Technische Universität Berlin
Fakultät IV Institut für Telekommunikationssysteme
Fachgebiet Nachrichtenübertragung Prof. Dr.-Ing. Thomas Sikora

Master Thesis

CHANGE DETECTION USING MODELS DERIVED FROM POINT CLOUDS

Florian Lehmann
lehmann.florian@posteo.de
Matr.-Nr.: 396038

14. February 2020

1. Evaluator: Prof. Dr.-Ing. Thomas Sikora
2. Evaluator: Prof. Dr. Anatolij Zubow

Thesis supervised by: Dr. Sergey Zuev
German Aerospace Center (DLR)
Institute of Optical Sensor Systems

Dr.-Ing. Sebastian Knorr
Technische Universität Berlin
Institut für Telekommunikationssysteme
Fachgebiet Nachrichtenübertragung

Masterthesis for Florian Lehmann

Change detection using models derived from point clouds

A digital replica of the real world is demanded in a wide variety of industries – from construction to quality assurance and autonomous driving. The increasing number of low-cost sensors for 3D measurements supports the emergence of new applications and research in this field.

The department for real-time data processing (EDP) at the institute for optical sensor systems of the German Aerospace Center in Berlin-Adlershof is developing a multi-sensor approach for 3D measurements called Integrated Positioning System (IPS). The IPS is a system for ego motion estimation without any prior knowledge of the environment. It is capable to operate in buildings, tunnels, etc. as well as outdoor spaces. In its basic configuration, the system combines a low-cost inertial measurement unit with a stereo camera system and can be extended by various sensors such as GPS or a thermal imaging camera. This combination of sensors allows the local referencing of the measurement data and the derivation of a dense point cloud from the captured images.

Various applications require capturing the same scene at different times. Meanwhile, the scene is subject to minor changes while the rough structure remains the same (e.g. moving objects in a room, parking cars on streets). These changes have to be tracked. For an efficient monitoring it is necessary to abstract the point cloud and to transform it into a simpler representation.

(a) Handheld setup of IPS

(b) Point cloud captured by the IPS

In this master thesis an approach for the derivation of a model from a point cloud will be developed. Based on the derived model, changes in the environment are to be detected.

For this goal the following tasks are to be carried out:

- Study of model derivation and simplification methods from point clouds
- Getting familiar with the IPS for 3D Reconstruction
- Implementation of a model derivation from point clouds
- Implementation of an algorithm for change detection in the derived model
- Evaluation of both approaches

DLR-OS
Dr. Sergey Zuev
Email: sergey.zuev@dlr.de

TU Berlin
Dr.-Ing. Sebastian Knorr
Email: knorr@nue.tu-berlin.de

Declaration of Authorship

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

_____

Berlin, the 14. February 2020

Abstract

Change Detection using Models derived from Point Clouds

This thesis examines the detection of geometric changes in 3D data based on models derived from the point clouds. The process chain consists of the registration of point clouds, the derivation of a model and the detection of changes in these models.

For the registration AprilTags are used, which, in combination with the trajectories of the respective measuring runs, allow the estimation of a transformation between two local coordinate systems without a preceding assessment of the tag position.

The point cloud is sampled down to a uniform distribution for faster processing and the normals and curvatures are calculated for the remaining points. A region growing process uses this additional information to divide the point clouds into planar and non-planar areas. The former are modeled as planes through simplification and meshing using a modified quadtree structure. The non-planar points are clustered by supervoxels and the extracted clusters are approximated by Axis Aligned Bounding Boxes. Both planar and non-planar modeling is performed for all datasets which are to be compared. The bounding boxes are used for the detection of changes. The box model of a dataset can easily be examined for intersections with a second dataset, due to the axis alignment. The intersections allow the detection of changed areas in the derived models.

The methods proposed are suitable for both indoor and outdoor applications, provided that the changed objects are well separated and the compared datasets cover overlapping areas. The accuracy depends on the chosen size of the bounding boxes as well as the size of the changed objects. The evaluation has also shown that the proposed method can be integrated into a realtime-capable system.

Keywords: perception, registration, model derivation, change detection, point cloud

---

Zusammenfassung

Change Detection using Models derived from Point Clouds

Diese Arbeit beleuchtet die Erkennung von geometrischen Veränderungen in 3D Punktwolken auf Basis einer vorhergehenden Modellableitung. Die dafür aufgebaute Prozesskette besteht aus der Registrierung von Punktwolken, der Ableitung eines Modells sowie der Erkennung von Änderungen in diesen Modellen.

Für die Registrierung werden AprilTags als künstliche Marker verwendet, die in Kombination mit den Trajektorien der jeweiligen Messläufe die Berechnung einer Transformation zwischen zwei lokalen Koordinatensystemen ohne vorherige Vermessung ermöglichen.

Die Punktwolke wird zur schnelleren Verarbeitung auf eine gleichmäßige Verteilung heruntergerechnet und die Normalen und die Krümmung für die restlichen Punkte berechnet. Ein *Region Growing*-Verfahren nutzt diese Zusatzinformationen, um die Punktwolken in planare und nicht-planare Bereiche zu unterteilen. Erstere werden durch Vereinfachung mit einer modifizierten Quadtree-Struktur als Ebenen modelliert. Die nicht-planaren Punkte werden durch Supervoxel in gleichmäßige Abschnitte unterteilt. Die so extrahierten Cluster lassen sich durch achsparallele Boxmodelle approximieren. Die planare und nicht-planare Modellierung wird für alle zu vergleichenden Datensätze durchgeführt. Die nicht-planaren Bereiche werden für die Detektion von Veränderungen genutzt. Das Box-Modell eines Datensatzes kann aufgrund der Achsausrichtung leicht auf Schnittmengen mit einem zweiten Datensatz untersucht werden. Die Schnittmengen erlauben die Erkennung von veränderten Bereichen im Modell.

Die vorgestellten Verfahren eignen sich sowohl für Innen- als auch für Außenanwendungen, unter der Voraussetzung, dass die geänderten Objekte gut separierbar sind und die verglichenen Datensätze überlappende Bereiche abdecken. Die Genauigkeit hängt von der gewählten Größe der Boxen im Modell sowie der Größe der geänderten Bereiche ab. Die Auswertung hat gezeigt, dass sich die vorgestellte Methode in ein echtzeitfähiges System integrieren lässt.

Keywords: Registrierung, Modellableitung, Veränderungserkennung, Punktwolke

---

Contents

List of Figures

List of Abbreviations

AABB	Axis Aligned Bounding Box
DLR	Deutsches Zentrum für Luft- und Raumfahrt
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IPS	Integrated Positioning System
LiDAR	Light Detection and Ranging
PCA	Principal Component Analysis
QTD	Quad Tree Depth
SIMD	Single Instruction, Multiple Data
SGM	Semi Global Matching

List of Symbols

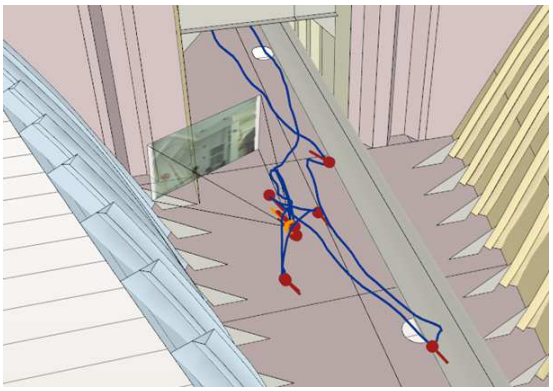
$\mathbf{v}$	Vector
n	Scalar, representing e.g number of neighbors/ dimension
$T_{s,t}$	Transformation matrix from source coordinate system to target system
Γ	Trajectory containing poses with 6 DOF and the corresponding timestamp
P	Point cloud containing points of one measurement run
M	Model containing planar surfaces and modeled non-planar objects
A	List of AprilTag positions and corresponding IDs

1 Introduction

A recent trend in the area of measuring systems is the integration of position information into measurement data. The three-dimensional localization of measurements allows the interpretation of these data in a spatial context [1, p. 40]. While stationary sensors can be surveyed once, mobile systems must be able to determine their position for each measurement. This localization is not trivial and represents a dedicated research area.

The Institute for Optical Sensor Systems at the German Aerospace Center in Berlin-Adlershof is developing a multi-sensor approach for 3D measurements called Integrated Positioning System (IPS). The IPS is a system for ego localization without any prior knowledge of the environment. It is capable to operate in buildings, tunnels, etc. as well as outdoor spaces. By combining pose estimation with other sensors, their measurements can be localized. An extensive field of application that benefits from the localization of measurements is the regular monitoring of technical environments, also known as condition monitoring. A current approach is to combine a helmet-mounted version of the IPS with an RGB inspection camera to take color images in environments difficult to reach such as under ground, or as shown in Figure 1.1 (a) in a hull of a ship [2]. This allows automated position tracking for each image captured by the system so that the image data can be easily located in a reference model as shown in Figure 1.1 (b). This work is based on the data acquisition of the IPS, whose general design and working principle is described in Section 1.1. The functionality is extended in the area of condition monitoring. The motivation and realized objectives are summarized in the Sections 1.2 and 1.3.

(a) Localization of images in 3D model.



(b) Localization of images in 3D model.

Figure 1.1: Condition monitoring of a ship [2].

1.1 Integrated Positioning System

In many use cases, an exact localization of the user within its surrounding is necessary. This can be done for example by using an external reference system like Global Positioning System (GPS) that uses the signal send by satellites to triangulate the users position. This has an accuracy of a few meters up to centimeters when using a differential GPS [3]. In any case, the use of this system is restricted to applications where the GPS signal can be received. For cases in which visibility to the satellites cannot be guaranteed, or external referencing is not possible from the outset, local navigation solutions are an option for position determination.

The Integrated Positioning System (IPS) combines a stereo camera setup with an Inertial Measurement Unit (IMU) to a multisensor solution for local pose estimation. The system can be freely configured regarding the number and type of cameras and the IMU as well as their geometrical relative orientation. By choosing the right hardware and setup the system can easily be adapted to a wide variety of use cases from very small systems mounted on robots, up to big setups measuring buildings. It can be extended by other sensors such as, for example, thermal cameras, laser measuring sensors, etc. In the following Figure 1.2 shows one example of the IPS used for handheld indoor applications and one for car mounted outdoor purposes. The configuration shown in the left Figure 1.2 (a) consists of an IMU inside the case, two infrared LED lights and two cameras, mounted at a distance of 20 cm. The relative orientations that describe the structural arrangements of the components, as well as their recording parameters, are known through calibration.

(a) Handheld version for indoor applications . (b) Car mounted IPS.

Figure 1.2: Two different IPS-setups (image ownership by DLR).

The accelerometer and gyroscope of the IMU allow to estimate the path, went between two points in time, by double integrating the measured acceleration and angular velocity in three degrees of freedom each. The combination of time stamps and according poses in 3D space is called trajectory. Each pose describes the position and orientation of the system with combined six degrees of freedom in a coordinate system with its point of origin at the starting point of the measurement. This estimation procedure is called inertial odometry. No external signals are necessary to derive a trajectory but the poses are not always certain since the sensors are noise sensitive and error prone. The position is for example drifting over time and accumulates a scale error. To validate the measurements a stereo camera system is capturing the scene continuously. The captured stereo images are analyzed to derive a relative transformation between two time stamps in a so called visual odometry. The combination of inertial and visual odometry results in a visual aided inertial odometry where the measurements of both systems are taken into account. This combination outperforms the single solutions.

Recent developments allow the derivation of a point cloud from the captured stereo images and trajectory. Point clouds consist of punctual spatial measurements without volume. These can also be derived for example using a 3D laser scanner system. Due to the high portability and low cost hardware the IPS represents a more mobile and affordable alternative. The IPS additionally provides a radiometric measurement for each point, so that the point cloud is colored based on the measured intensity values. An example can be seen in Figure 1.3 where the IPS depicted in Figure 1.1 (a) has been carried through a ships hull recording the trajectory and a point cloud.

