud 1 and n with the number of all points from point cloud 2. d is the number of nearest points that are obtained and stored. This does not meet the requirement of the application to work in real time with point clouds with a higher number of points. To enhance the efficiency of this method many different acceleration data structures have been introduced. The one used in the actual implementation of the application for this thesis is the so called KD-tree.

1.1.2 KD-Trees

"k-d trees are a generalization of binary search trees."[7]

In theory this means that all points are represented by a binary tree structure. The root node contains the whole point cloud. The variable k defines the dimensions which is 3 in our example, hence we are working in 3D space when aligning scans. This variable can be higher if e.g. the colours of scans are as additional information available. Building such a tree has an average runtime in the complexity of $O(\log n)$, to optimize it $O(n \log n)$ [1] Their main advantage is their runtime for knn-queries:

"for nearest neighbor queries [empirically observed average running time of $O(\log n)$]." [1]

Using KD-trees for a knn-query results in a much better performance since building a kd tree and using

it costs less than using brute force method. Furthermore through parallelization the actual runtime can further be reduced. The code excerpt below shows a snippet which uses a KD-tree for a knn-query executing parallel. This implementation is used during the execution of the fine alignment of the version of the ICP this application uses. The code is taken from the stanford university.

```
/// Find closest point
#pragma omp parallel for
for(int i=0; i<X.cols(); ++i) {
    Q.col(i) = Y.col(kdtree.closest(X.col(i).data()));
}
```

Newer researches have shown that cached KD-trees algorithms using GPU programming have better results[3], this will not be the case in our application since the GPU is fully occupied by providing a rendered stereoscopic view.

The resulting correspondencies can now be used to calculate a rigid-body transformation that aligns two point clouds. To achieve this, the application uses a commonly known algorithm called Iterative closest point (ICP).

1.1.3 The iterative closest point algorithm

"ICP starts with two meshes and an initial guess for their relative rigid-body transform, and iteratively refines the transform by repeatedly generating pairs of corresponding points on the meshes and minimizing an error metric."[9]

In this case the Transformation is a translation and a rotation matrix that when applied to the point cloud aligns it to the target point cloud.

The initial guess is provided by the user with a coarse registration. The ICP then makes use of step 2-4 (see top) always iteratively converging to a local optimum.[8] The ICP algorithm in the variant that was and is widely used in registration problems and was introduced 1992 by P.J. Besl and Neil D. McKay[8]. Over the years several optimization steps have been added to the ICP in order to boost the speed and efficiency.

The variants used by our program is the regular point to point variant using reiterative weighting. This method uses xxxxxxxx for more stable solutions. Due to problems with the integration of sparse ICP, which uses a more stable error metric the simpler and faster ICP variant has been chosen. This tradeoff can lead to NAN or INF results when using the method.

Before starting the actual algorithm it has to be considered that the real time requirement can obviously

not be fulfilled with larger point clouds as shown in measurements in table 1 [see table with times]. But aligning and grouping several point clouds to one and aligning this greater point cloud will sooner or later result in a large point cloud that has to be aligned. To still fulfill the real time requirement a subsampling method is needed.

1.1.4 Subsampling

The runtime tests as shown in the table show a significant drop rate in runtime when adding too much points. This is especially the case if the point number reach certain limits. We suspect this is the case when the needed storage for the kd trees which lays in $O(n * m)$ [beleg not found] surpasses the computers cache.

To avoid this problem a subsampling method has been introduced, which uses randomized ranged subsampling. Every point cloud of the group of clouds that is aligned to the target group is counted. The resulting number is divided by the subsampling range. The range indicates that every xth point is sampled. The exact point is randomly chosen in between the ranges distance, counting upwards. The resulting subsampled point cloud is now used for the ICP algorithm, reducing the complexity of the actual task by the factor x.