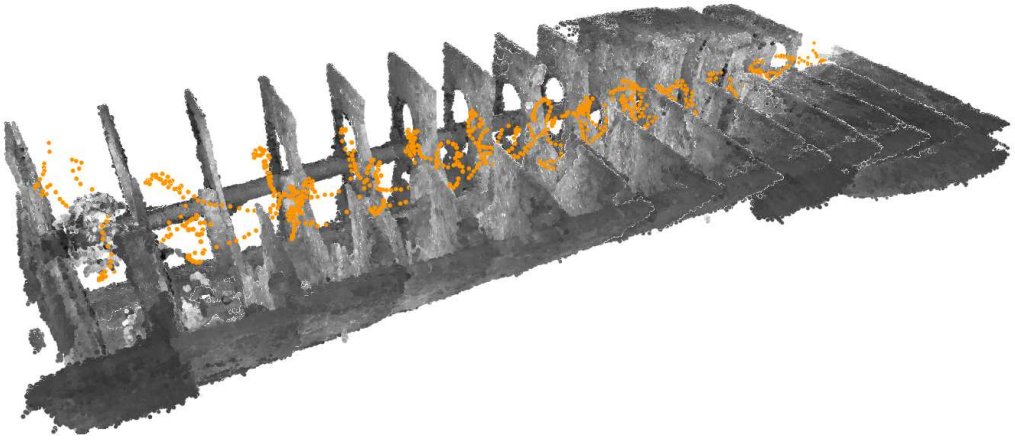
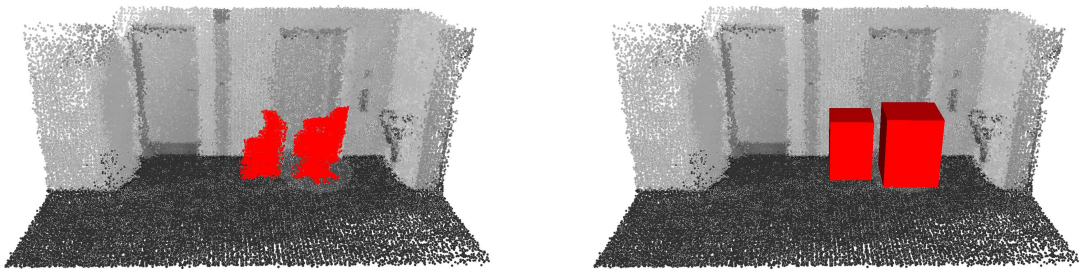


Figure 1.3: Point cloud of a ships hull captured with the IPS. Trajectory indicated by orange points (point cloud ownership by DLR).

1.2 Motivation

Monitoring environments investigates changes that occur over time. For this reason, repeated measurements in the same environment at different times are necessary. The comparison of the measurements indicates the change and is referred to as *Change Detection*. This is an important research area of remote sensing in which, among other things, the change due to urbanization, deformation, land use, etc. is monitored in large scale coverages and long-term intervals [4]. The monitoring of smaller spatial changes in shorter time periods over hours, days or weeks is also worth considering. In various local application areas, such as construction sites, it is important for the planning to include information about changes to a previous state. One way to do this is using the IPS to capture point clouds at different times in a to be monitored environment. The comparison of the gathered three-dimensional data points allows to detect geometric changes over time as shown in Figure 1.4.



(a) Monitored indoor environment. (b) Detected and modeled change.

Figure 1.4: Change detection in an indoor environment. Change indicated in red.

The point clouds of the IPS are dense due to the high resolution of the cameras and the possibility to scan locations several times during a recording session. When comparing two dense point clouds, many points have to be compared. In addition, point clouds are difficult to visualize because no volumetric information is available. Also, not all applications support the visualization of point clouds at all. With regard to the fields of application of the IPS, a fast processing of the results is intended as well as a good visualization. In many applications the surfaces of objects are reconstructed by a mesh structure, as shown in Figure 1.5, using the respective point clouds. These structures can be simplified with the help of known methods from the wide research area of computer graphics [5]. In addition, almost all 3D visualization tools support 3D mesh formats. This determination of a model from a point cloud is called *Model Derivation*.

(a) Point cloud of a fire extinguisher. (50000 Points)

(b) Mesh representation of the same object. (1600 points and 2500 triangles)

Figure 1.5: Two 3-dimensional representations of a fire extinguisher (adapted from [6]).

The chosen representation can also affect the change detection, since not the entire point cloud is used for comparison, but the simplified mesh structure. By deriving a reasonable model, the change detection is simplified and accelerated. This thesis presents an approach for change detection using a 3D model derived from point clouds. The main focus is on applications in indoor environments with geometric changes between a few decimeters and a few meters.

In order to compare or combine 3-dimensional entities spatially they must be within the same frame of reference. Each point cloud captured by the IPS is defined in a particular coordinate system with its origin at the starting point of the measurement. When the starting pose changes, the captured points appear at different positions. As seen in Figure 1.6 the overlaid point clouds occupy different regions in the coordinate system despite representing the same scene. The process of transforming the coordinate systems into each other is called *Registration*.

Figure 1.6: Two not registered point clouds capturing one scene, differing by a rotation.

1.3 Objectives

In the context of this work, two important areas in the development of the IPS have been pursued. On the one hand, a method to derive 3D meshes from the captured point clouds was implemented within the IPS-Framework for the first time. On the other hand, the proposed comparison method can be used to analyze 3-dimensional scenes for spatial changes on the basis of the derived models. Both approaches open up completely new fields of applications. A registration method based on the IPS capabilities of detection AprilTags is described, but alternatives are not investigated, as this is not the focus of this work. In the following a summary of the contributions of this work to the development of the IPS is stated:

- design and implementation of a dedicated local registration application based on AprilTags
- implementation of a planar simplification and meshing algorithm
- design and implementation of a new change detection method
- four new datasets are recorded with different configurations
- evaluation of the stated approaches

These stated objectives are described in this thesis. For the fundamental understanding Chapter 2 introduces and explains the theoretical principles and technical terms this work depends on. In Chapter 3 other works are discussed which address the same topic and the proposes methods are put into relation. Chapter 4 addresses the implemented methods in detail, whereas Chapter 5 evaluates them based on different datasets regading their accuracy and runtime. Finally, Chapter 6 draws a conclusion and gives an outlook to possible future works.

2 Fundamentals

The methods presented in this thesis are based on various methods from the research areas of computer vision, computer graphics and data analysis. This chapter describes these methods in detail. First, several data structures are explained in Section 2.1 that enable efficient access to multidimensional data. This data can be available in many different forms. Section 2.2 describes the 3D representations used in this work, such as point clouds and mesh structures. Furthermore, procedures for processing these data types are explained. The last Section 2.3 states the view geometry of monocular and stereo cameras and the underlying mathematical model as well as procedures for reconstructing depth information from the obtained image data.

2.1 Space Partitioning

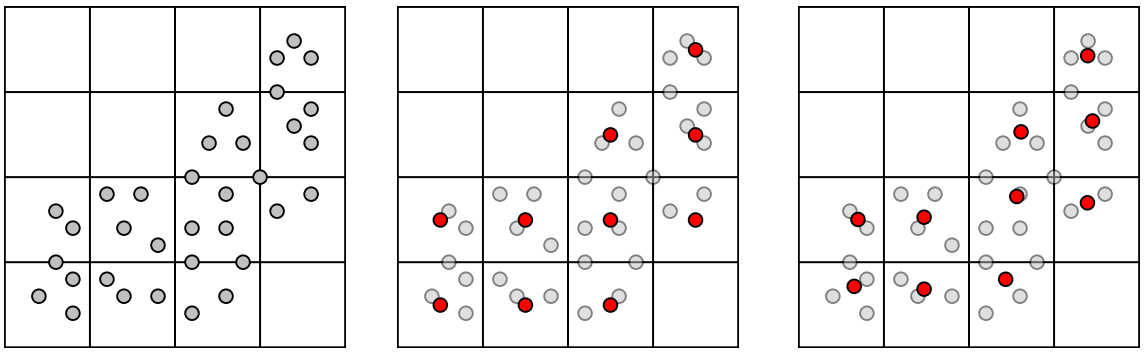
In their simplest form, multidimensional data are sets of coordinates in a multidimensional space. These data points can be annotated with information such as color information, labels and measurements. A set of data points often consists of several hundred thousand data points. Traversing an entire point set can easily become computationally intensive. In particular, the pairwise comparison, for example when searching for nearest neighbors, is very time-consuming. Spatial partitioning techniques provide efficient processing by dividing the space into regions and testing if points are within the same region in this space [7]. This reduces the number of pairwise comparisons drastically. This section introduces efficient representations for multidimensional datasets that allow fast access or include neighborhood information. The different spatial partitioning techniques Quad- and Octree, as well as k-d tree and voxel grid are explained. The descriptions are based on [7]. These techniques are not necessarily restricted to be used in processing visual data, but can also be used in other fields that rely on data analysis. In the following section, the description is limited to the use of 2- and 3-dimensional point sets, which are referred to as point clouds.

2.1.1 Voxel Grid

A dataset containing many points has no information on spatial relations, so that processing points in a certain region requires iterating over all points in the set. A simple solution is to precompute some spatial information in a appropriate data structure. An efficient way to achieve a uniform resolution is to use voxel grids [7, p. 285]. A grid divides the considered space into a number of regions or cells of equal size as shown in Figure 2.1 (a). These cells relate to volume elements and thus are called *Voxels* in 3D, in the same manner as *Pixels* represent picture elements in 2D. The data structure of the voxel grid contains information about the dimensions and position of the respective cell as well as the points it contains. This uniformity allows fast access to point sets within a certain region. Instead of iterating over all points, only the individual cells have to be checked..

The efficiency of a voxel grid is depending on its resolution, defining the cell size in comparison to the overall point set size and its complexity. An incorrectly selected cell size can cause performance issues. If the grid is to fine, many cells have to be updated with the information on the contained points which will take more computation time and space. In the worst case the voxel grid contains more cells than underlying points. A grid being to coarse leads to over simplification so that still many points have to be accessed.

The point density in point clouds can vary within one dataset. This can be caused by redundant measurements and can interfere with further processing. Another common application of voxel grids is the filtering of point clouds to have a certain density. Therefore, a voxel grid of the desired resolution is computed and the center of each non-empty cell used as new point. This procedure is shown in Figure 2.1 (b). For higher accuracy the mean position of points in the voxel can be used as shown in Figure 2.1 (c).



(a) Voxel grid representation.

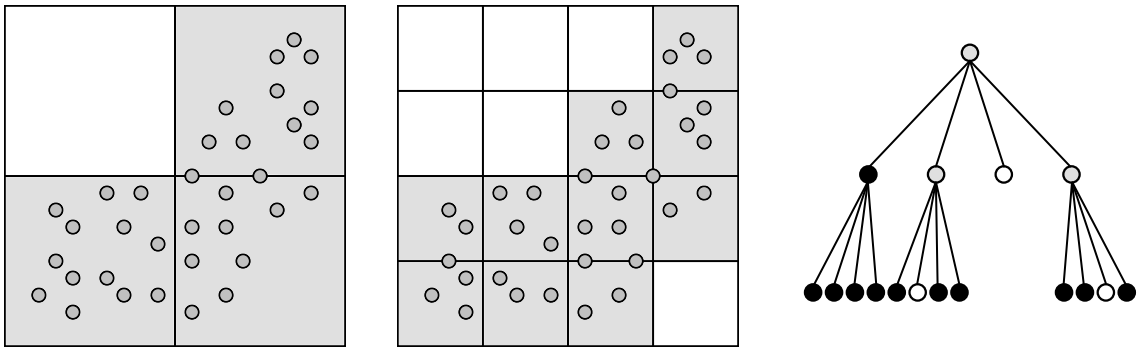
(b) Filtered point cloud.

(c) Center approximation.

Figure 2.1: Voxel grid representation of a 2D point cloud.

2.1.2 Quad- and Octree

Uniform grids cannot properly handle point sets of different sizes and distributions due to their fixed dimensions. Sparsely populated point clouds, for which the cell size is chosen too small, require a lot of time for processing, since many empty cells have to be checked. By computing a hierarchical grid with varying cell size in each level, this problem can be addressed. One way to combine a hierarchical grid with a tree-like structure that is easy to traverse is the *Quadtree*. The quadtree represents an axis-aligned hierarchical partitioning of a 2-dimensional space as pictured in Figure 2.2. A tree consists of its *Root* representing the starting point, *Child nodes* dividing the space into different parts and *Leaves* containing the data points. The root is located in the origin of the coordinate system of the point cloud and has four children. Each child covers one quadrant of the space the local coordinate system the parent node covers and has itself up to four children. Additionally, each child contains it center point, respectively information on its dimensions and information on the usage of its children (fully, partly, not used). A child being at the end of the tree structure is called leaf and holds the points fed into the quadtree. When building a quadtree, the extents of the depicted point cloud and the desired depth of the tree or a minimum cell size has to be known. The depth describes the number of levels containing children. At each level, the space is split in half along the x and y axis. The children contain the resulting parts. This step is repeated for each child until the desired depth is reached, as shown in Figure 2.2 (a) and 2.2 (b). The children of the last level then contain the points of the point cloud. This principle is not limited to the 2-dimensional space and can be used for any n-dimensional space. In the 3-dimensional case, it is called *Octree* subdividing the space in eight octants along the x, y and z axis.



(a) First level of quadtree. (b) Second level of quadtree. (c) Resulting tree structure.

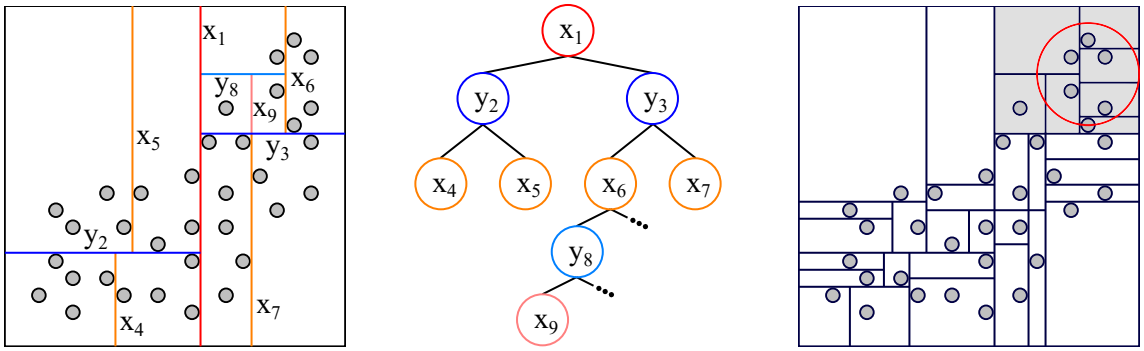
Figure 2.2: Quadtree build from a 2D point cloud.

2.1.3 *k*-d Tree

A more general tree-based representation is the *k*-dimensional tree or *k-d tree* introduced in [8]. While oc- and quadtrees divide the space in all dimensions at once, a *k*-d tree divides along one dimension at a time. These splits do not necessarily have to be uniform. The prefix *k* specifies the number of dimensions the tree subdivides. The tree dimensionality does not necessarily have to match the dimensionality of the divided space. This can be useful if a dataset has only a very small spread in a certain dimension. One level represented in an octree can be seen as three level *k*d-tree split along the x, y and z axis. Splitting along one axis instead of two or three can be implemented more easily and is numerically robust [7, p. 319].

The tree structure consists of nodes with two child nodes each, which contain the axis and value dividing the point set. The leaf nodes contain the data points. In each division step the point set is split in half along one particular axis. Often the division is in a round-robin manner by switching between dimensions in a circular fashion. The resulting tree is mostly not balanced. This subdivision scheme and parts of the corresponding tree based on an example are depicted in Figure 2.3 (a) and 2.3 (b). Alternatively the order also can be freely chosen for example by dividing the axis with the widest spread, to form more balanced trees.

Nearest neighbor searches and range searches can easily be done with the help of *k*-d trees. When searching for points within a certain range only those tree nodes have to be queried that the search region overlaps as shown in Figure 2.3 (c). This reduces processing time for common problems as for example normal estimation where neighbors of each point in a point set are needed.



(a) Subdividing scheme. (b) Tree structure building. (c) Range query in a *k*-d tree.

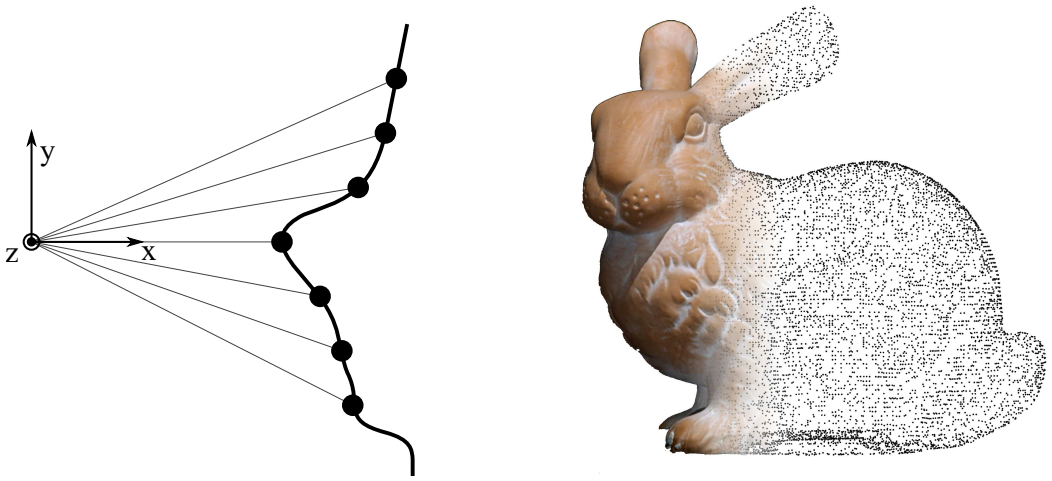
Figure 2.3: *k*-d tree build from a 2D point cloud.

2.2 Representations of 3D Data

The location of measurements in space and the processing and derivation of the resulting data is a common problem in computer vision and computer graphics. In this thesis the focus is on 3-dimensional data that can be represented, processed and stored in various forms. In this scope, point clouds, one of the most simple representations, as well as triangle meshes are introduced in Section 2.2.1 and 2.2.4. Furthermore, different processing steps for point clouds are required in this work. Section 2.2.3 explains the concept of alpha shapes for the extraction of concave hulls and Section 2.2.2 introduces methods for normal estimation in point clouds.

2.2.1 Point Cloud

A large variety of sensors and sensor types provides the acquisition of spatial information. Laser scanners use a beam of light to scan the surroundings for example point by point using Light Detection and Ranging (LiDAR). This system is an active method in which a single laser beam is directed at a point and the propagation time is measured. There are many other methods for example RGBD cameras that capture color and depth information using structured light for distance measurement or completely passive systems such as the IPS utilizing multiple cameras and the underlying stereo geometry. All these techniques allow the perception of the surrounding environment by creating point-wise measurements defined by 3-dimensional coordinates as pictured in Figure 2.4 (a).



(a) Point cloud from distance measurement. (b) Point cloud of a scanned figure [9].

Figure 2.4: Examples for point clouds.

A point cloud is the spatial representation of these measurements in one coordinate system. Depending on the sensor the data points in 3D can be augmented with further information like illumination, color, roughness and other material properties. An example can be seen in Figure 2.4 (b) that shows an image of the Stanford bunny figure [9] combined with the respective point cloud.

Multiple point clouds can be combined in one frame of reference and for example georeferenced by registering each point cloud in the respective coordinate system. By combining different approaches like remote sensing based, UAV based and hand-held systems large environments can be captured in detail [10]. These combined point clouds are unorganized point sets with no knowledge about the capturing situation. Depth sensors like Kinect and other RGBD sensors provide organized point clouds. The captured points consists of the 3D information in both cases but in organized point clouds the memory is organized in a 2D array. The processing of adjacent points can be simplified by these additional information [11]. Algorithms that are designed for unorganized point clouds usually work on organized but not the other way around. This thesis completely depends on unorganized point clouds.

2.2.2 Surface Normal and Curvature Estimation

During the acquisition process of point clouds, the information about the surface structure at the respective points is lost, since these are exclusively point measurements not taking any spatial relations into account. The structure of the underlying surface can be described by a normal vector and a curvature (see Figure 2.5 (a)). At each point in the point cloud a tangential plane can be defined. The normal vector is a normalized vector standing perpendicular to this plane. The curvature describes the surface shape at the respective point. The analysis of adjacent points in a point cloud allows the approximation of this information and thus the estimation of the real surface.

In the following a simple normal and curvature estimation method is described on the basis of [12]. The proposed method is based on a first order 3-dimensional plane fitting within a certain set of neighboring points $\mathcal{P}^k$ around each point of a point cloud. k indicates the number of neighbors. This plane is defined by a point $\mathbf{x}$ and a normal vector $\mathbf{n}$. The distance from a point $p_i \in \mathcal{P}^k$ to the plane is define as $d_i = (p_i - \mathbf{x}) \cdot \mathbf{n}$. The estimation can be summarized in a least square problem within the neighborhood $\mathcal{P}^k$. The point $\mathbf{x}$ and the vector $\mathbf{n}$ are computed in a least square sense so that $d_i = 0$. The estimation of

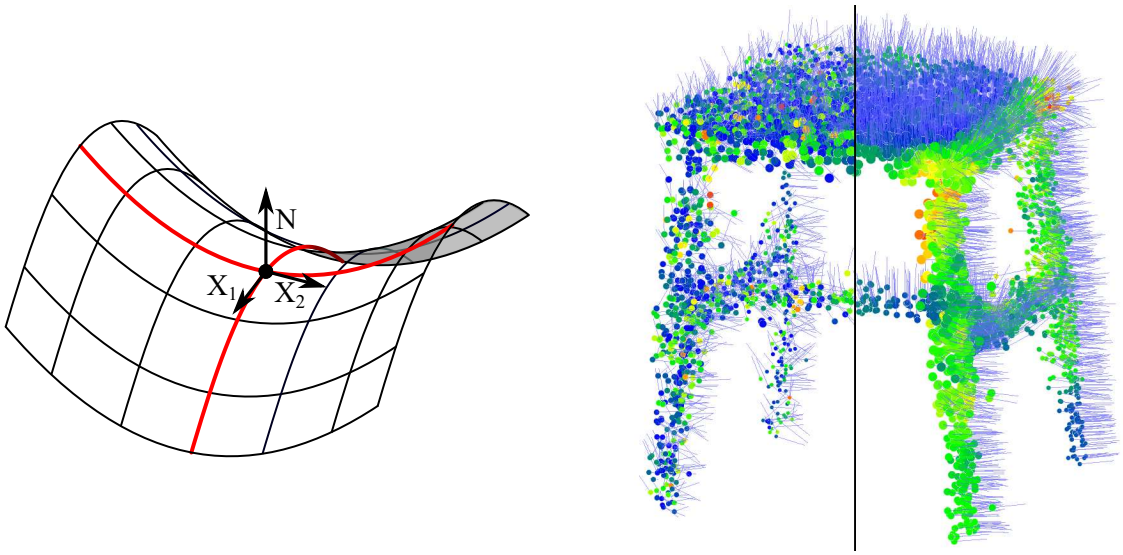
the centroid $\mathbf{x}$ of $\mathcal{P}^k$ is done by:

$$\mathbf{x} = \bar{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \tag{2.1}$$

Normal vector $\mathbf{n}$ can be estimated by analyzing the eigenvalues λ_j and eigenvectors $\mathbf{v}_j$ of the covariance matrix $C \in \mathbb{R}^{3\times3}$ of $\mathcal{P}^k$. The covariance matrix is expressed as:

$$\begin{aligned} C &= \frac{1}{k} \sum_{i=1}^{k} (p_i - \bar{p}) \cdot (p_i - \bar{p})^T \\ C \cdot \mathbf{v}_j &= \lambda_j \cdot \mathbf{v}_j, j \in \{1, 2, 3\} \end{aligned} \tag{2.2}$$

The eigenvectors of this matrix span a vector space corresponding to the principal components of $\mathcal{P}^k$ with its origin in the calculated centroid $\mathbf{x}$. By ordering the eigenvalues and eigenvectors according to their size so that $\lambda_1 < \lambda_2 < \lambda_3$, the eigenvector $\mathbf{v}_1$, corresponding to the smallest eigenvalue λ_1, represents the approximation of the normal vector $\mathbf{n}$. The sign of the derived vector is unknown. By adding an additional viewpoint $\mathbf{v}_p$ that is given by the position of the capturing system this direction can be set. For this the constraint $n_i \cdot (v_p - p_i) > 0$ must be fulfilled. When working with point clouds containing different viewpoints this constraint is not applicable.