To avoid large overhead due to the creation of randomized numbers the function uses the default and efficient random Microsoft c++ library `<random.h>` and only array access with offsets. If the ICP is called again with the same point clouds, the application uses the already stored values, further increasing its efficiency.

2 Interactive VR-Concepts

2.1 The virtual environment

The virtual environment is defined as the virtual entity that surrounds the user of a VR technology during the time he is using an HMD. Different to augmented reality which adds an additional layer/interface to the existing reality, letting the user see his natural environment, VR completely surrounds the user and the actual environment is not visible anymore. What the user can see and perceive is solely depended from the created virtual world[6].

This environment should provide orientation in the virtual room as well as interfaces to work in it. The virtual environment is not limited but moving in it is not as easy, as natural movement is restrained to the maximum trackable area of the used VR equipment.

2.1.1 The virtual workspace

In order to provide the user an appropriate workspace in VR, a virtual room has been designed. The maximum recommended trackable space for the virtual room using the Vive System is a 3.5 meters x 3.5 meters quadratic area. To assemble the point clouds efficiently the user has to gather the components and place them on an area. In order to support the users a virtual table has been designed. It is placed in the middle of the room with exact 1m distance to the room's walls. The resulting size of 1.5x1.5 meters is sufficient for most scans of objects used in this thesis. For a list of used scans and their size see [appendix link] To avoid cyber sickness or claustrophobic events, the room has only 1 wall going towards the ceiling. The other 3 sides of the virtual room have smaller walls of about 1 meter height. They allow sight to the applications background graphic. Its task is to emulate more space than there is, to avoid the user feeling trapped in a small 3.5m 3.5m room. The small walls should work as a fence signalling the end of the trackable area. The resulting room can be seen in figure xxx in a third perspective view [insert virtual room picture]

2.1.2 Scaling

Greater objects may not fit in the room at all and need to be scaled to fit. In order to establish a norm, scans scaled greater than the table, are scaled down to fit the table. On the other hand too small scans are scaled upwards to reach a minimum size of 0.5 meters. The scaling function is immediately called after the loading function is completed.

2.2 User centred Design for VR

When building an VR based application, traditional 2D user interface design cannot be simply adapted to 3D.(BELEG!) Using Popups or mouse emulation projected onto the view frustum of the user constraints interaction to a 2-Dimensional experience which is rather inefficient, compared to the possibilities VR could provide. To include VR Technology to the project, especially the cgV-Framework, new user interaction methods have to be provided and tested. Since the user is the active part of the application, controlling the actions and main methods of the scan alignment process, user centred design had to be chosen. Human centred design processes for interactive systems, ISO 13407 (1999), states:

"Human-centred design is an approach to interactive system development that focuses specifically on making systems usable."

There are many aspects of UCD, the following chapters will focus on the definition and examples used in this thesis corresponding application, showing their usage and further features.

2.2.1 Constraints and Affordances

Affordances describe the relationship between an abstract object or goal a user can achieve with the given actions he has[4]. To improve interaction design it is necessary that these affordances are appropriate and the given actions as easy doable[4]. In our application an affordance would be the relation between moving a scan that has been picked by moving the tracked motion controller.

Constraints on the other hand are limitations of the given actions a user has. This limitations may be necessary to not confuse the user or to simplify the usage of given methods. A user may move a scan by fault out of the trackable area or below the ground. By adding a simple collision detection that prevents the user from moving scans out of these limitations moving scans is less prone to errors. This would be a constraint to the action of moving the scan with the tracked motion controller. In the application moving scans below zero level is not possible, the scans bounding box is, moved tracked and evaluated. If an action would position any transformed corner point of the box, the translation is stopped and the

corresponding controller receives a signal to vibrate. This indicates a violation of physical constraints to the user, preventing him from achieving an unwanted program state with scans hidden below zero ground level. These scans would be invisible.