(a) Normal vector and curvature describing a surface.

(b) Influence of the chosen neighborhood size. Colors indicate curvature, lines normals. left $k = 5$, right $k = 50$

Figure 2.5: Normals and curvature estimation.

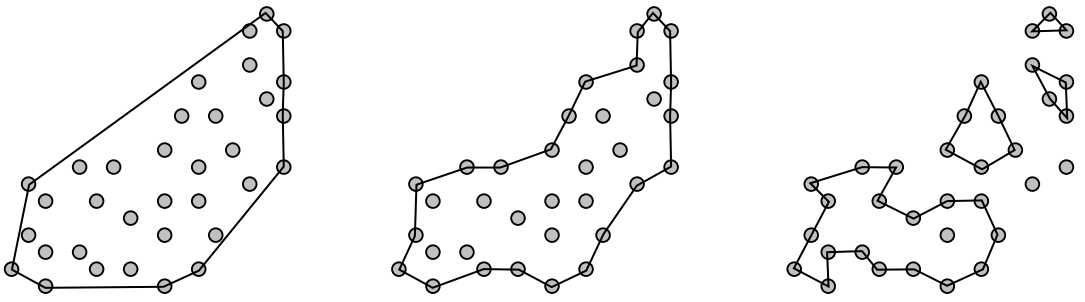
The computed eigenvalues derived in the normal estimation can also be used estimate the curvature σ_{p_i} of the neighborhood $\mathcal{P}^k$ [13]. If $\lambda_1 = \min_j(\lambda_j)$ the curvature can be estimated as:

$$\sigma_{p_i} = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2.3}$$

The curvature as well as the normal vector highly depend on the chosen neighborhood $\mathcal{P}^k$. The neighborhood can be set on the one hand by defining a fixed k representing the k nearest neighbors or on the other hand by defining a certain search radius where k is changing dynamically depending on the number of points within the radius. In both cases, a k-d tree as represented in 2.1.3 can be used to access the adjacent points efficiently. Setting the used neighborhood too large in comparison to the point set leads to a loss of details close to edges. In turn, a small neighborhood makes the approximation of the normals susceptible to noise. An example can be seen in Figure 2.5 (b). In general, both the method and the neighborhood size have to be chosen carefully.

2.2.3 Alpha Shapes

For point clouds, an interior and an exterior region can be defined. The transition between the two regions is represented by the connected points of the boundary. There are different ways to deduce the boundary. The convex hull encloses the entire point cloud like a rubber band stretched around the points as seen in Figure 2.6 (a). This hull is unambiguous, but indentations are not modeled. For a more detailed modeling of the boundary a concave hull is required. However, this is not unambiguous since the level of detail can vary. [14] describes a method to extract a concave hull called alpha shapes. Alpha defines the radius



(a) Convex Hull at $\alpha \to \infty$.

(b) Alpha shape resulting in a concave hull.

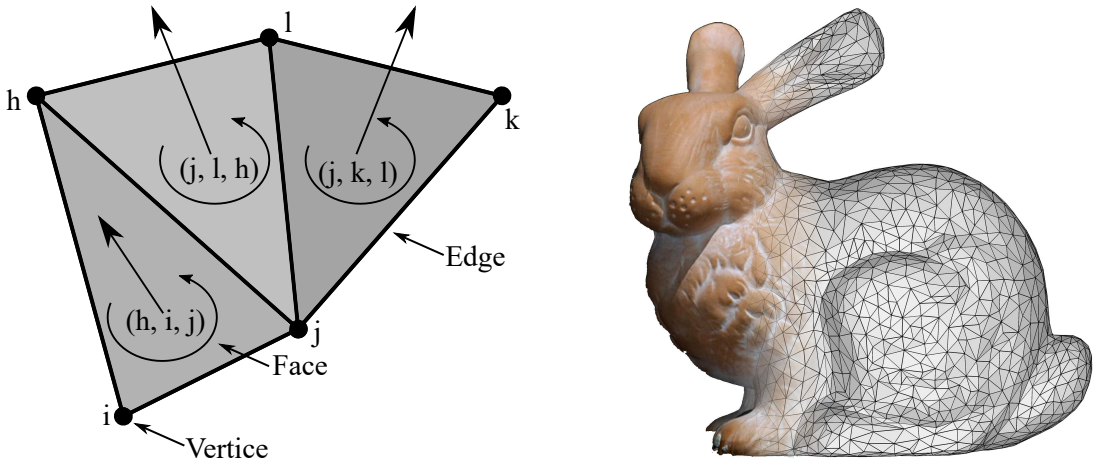
(c) Alpha shape resulting in subdivided regions.

Figure 2.6: Different hulls of a point set (Inspired by [14]).

of a circle. This circle is moved around the point set, approaching the points from each side. If the circle is small enough a more detailed boundary is extracted. A too small α leads to degenerated cases as exemplary depicted in Figure 2.6 (c). The method can be extended to 3D space by using a sphere to distinguish interior and exterior regions.

2.2.4 Triangle Mesh

The points of a point cloud are measurements sampled from surfaces. Each point represents one particular position on this surface and an infinitely dense point cloud would completely reconstruct this surface. By approximation similar patches on the surface as closed 2-dimensional envelops, the surface can also be modeled. One well known way to represent surfaces is the use of meshes build up from triangles. These so called *Triangle Meshes* are a fundamental structure in computer graphics [15] and belong to the boundary representations of different methods to describe geometric models. Boundary representations only describe the surfaces of objects and do not make any statements about the enclosed volume. Most shapes can be represented with triangles, as pictured exemplarily in 2.7 (b). More complex shapes can be approximated by subdividing them into smaller triangles.



(a) 2D mesh with normals (Inspired by [15]). (b) Boundary mesh of a scanned figure [9].

Figure 2.7: Examples for 2D and 3D triangle meshes.

Triangle meshes consist of many triangles that are joined along their edges to form a surface. Each triangle is defined by three points, called *Vertices*. Theses points are

connected by three edges spanning a surface. This surface lies in a plane and is referred to as *face* (see Figure 2.7). Furthermore, the vertices, edges and faces can be annotated with additional information that allow a realistic representation. Vertices can get color information that can be interpolated to colorize the spanned face. If a higher resolution without the creation of new triangles is necessary, texture coordinates can be appended to enable the projection of images onto the face. For faces and vertices normals are used to define the interior and the exterior. Different shading techniques utilize the normals to create more realistic visualizations of models. Face normals can be implicitly defined by the order of the spanning vertices. In case of Figure 2.7 the faces are defined counter-clockwise.

In comparison to other polygons, like for example quads that consists of four points, the face of a triangle is always well defined and thus unique. Triangle meshes are widely used in computer graphics due to their uniformity. This uniformity allows various simple operations that makes them particularly suitable for Single Instruction, Multiple Data (SIMD) based graphic pipelines as implemented in modern graphic cards [16].

2.3 Data Acquisition

Point clouds can be obtained by different methods of depth estimation. Active methods such as LiDAR measuring point by point or RGBD cameras that capture a 2D matrix at one point in time benefit from controlled conditions by the emitted signal. These methods are usually very accurate, but active sensors also have their drawbacks in terms of applicability. For example, infrared based methods are limited to applications that are not exposed to the influence of external infrared light sources like the sun.

Another approach in depth estimation coming from the photogrammetry is structure from stereo. In this field the images of two or more cameras are analyzed to derive spatial information on the captured scene. The use of passive cameras does not require an active system, but the system has to adapt more to the environment. The navigation system IPS as described in 1.1 uses two cameras permanently mounted in one setup to obtain spatial information over time.

In the following the theoretical background for point cloud acquisition from stereo cameras is given. Starting with introduction to the camera model in Section 2.3.1, the combination of cameras in a stereo setup in Section 2.3.2 and the final depth estimation in Section 2.3.3 the point cloud derivation from one particular viewpoint is explained. The

combination of point clouds from different viewpoints is summarized in Section 2.3.4. Additionally, AprilTags, that are used to register multiple point clouds captured from one scene, are introduced in Section 2.3.5.

2.3.1 Camera model

To extract information about the captured scene in an image, like detecting salient points and estimating their distance to the camera, a mathematical model of the projection and capturing process is necessary. The pinhole camera model is a widely used model to describe a projective geometry and the capturing parameter of a camera. The working principle of the *Camera Obscura*, called pinhole camera, gave the model its name. It assumes a dot-sized aperture collimating the light rays coming from an illuminated object as shown in Figure 2.8. It allows the mathematically correct projection of a point $\mathbf{M}$ in 3D space to a point $\mathbf{m}$ on a 2-dimensional image plane as the sensor of a camera is.

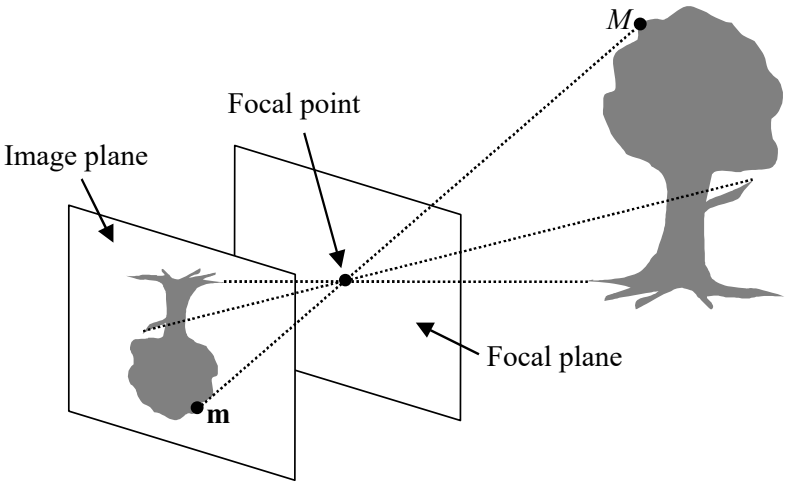


Figure 2.8: Projection with a pinhole camera by [17, p. 41].

All calculations required for the transformation of coordinates or the projection are performed with the help of homogeneous coordinates. Therefore, all coordinates are extended by a scaling factor that is set to 1. The object coordinates are thus 4-dimensional vectors and the resulting image coordinates 3-dimensional vectors. The results can be easily retrieved by scaling the vectors so that the additional dimension is 1 again.

The 3D point M in 3D space is projected onto the image plane via the focal point C, which represents the optical center of the system. The resulting point m in the image plane is mirrored symmetrically to the focal point. For better illustration, the image plane

can also be displayed unreflected in front of the focal point. This results in the following schematic representation of a camera (see Figure 2.9).

A normal, spanned between focal point and image plane, has its point of intersection with the image plane in point c. This point is called the principal point and is the origin of the sensor coordinate system. To transform this coordinate system into the image coordinate system, with the origin in one of the image corners, a translation has to be applied.

Usually, the captured points are given in their own frame of reference O. The camera is also set within this coordinate system. Its position and orientation is defined by a 3×3 rotation matrix R and a 3×1 translation vector t. Both combined create a 4×4-Matrix representing the so called exterior orientation. The camera model assumes all object points to be in the camera coordinate system with its origin in C, the X and Y axis parallel to the corresponding image plane and the Z axis pointing towards c. To transform all object coordinates into the camera coordinate system, the exterior orientation has to be applied to all 3D points:

$$M_C = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0^T & 1 \end{bmatrix} M_O \tag{2.4}$$

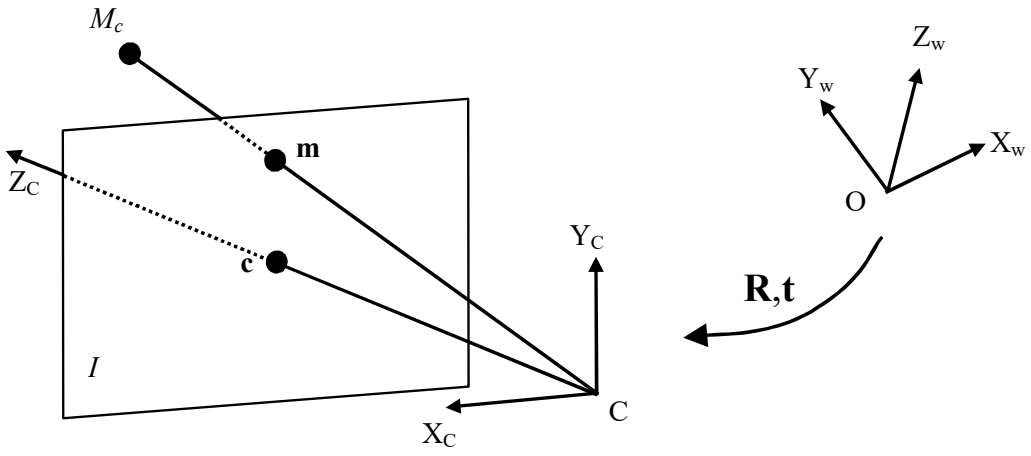


Figure 2.9: Abstract representation of a camera by [17, p. 42].

Assuming all object points to be in the camera coordinate system the projection from 3D to 2D is modeled by the following matrix multiplication of the matrix P and the point coordinate M:

$$\lambda \cdot \mathbf{m} = P \cdot \mathbf{M} \tag{2.5}$$

λ represents a scale factor and P a projection matrix. P can be denoted into a projective part transforming object coordinates to the image plane and a part representing the interior orientation of the camera. The interior orientation transforms the projected points into the coordinate system of the image.

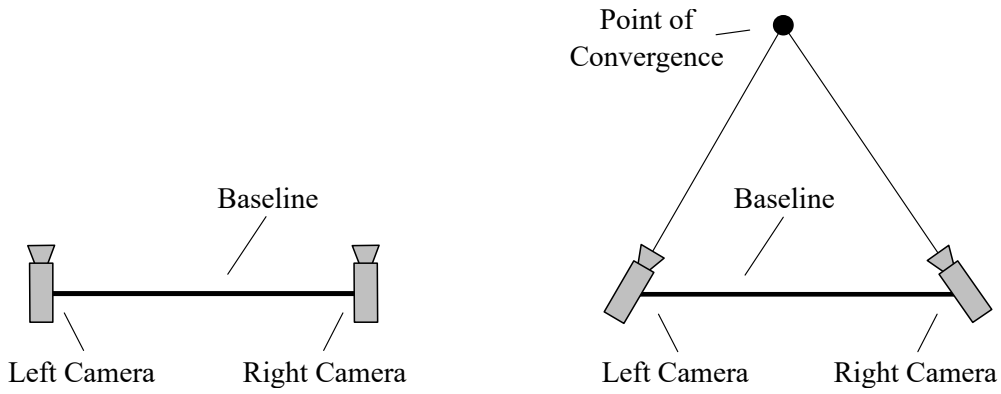
$$P = K \cdot P_N = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \text{with } \alpha = \frac{f}{\delta} \tag{2.6}$$

f is the focal length of the optical system in meters. The factor δ is a scale factor, denoting the pixel size in meter per pixel, that transforms the sensor coordinates in meters into image coordinates in pixels. Finally, the projected image is translated by the value of the principal point u_0, v_0. This information can be obtained by calibrating the camera system. By applying the projection matrix to the object points in the 3D space, the Z-dimension is not considered anymore, so the depth information gets lost. It is not possible to retrieve this information directly from the image without additional information.

2.3.2 Stereo Vision

The usage of multiple cameras and the knowledge on their relative orientation allows the estimation of spatial information. Camera systems consisting of two cameras that are positioned relative to each other are called stereo systems [18]. Based on their arrangement stereo systems are classified in two categories [17], as shown in Figure 2.10. The most simple setup, regarding their mathematical properties, is the parallel arrangement as shown in Figure 2.10 (a). This arrangement is called stereo normal case. The cameras are positioned with the same orientation and only shifted along one axis. In most cases the cameras are somehow tilted. If their the viewing direction is directed towards each other as shown in Figure 2.10 (b) the setup is called convergent camera geometry. This is also called the general stereo geometry.

In order to be able to derive statements about the structure of the captured scene, an object point that is visible in one camera must also be visible in the other. Since the points can lie anywhere in the image depending on their relative position, corresponding pairs of points have to be found first. Regardless of their respective arrangement, the difficulty with stereo systems lies in this assignment of corresponding pixels. An object point M that is captured by both cameras is mapped to a pair of corresponding image points m_1



(a) Parallel stereo setup. (b) Convergent stereo setup.

Figure 2.10: Relative camera geometries for stereo setups (Adapted from [17, p. 66]).

and m_2. The projection is based on the projective geometry described in Section 2.3.1. In the general case the capturing geometry can be pictured as in Figure 2.11.

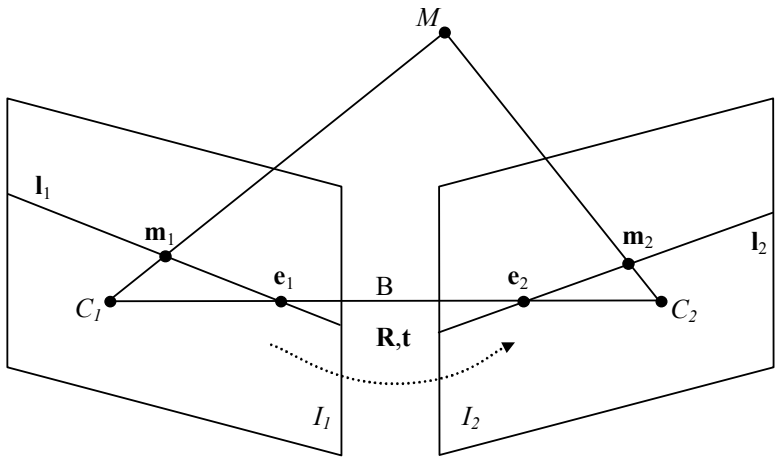


Figure 2.11: General stereo geometry including epipolar geometry [17, p. 67].

In the case of stereo vision the relative orientation of both cameras is fixed and thus can be extracted beforehand. This can be done for example in a previous calibration. With those information each camera can be described by its pose in 3D space consisting of the focal point or camera center C, a view direction and the regarding focal length f, defining the image plane I. The distance between both camera centers C_1 and C_2 is called *baseline*. This line intersects both image planes I_1 and I_2 in the two points e_1 and e_2 called epipoles. Due to the fixed arrangement and the geometry defined with it, so-called epipolar lines can be defined. Each image point m_1 in image I_1 has a corresponding epipolar line l_2 in image I_2. An object point M that is projected to the respective image points m_1 is also

projected to an image point m_2 somewhere on the epipolar line l_2. When searching for corresponding image points only those lines have to be taken into account.

When using a parallel stereo system, without any distortions, the epipolar lines are on the same horizontal line as the corresponding pixel. Finding corresponding point pairs is thus reduced to scanning a line. To take advantage of this, convergent stereo systems can be converted into parallel ones by rectification. Knowing the relative orientation between both cameras the rectification transforms both images I_1 and I_2 to a common image plane parallel to the baseline. Rectified images do not only simplify the correspondence problem, they also allow the calculation of depth information as shown in Figure 2.12.

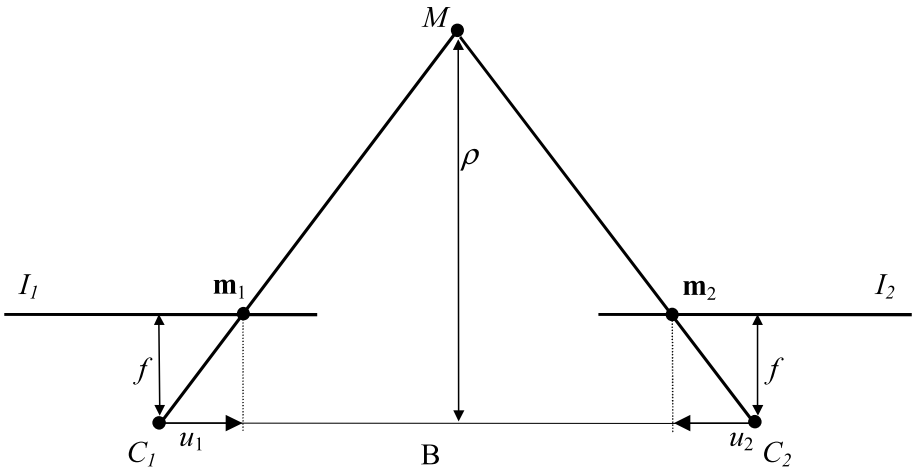


Figure 2.12: Distance estimation in a stereo normal setup.

In a stereo normal setup the distance ρ of the object point M to the baseline can be calculated by:

$$\rho = \frac{f \cdot B}{\delta \cdot d_u} \tag{2.7}$$

The disparity δ describes the distance between the pixel coordinate of the point in the left image to the pixel coordinate of the corresponding point in the right image. It is calculated by:

$$\delta = |u_1 - u_2| \tag{2.8}$$

f represents the focal length, B the length of the baseline and d_u pixel size of the used camera. It is assumed that both cameras have an equal internal parameters.

2.3.3 Semi-Global Matching

Stereo images allow the estimation of depth by finding corresponding pixel pairs in the images of the left and right camera. The first step in depth estimation is the derivation of disparity values for each image point, combined in a so called *disparity map* as seen in 2.13 (c). Estimating a dense disparity map can be a challenging task due to the high amount of pixels that have to be compared. Especially in real time systems like the IPS an efficient processing is demanded. Semi Global Matching (SGM) introduced by [19] tackles this problem. It is a state of the art stereo matching algorithm which represents a good trade-off between accuracy and computational complexity. SGM allows the pixel wise disparity estimation based on mutual information while being robust against radiometric differences. The algorithm takes a rectified image pair, as described in Section 2.3.2, as input so that the epipolar lines are parallel and only linewise comparisons have to be done. The resulting disparity map follows *semi global* smoothness constraints that preserve sharp boundaries. The matching costs of each pixel and possible disparity values are computed from different directions to ensure the constraints. All costs combined for each pixel form a matrix. By finding the optimal path through this matrix the real disparity is calculated (see Figure 2.13 (a)). By using dynamic programming to reuse already calculated values the complexity can be decreased while increasing the memory demand. In postprocessing outliers are removed and gaps to a certain size can be interpolated.

One major drawback of SGM in combination with a fully passive system like the IPS is the inability to match pixels in homogeneous areas. Here the matching comes to no result. One advantage of SGM however is the possibility of parallelization due to the use of simple operations independent from neighboring pixels and the rare use of conditional statements [20].

(a) Directions for cost aggregation [19].

(b) Exemplary image of the left camera.

(c) Visualization of the disparity map..