2.2.2 Visual - physical conflict

By creating constraints to moving objects out of a given space, a user may experience a so called visual-physical conflict. The hand moving the object may go through an area but the application does not apply the physical appropriate result[4]. To avoid or at least lower the impact of this conflict user-feedback can be used to prevent the user from provoking these conflicts by accident, like in the vibrating controller described in the previous chapter.

2.2.3 Feedback design


Feedback is an essential user experience, which should result in information gain about the result or status of a task or object the user interacted with[?]. A use case in our program would be the switching of colours of a scan when its selected or not. A global selection colour (Bright Red) indicates that this object is selected and will be reference point for further actions. Since scan alignment needs at least 2 scans to be practical the application has a 2 step selection: Selecting the first scan changes its colour to bright red. But when the next scan is selected the first one changes its colour to a less brighter red signalling it is the secondary pick. Applying the alignment now starts an ICP which moves the second picked scan (the one that is now active) toward the first one.

2.2.4 Picking and selection process

Working in VR allows the user to reposition scan components as he wishes to every spot. To align them they need to get close together but if they start already close together they do not look ordered and users tend to split them first, to see the individual scans at least before putting them back together. To solve this problem, all scan components start in a line at a fixed height of about 2 meters at one side of the virtual room. Therefore the user can examine the scans separate. To pick a scan naturally the user could just walk to the side of the room and pick them up there. This would force one to repeatedly walk across the virtual room to the one wall and back to the table in order to assemble the scan by picking the components. To increase the efficiency and to decrease the effort to reach the scans, the tool has a build in drag function and a corresponding ranged pick function. This function works as a so called "extender grab"[4] and allows the user via box ray intersection to grab a scan which is out of his personal

space. The hand remote "shoots" a ray from its front direction which now is tested in a Bounding box ray intersection function. If it intersects a component this component is picked. The user now can drag this component along the ray of the remote towards himself or even away. The component is mapped to the hand remote and performs any translation and rotation the remote does too, but never changes in the remotes coordinate system (distance is preserved as long as the user does not change it manually). If there is more than one component in a row that are intersected the one nearest to the user is chosen.

2.2.5 Grouped scans

During the alignment the user may find the correct rotation and translation to align to scans. To save this but also be able to manipulate the scans rotation and translation without destroying the alignment, the user group scans. These groups now work as one scan with a shared Bounding box and shared translation and rotation and colour. If the user tries to pick the scan they both are selected and if he starts an ICP with the selected group and another group, both scans points are used. This may come unhandy when the scans are grouped incorrectly and now need to be separated again. To solve this an extra function was implemented. This function splits a group by translating them towards the averaged normal directions of their individual points.  By using the cross product of their averaged normalized normal vector and a dynamic vector which is generated out of the amount of the amount of components the group has. This is achieved by dividing a circle by the number of scans. The resulting angle is used in a rotation matrix around the z axis. By multiplying the matrix with the (1,0,0) vector x times normalized directions are derived. To avoid the possibility that bigger scans can overlap even after the split the minimal distance needed to translate the scans without overlapping is calculated using 2 restrictions:

1. The maximal Bounding box diagonal is equal to the diameter of a circle around the whole component. To avoid overlapping the point to point distance between the middle points of their bounding boxes has to be at least bigger than the maximal Bounding box Length of the biggest Bounding box
2. The 3 points: averaged middle point of the whole group and the translated Bounding Box middle points of 2 of the scans form a nearly icosceles triangle.

Therefore the minimal translation distance equals (gleichschenkliges dreieck formel). The application uses a factor of 1.1 of this minimal distance to avoid errors due to the fact that the computer works with floats and doubles which are not always precise and that there may be a distance between the averaged middle point of the group and the middle point of the Bounding Boxes of the scans. This is especially the case if the scans contains outliers.

After this splitting translation was executed, the user now can chose a single component out of the group

and separate it. The remaining group turns back to its initial state, the separated scan is lifted above the group.