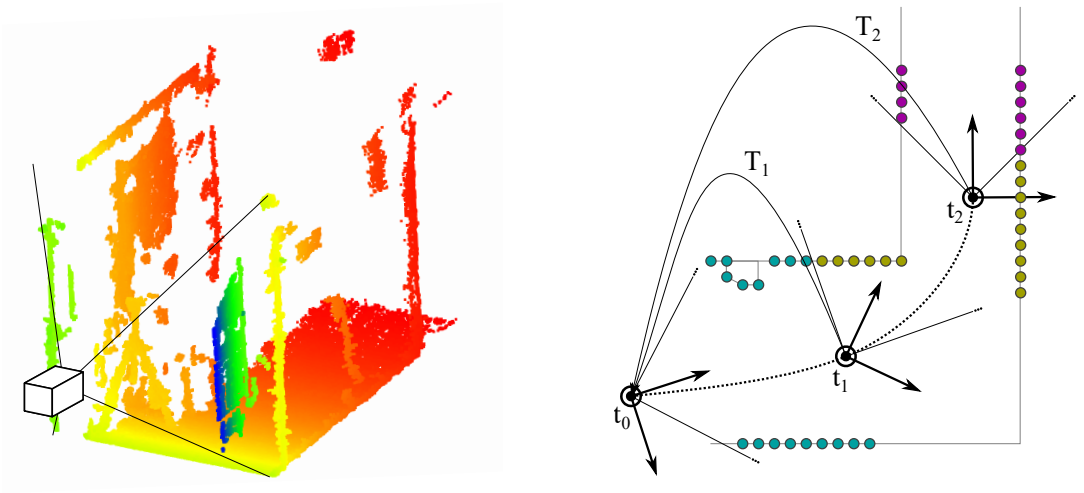
Figure 2.13: Illustrations of Semi Global Matching.

2.3.4 Point Cloud from registered Depth Maps

To generate a point cloud out of images the 2-dimensional image points have to be transferred into the 3-dimensional space. By combining images taken from different positions, large areas can be displayed in a point cloud. This requires estimating the 3D coordinate for as many pixels as possible.

For each pair of rectified stereo images, SGM derives a disparity map for the pixels of the left camera. Based on the stereo information each disparity value can be converted into the distance between the left camera center and the depicted object point. The combination of the x and y sensor coordinates with the calculated distances provides a 3D coordinate for each point in the coordinate system of the left camera. By doing so, the image points can be reprojected into 3D space as pictured in Figure 2.14 (a) creating a local point cloud.

The information about the position of the camera in a local coordinate system allows the unification of the retrieved point clouds. The IPS determines a trajectory as described in Section 1.2. For each camera frame captured the relative pose to the starting point is known. Every point cloud has to be transformed into this coordinate system of this point. This is done by applying the respective transformation T_i derived at the capturing time t_i on each point in the point cloud. In postprocessing, the derived point cloud is filtered to suppress noise and outliers and a voxel grid filter is applied to achieve a desired resolution.



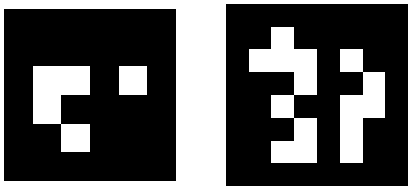
(a) Reprojected image points into 3D. Colors indicate distance from the camera.

(b) Point cloud from multiple views. Points of one color belong to one point cloud frame.

Figure 2.14: Point cloud derivation from depth maps and trajectory.

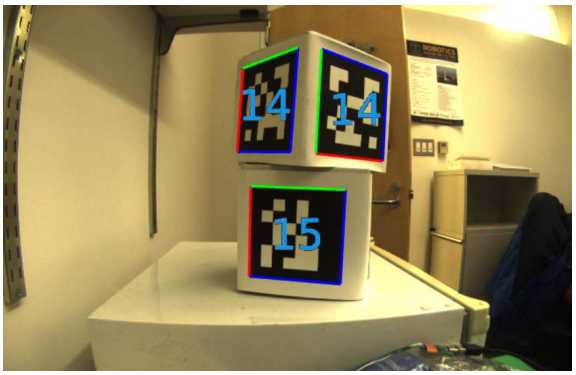
2.3.5 AprilTags

AprilTags are fiducial visual markers that allow the creation of a controllable environment for pose estimation. They represent artificial visual features that can be robustly detected in images. By being placed in the monitored environment, AprilTags serve as distinguishable control points. They were introduced in 2011 by [21] and are widely used in different applications especially in robotics and mapping applications. Each tag is a square with 2D bar code like pattern consisting of black and white squares as seen in Figure 2.15 (a). The arrangement of the squares represent a visual encoding that allows the identification. It can be easily detected even under difficult lighting conditions. There are different configurations depending on the usage. A fine subdivision allows the differentiation of many tags, a coarse one of less. The latter is easier to detect from a greater distance. [21] presents a system that provides a robust detection method. It is based on a line detection approach and a recursive depth-first search considering the crossing points of the four edges. By estimating a homography based on the detected squares the tags can be localized in six degrees of freedom from one single image (see Figure 2.15).



Tag16_05. Tag36_11.

(a) Different AprilTag-Versions.



(b) AprilTag detection in an image.

Figure 2.15: Examples for AprilTags (images from [21]).

3 Related Work

Detecting changes is a common goal when data from a particular environment is collected at different times, and therefore, methods for detecting changes in different applications have been developed for decades. A prominent example is the investigation of satellite data [22]. The fixed orbits of satellites enable regular data acquisition over a defined period of time. After successful registration, the data can be examined for example for changes in land use, buildings or seasonal conditions. These data are often image data, which makes the detection of changes a 2-dimensional problem. With the development of capturing techniques using stereo vision or LiDAR sensors, height profiles can be recorded. These represent a 2.5-dimensional data and can be used for example to check building heights or land deformations [23].

With the rise of ground-based sensors such as laser scanners or mobile mapping systems using photogrammetric approaches, change detection in the 3-dimensional domain became of interest. A wide variety of methods for detecting changes of any scale have been developed in recent years. Some methods work directly on point cloud data, others use simplified models of the captured environment.

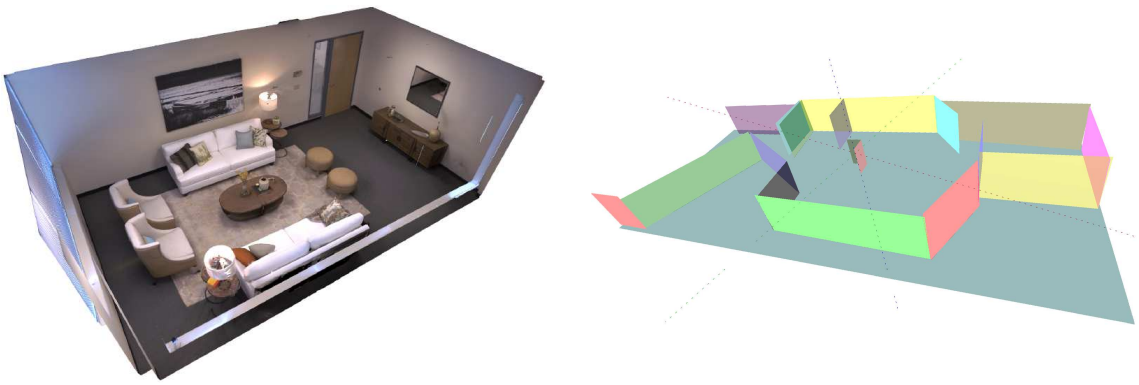
This chapter introduces several methods addressing the problems of model derivation and change detection. Therefore, Section 3.1 describes techniques to derive models of different accuracy. Section 3.2 further describes different procedures in detecting change with and without the usage of models. A dedicated consideration of different methods for the registration of 3-dimensional data is not given, since the registration is not the main focus of the work.

3.1 Model Derivation from Point Clouds

A wide variety of applications are dependent on specific models derived from point cloud data. Instead of working directly on the raw data, these models, containing further information of the captured environment, can be used. The model specifications in turn depend on the requirements of the respective application. In some cases it is desirable to accurately reproduce reality, in other cases to abstract and compress the data retaining spatial information. In each case the underlying point cloud is a measurement representing only

a small part of the real world that is additionally proned by noise during the capturing process and often inconsistent. These influences determine the model creation and the visualization directly. [11] refers to this as the modeling and visualization problem. In the following different, representations and techniques for model derivation are described.

For simulation and visualization purposes, a high resolution model containing a high level of detail is needed. A research group from Facebook Research introduces in [24] a photo realistic 3D dataset of scanned indoor environments. Each scene contains a dense mesh with high-dynamic-range textures and information about reflective surfaces. Additionally each primitive has semantic and instance information that can be used as ground truth in machine learning based research. Further studies describe the used methods of abstraction and surface reconstruction [25] [26]. The described model derivation is highly dependent on a dense and low-noise point cloud which is captured with a hand held camera based scanner similar to the IPS. This system also has an active infrared pattern projector as well as two ultra wide angle cameras to enhance depth and positional information. The resulting models represent the underlying point cloud accurately, but do not provide a significant simplification due to dense meshing. An example is given in Figure 3.1 (a).



(a) Digital Replica Dataset Example [24]. (b) Model build from planar primitives [27].

Figure 3.1: Models representing exact replications.

Another approach to abstract and simplify 3D data is the fitting of geometric primitives into the point cloud data. This approach assumes that all objects can be approximated by combining many simple 3D objects, called primitives. Each of these primitives is a certain instance of a parameterizable template or object type (e.g. planes, cylinders, cones, etc.). An example can be seen in Figure 3.1 (b) where an indoor environment gets coarsely modeled by fitted planes. [28] summarizes several different techniques

to retrieve geometric primitives. As long as the fitting performs well, the results are visually appealing. Previous knowledge of the scene to be recorded is necessary, as these determine the parameterizable templates and only these are approximated.

The methods described so far derive a geometric model from the given point cloud. These can also become extensive with large or very detailed scenes. Exact geometric replicas are not always necessary. In some applications, strongly abstracted models are advantageous. [29] determines a Gaussian Mixture model of a scene by analyzing the point cloud. The clusters of points are described here by multidimensional Gaussian functions, which most accurately reproduces the set of points considered. Each cluster can then be represented by its mean and its standard deviation, significantly reducing the amount of data. Gaussian Mixture Models allow simpler processing of large data sets in exchange for realistic point cloud visualization.

A very simple approach in deriving a semantic model from point clouds is by segmenting the scene and classifying segments as interesting or ignorable parts, as for example done in [30]. Especially in automotive application a fast processing of measurements is necessary for security reasons. By considering the recording patterns of laser scanners, which provide an ordered point cloud, a fast segmentation is possible. The scene is divided in different classes. One class contains planar features including for example the ground plane and buildings, another class contains movable objects. For efficient processing only movable objects have to be processed.

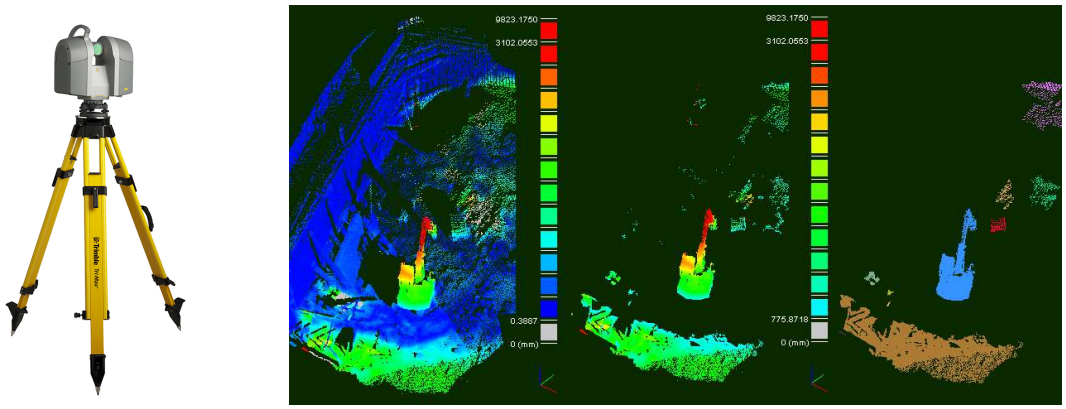
3.2 Change Detection in Point Clouds

The detection of change is based on measurements taken at different points in time and can be done at different scales. One intuitive approach for this is the direct point based comparison of two point clouds. [31] provides a detailed description of change detection in point clouds acquired with ground laser scanners (see Figure 3.2 (a)) at construction sites. The first step before detecting change is the proper registration of both point clouds. Working with laser scanners, this can be done by physical targets that are matched in 3D space and by manual point assignment. Iterative Closest Point (ICP) is a commonly used method that iteratively approximates the transformation based on salient points in the point cloud. In [31] ICP is discarded due to a lack of robustness. The registered point clouds are represented in an octree data structure to speed up the comparison by comparing only points that are within the same region. For the change detection, different

metrics for direct cloud to cloud comparisons are evaluated and the Hausdorff distance finally used. The Hausdorff distance between to point sets is computed for each point p in a point cloud S and represents the distance to the closest point in point cloud S':

$$d(p, S') = \min_{p' \in S'} ||p - p'||_2 \tag{3.1}$$

The distance is computed for all points and leads to a result as shown in the left image of Figure 3.2 (b). After computing the distance for all points, points with a smaller distance than a certain threshold are removed. The remaining points represent the detected change. For better scene understanding the changed points are clustered and outliers are removed (see Figure 3.2 (b). An evaluation of different point cloud sizes for detecting change showed that computing the Hausdorff distance is computationally expensive and the calculation depends on the chosen octree depth. Similar approaches are implemented by [32], detecting changes in nuclear power plants also in point clouds from ground laser scanners, and by [33] detecting changed objects in urban environments with a mobile car mounted laser scanner system. By using laser scanners these approaches profit from a low noise level, allowing exact registration. This does not apply to the IPS. Furthermore, the change detection should be real-time capable, which might be challenging to achieve by calculating the Hausdorff distance for huge point sets.



(a) Example of a ground laser scanner [34].

(b) From left to right: calculated Hausdorff distances, distance thresholding and 3D connected components extraction. Blue points indicate a distance smaller than a few millimeter, green and red distances bigger than a meter (Image from [31]).

Figure 3.2: Change detection in point clouds from laser scanners.

In the construction of detailed models of large scale urban environments, elaborate methods and sensors, as for example a combination of airborne and ground based scan-

ning techniques, are used for 3D reconstruction. The acquisition of the model is accordingly complex and cost-intensive. Therefore, the re-scanning of an area after structural changes should be kept to a minimum. To achieve this, [35] and [36] suggest to detect changes with the help of inexpensive sensors such as cameras and then carry out detailed measurements only in the identified areas. Here a change is detected by comparing images from different viewpoints with the previously captured extensive 3D model of the analyzed scene. By doing so, no 3D model has to be derived from the images. The change detection can directly be performed on the newly captured image data. Each viewpoint is known relative to the given model. By projecting one image onto the model and reprojecting the colored points back to another image inconsistencies between the image and the reprojection can be detected. This is also done vice versa. Those inconsistencies indicate changes between the model and the image sequence. By comparing multiple viewpoints the certainty of the detected changes increases as described in [37]. This method depends on a detailed model and registered images.

The described techniques so far rely on a geometric model and are not real time capable due to the large amount of data points. An efficient representation is necessary to compare large datasets. An abstraction of the sensed environment as described in [29] is used in [38]. An autonomous mobile security robot detects changes automatically in an unknown environment by analyzing the point clouds captured in regular patrols. All runs are robustly registered by extracted visual features and for the reference run a Gaussian mixture model is derived from the point cloud. The model divides the point cloud in multiple Gaussian distributions that represent regions with similar normal direction and color. Each distribution is specified by a 3D mean value in euclidean space and the corresponding covariances as well as a color mean value and its covariances. The change detection is then carried out by comparing the point cloud of each run with the reference model. For each point in the compared point set a probability, that the point is somehow changed, is calculated based on the distributions of its corresponding region:

$$p_{diff}(p) = e^{-(p-\mu_c)^T C_C^{-1}(p-\mu_C)} \tag{3.2}$$

This is done for the color value as well as for the position of the points. If multiple distributions are in the region of the point, the probability in each distribution is calculated and the smallest one taken. A low probability indicates that the point already exists in the reference model, a high one indicates that a change in the position or color appeared.

4 Method

Simplifying representations and efficient procedures enable the extraction of meaningful information from the multitude of three-dimensional information as represented by a point cloud. By comparing different datasets of a scene, additional temporal information can be obtained. This chapter proposes a method that implements such a comparison.

The IPS is used to acquire a trajectory and a point cloud for each run as described in Section 1.1 and 2.3.4. In this thesis, two point clouds are model-based compared against each other and the detected changes are visualized. By doing so, no pointwise comparison is necessary anymore. The proposed method abstracts the point cloud to a simplified model. This model is annotated with change information by comparing the model with another abstracted point cloud acquired at a different point in time. In some scenarios, the point clouds, and thus the models, are not superimposed. In this case, an additional registration step transfers both models into the same coordinate system. In order to implement this model-based change detection, the procedure is divided into the three parts *Registration*, *Model Derivation* and *Change Detection*, as shown in Figure 4.1.

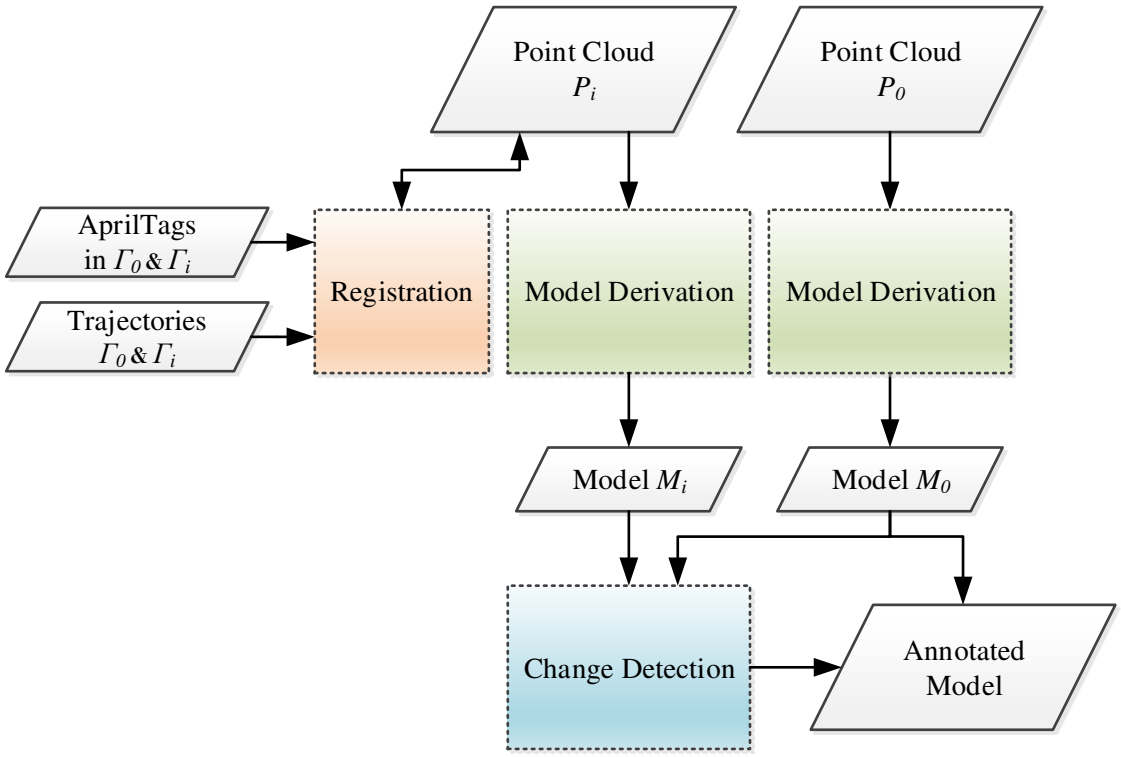


Figure 4.1: Overview of the method.

For the ease of understanding, the used datasets are referred to as a reference dataset P_0, denoted with 0, and a comparison dataset P_i, denoted with i. The registration utilizes the positional information derived from AprilTags detected in the stereo images. Combined with the trajectory of each run corresponding point pairs are derived and a transformation of the coordinate system of P_i into P_0 is calculated. This procedure is explained in detail in Section 4.1. In the next step, a simplified model is derived from each of the two point clouds by segmenting distinct regions. Planar surfaces and non-planar areas are extracted and processed independently of each other. In the application considered, planar surfaces make up a large part of the point cloud, but are not affected by geometric changes. This reduces the number of points in the analyzed point-sets. In order to retain the geometric information in the model, each plane is simplified and planar triangle meshes are derived from the points. The extraction and modeling of planes is described in Section 4.2. The remaining non-planar objects are abstracted by Axis Aligned Bounding Boxes (AABBs), which in turn are modeled as triangle meshes as shown in Section 4.3. The usage of AABBs allows a fast approximation due to the simple definition of corner points. By using bounding boxes that are axis aligned, the comparison of objects is reduced to the comparison of the values on the respective coordinate axes. This is done for the reference and the comparison dataset, resulting in the respective models M_0 and M_i. The comparison of the bounding boxes of both models provides the annotation of model M_0 by changes with respect to model M_i. This procedure is realized by a crosswise comparison as shown in Section 4.4.

4.1 Registration

When comparing two 3D scenes based on the direct comparison of three-dimensional entities, both representations have to be in the same frame of reference. This process is called registration and can be realized by transforming one coordinate system into the other. Since the scenes differ only by translation and rotation, and a possible scaling error is neglected, the coordinate systems can be transformed into each other by applying an euclidean transformation T. This transformation has six degrees of freedom and is applied as stated in Equation 2.4 to all points of the point cloud which are to be transformed. T must be determined to successfully accomplish the registration. The estimation and registration process as realized in this work is depicted in Figure 4.2.

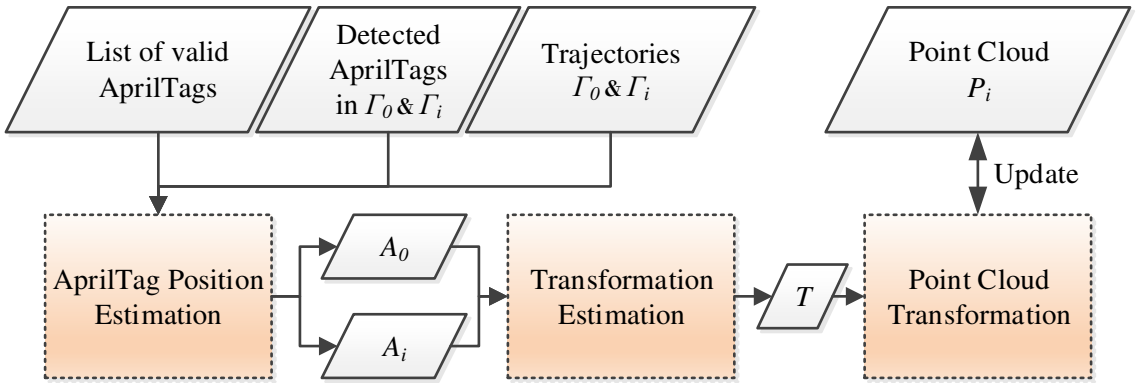


Figure 4.2: Data flow of the registration procedure.

With a minimum of six corresponding points, three in each system, the transformation matrix T between the coordinate system of P_0 and P_i can be retrieved [18]. In addition to the point clouds P_0 and P_i the IPS captures a trajectory Γ_0, Γ_i and detects AprilTags in the images. The underlying detection framework provides the 3D position and ID of each detected AprilTag relative to the left camera at the time of acquisition. Theses information are used to derive corresponding point pairs. As the tag positions are only relative to the left camera at the time of acquisition, they have to be transformed into the same coordinate system as the respective point cloud. This process is described in detail in Section 4.1.1. The position estimation of the AprilTags has to be done for all datasets which are to be registered. For each dataset the local pose and ID of all detected AprilTags are stored in a list A. In order to avoid the usage of false detected AprilTags, only valid IDs are used. Detected IDs that are not in a list of valid IDs are discarded. The registration process uses only the most certain point pairs from A_0 and A_i. The extraction of corresponding points and the estimation of T is explained in Section 4.4. After successfully calculating T the point cloud P_i is transformed and the coordinates updated.