2.2.6 Scan positioning

When loading a scan into the application there are several problems that occur:

1. Where should the initial position be when there are no information available?
2. What if the available information are invalid e.g. under the $z=0$ level?
3. What if the user wants to reset his actions because there are too many errors and he needs to start again?
4. How should positioning be executed without confusing the user with instant teleportation of the components

To solve this problem several functions are introduced to reposition components in different ways.

The reset function

This function works as a simple reset. The user forces the components to all disassemble and reposition onto the default wall even if their loading position was somewhere other than default. This function is a hard reset even deleting colour and group information.

The position above table function

This function is a repositioning function to reposition a picked group to the middle table in the lowest height available. This height is determined by the lowest corner z -coordinate of the lowest bounding box of the group. Additionally the groups averaged middle point is now in the middle of the table, which is by default the middle of the room.

The loading functions

These special functions are not directly called by the user. They are instead called whenever the user decides to change the workspace by loading a new project or directory loading. A newly created component always causes the program to initialize its transformations unless they have a flag set that prevents the program from overwriting the transformations. These flags are used in project files to signalize that this is a component the user has already worked with and this should not be repositioned.

The reposition execution animation

To avoid instant teleportation, every repositioning that is not handled by directly grabbing a component thus attaching it to the hand controller, is instead animated as a linear blend between the original trans-

lation position and the target translation position. Additionally a second animation executes the rotation (if there is one) at the same time. These linear blend is executed on colours as well, making hard colour switches only available for blinking animations or the picking event. This is especially important to signalize that blinking colours are indicating that the user has to confirm an action. Additionally the user can see what he caused with his actions instead of a rapid state change of the program. When several translations are executed by one action it could be irritating with no fluent transition to the new state.(AUSDRUCK)

2.2.7 Additional application concepts

Some features of computer programs have evolved over the time and it is common to include them to all applications because of their general purpose. For example the possibility to restore the program state before the last action is a well known feature and commonly expected to be in every program. (FIND QUOTE) This feature is implemented by a stack which tracks all program states since the last project loading was called. The inverse function, restoring a state which was after the actual state but has been falsely reversed, is implemented too.(AUSDRUCK!)

10 usability report

- Get users - Get their feedback

2.3 eine Grafik

2.3.1 Etwas Mathe

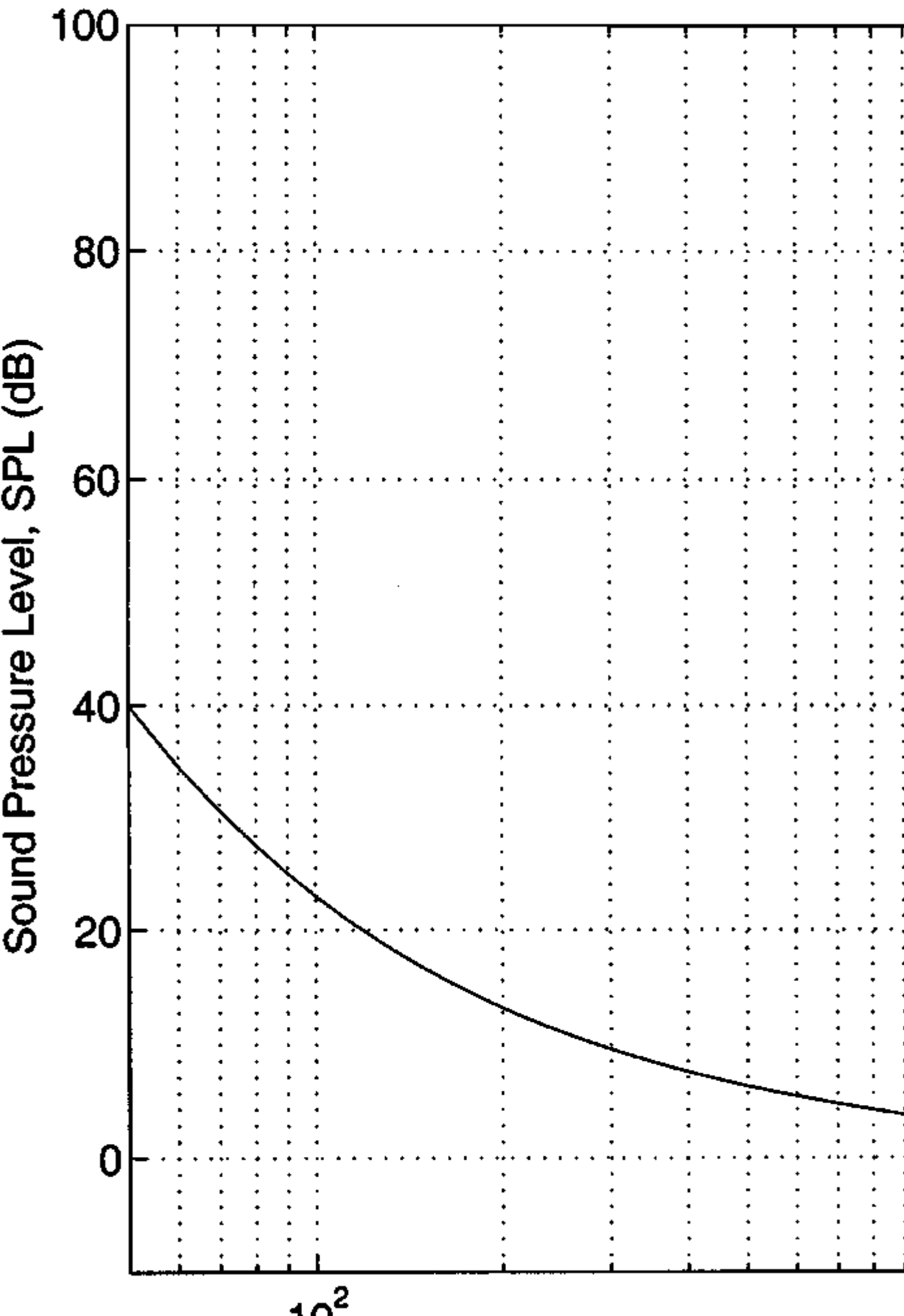
$$\sum_{i=1}^{100} x_i$$

noch mehr text

2.3.1.1 Verweise auf Literatur

etwas quelltext text

[]



```
//comment  
for(int i = 0; i < 100;i++)  
{  
    test(i);  
}
```

Bibliography

- [1] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.
- [2] Chen Chao and Ioannis Chao. Semi-automatic range to range registration: A feature-based method. In *Fifth International Conference on 3-D Digital Imaging and Modeling*. IEEE.
- [3] Vincent Garcia, Eric Debreuve, and Michel Barlaud. Fast k nearest neighbor search using GPU. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, jun 2008.
- [4] Jason Jerald. *The VR Book*. Association for Computing Machinery and Morgan & Claypool Publishers, 2015.
- [5] Shijun Ji, Yongcong Ren, Zhao Ji, Xiaolong Liu, and Gao Hong. An improved method for registration of point cloud. *Optik*, 140:451–458, jul 2017.
- [6] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: a class of displays on the reality-virtuality continuum. In Hari Das, editor, *Telemanipulator and Telepresence Technologies*. SPIE, dec 1995.
- [7] Andreas Nuchter, Kai Lingemann, and Joachim Hertzberg. Cached k-d tree search for ICP algorithms. In *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*. IEEE, aug 2007.
- [8] Neil D. McKay Paul J. Besl. Method for registration of 3-d shapes, 1992.
- [9] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE Comput. Soc, 2001.
- [10] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596, may 2007.

Acknowledgments

Die Danksagung...

Copyright Information

Hier soll jeder Autor die von ihm eingeholten Zustimmungen der Copyright-Besitzer angeben bzw. die in Web Press Rooms angegebenen generellen Konditionen seiner Text- und Bild"ubernahmen zitieren.