4.1.1 Position Estimation of AprilTags

With the help of AprilTags as fiducial markers, known and assignable 3D positions can be derived. The IPS framework provides a robust method to detect these markers in the images and to calculate the corresponding 3D coordinate with the help of the stereo information. To use all detected markers of an entire run for registration they must be located in one coordinate system. The local transformations involved in the position estimation of each tag are visualized in Figure 4.3.

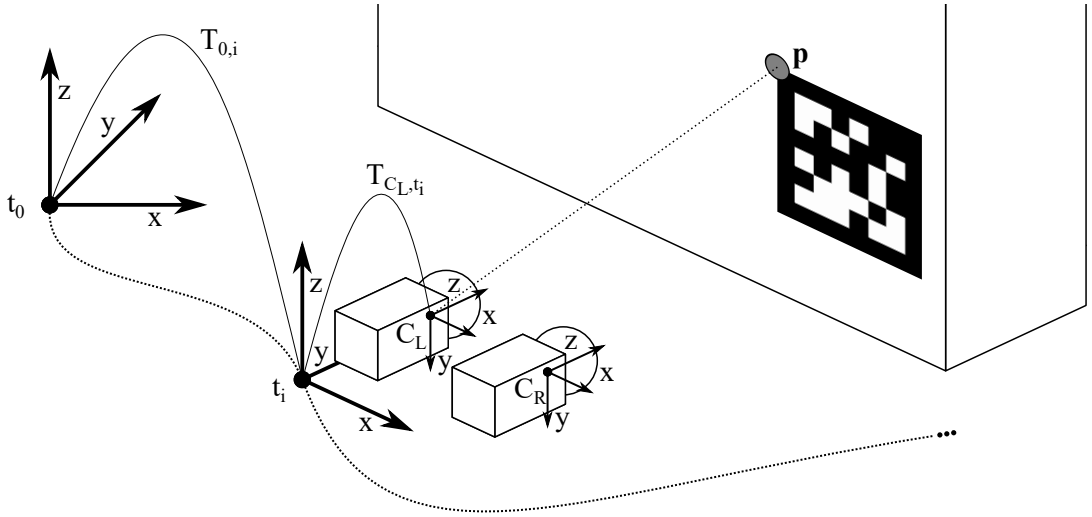


Figure 4.3: Position estimation of an AprilTag including the transformation into the local coordinate system.

At the beginning of a run, a local coordinate system is spanned at the starting point. The derived trajectory is defined within this system and describes the position of the IMU at a point in time t_i. Due to the previous calibration, the relative position of the left camera to the IMU is known and thus also the transformation between the two associated coordinate systems. These information combined allow the transformation of the detected tags into the local coordinate system. Starting from the 3D position $\mathbf{p}$ of the AprilTag relative to the left camera the position in the local coordinate system at time t_i is calculated by transforming $\mathbf{p}$ with the transformation matrix T_{C_L,t_i}. The result is further transformed into the coordinate system at time t_0 with the use of the transformation $T_{0,i}$.

$$\mathbf{p}' = T_{0,i} \cdot T_{C_L,t_i} \cdot \mathbf{p} \tag{4.1}$$

Additionally to the coordinates describing the position of the IPS, the framework provides a covariance matrix characterizing the certainty of each measurement. The covariances are propagated through the transformations, whereas the uncertainty is also given for each marker position. The local 3D positions and corresponding IDs as well as the covariances for each detected AprilTag are stored in a list and passed to the transformation estimation.

4.1.2 Transformation Estimation

To transform one known coordinate system into another, a minimum of three corresponding point pairs is necessary. These points are represented by AprilTags that are distributed in the observed scene. After successfully deriving the positions, their IDs and covariances, a list of possibly matchable points is provided to derive the transformation. This list contains only positions with valid IDs, but it is not guaranteed that all captured markers appear in both datasets. Furthermore, most IDs appear several times in the dataset. The cameras acquire multiple images per second, so that a tag can be seen in a series of images. Also, it is possible that an area is observed several times during longer sessions. Multiple occurrences of a tag require the selection of a particular one.

As the length of the trajectory increases, so does the uncertainty of the current position. Therefore, the position of markers detected later in the local coordinate system is also less reliable (see Figure 4.4). Contradictory the uncertainty decreases if a marker is better visible in one images than in another one. Since the uncertainty is known from the covariances, it can be used as a measure to filter the captured markers.

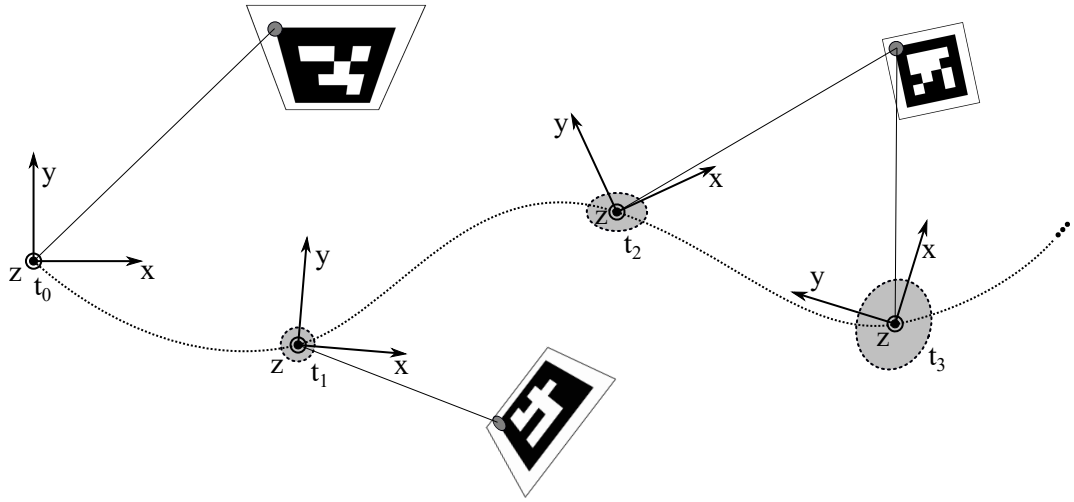


Figure 4.4: AprilTags selection based on uncertainty for the transformation estimation. Gray spheres indicate increasing uncertainty of the respective pose.

A cross check first discards all IDs that do not appear in both point lists. For the remaining points, the uncertainties are calculated from the standard deviation along each axis derived from the covariance matrix of each point:

$$u_i = \sqrt{\sigma_{x_i}^2 + \sigma_{y_i}^2 + \sigma_{z_i}^2} \tag{4.2}$$

By doing so, the best visible marker with the lowest uncertainty is retrieved. All other points with the same ID are discarded. This is done for all IDs and results in a list of unique and distinguishable points that appear in both trajectories. Marker IDs that appear in both lists of the to be registered runs provide the point pairs necessary for the transformation estimation.

These point pairs are order and placed in two matrices M_0 and M_i which used to set up a linear equation system satisfying:

$$M_0 = T \cdot M_i \tag{4.3}$$

By solving this equation system using a singular value decomposition T can be retrieved [18]. The resulting transformation matrix is finally applied to the points of the point cloud to be registered P_i.

4.2 Model Derivation of Planar Surfaces

The model derivation starts with the processing of planar surfaces. These constitute large parts of the point cloud in indoor environments and can be easily simplified due to their uniform alignment. In this work a triangle mesh approximates the points within a plane in order to achieve a high simplification rate with good representability at the same time. The geometric information is represented by the triangles. Calculating as large triangles as possible reduces the number of points to a smaller number of vertices. All points inside a triangle are represented by the face. In the following, a method is proposed to derive planar meshes from a given point cloud based on the work of [39]. This is composed of four steps as shown in Figure 4.5. First, planar surfaces are extracted by a segmentation as described in Section 4.3.1. The extracted points are processed further, while the remaining points are handled separately. During the next step, the extracted planes are simplified. The simplification is based on a quadtree structure in which the points and the boundary of the plane are inserted. The boundary extraction and the subsequent simplification of the point cloud is described in Section 4.2.2. A meshing process transforms the simplified point cloud based on the quadtree into a triangle mesh. This process is explained in Section 4.2.3. The resulting meshes are stored in the model structure.

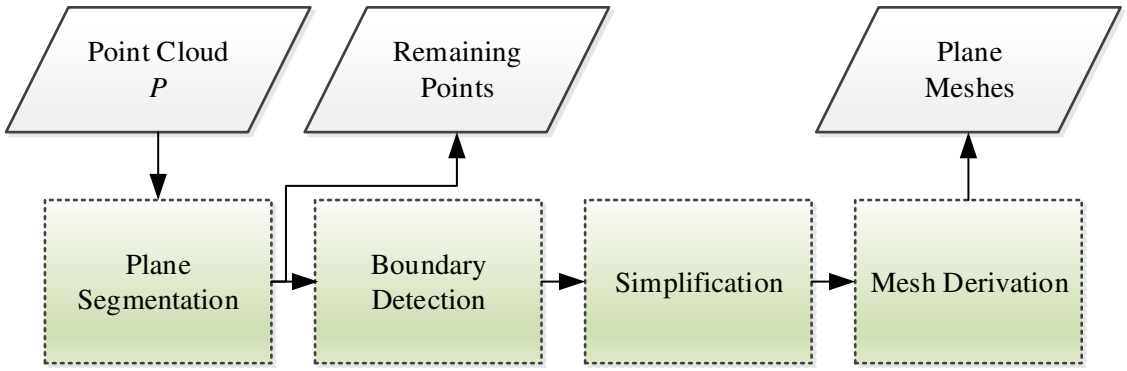


Figure 4.5: Process of the planar model derivation.

4.2.1 Plane Detection

The first step in the simplification of planar surfaces is the extraction of points representing individual planes in the point cloud. To identify planes, the points are analyzed individually and classified based on their properties. Normals and curvatures are calculated for each point in the point cloud as described in Section 2.2.2. It has been observed that a search radius of $r = 3 \cdot leafSize$ is suitable to calculate these values for point clouds as provided by the IPS. The leaf size is determined in an earlier step of the point cloud generation.

A region growing process evaluates the points regarding their curvature and normals and iteratively assigns all points in a neighborhood to related areas. This approach ensures that only contiguous regions are detected and the planes are well separated. The process is parameterized by a smoothness threshold and a curvature threshold. The processing starts at the point with the small curvature value in the point cloud. A small curvature indicates a flat area and thus a possible plane. The normal and curvature of the starting point serves as a reference value for all point comparisons with respect to this plane. This starting point is called seed point. The seed point is compared against its k nearest neighbors in terms of its normal direction and its curvature. All neighbors with normal directions that are within the previously defined threshold are assigned to the same plane as the seed point is. The number of neighbors k is a parameter that strongly influences the plane detection. If k is too small the region growing stops easily, if k is too large neighbors too far away are added to a plane. If they differ more than the threshold, the neighbor is not taken into consideration for the particular plane but can be evaluated for another plane. All points added to a region in turn serve as seed points again. In the next iteration this process is repeated for all new seed points. Neighbors that have already been evaluated

are now omitted. This process continues until no more points can be added to the current plane. After finishing a plane extraction, again the point with the smallest curvature value is searched and the procedure is repeated until there is no point with a curvature value smaller than the curvature threshold value. This means that every point has been checked at least once in the region growing process.

All points that have not been segmented are considered to be part of the non-planar point set. In addition, planar segments that are too small are also added to this point set. The minimum size of a plane is given as parameter. With the completion of the process, the initial point cloud is divided into a set of points representing individual planes and a set of points belonging to non-planar objects. An exemplary result can be seen in Figure 4.6. The individual plane point sets are further processed within the plane modeling.

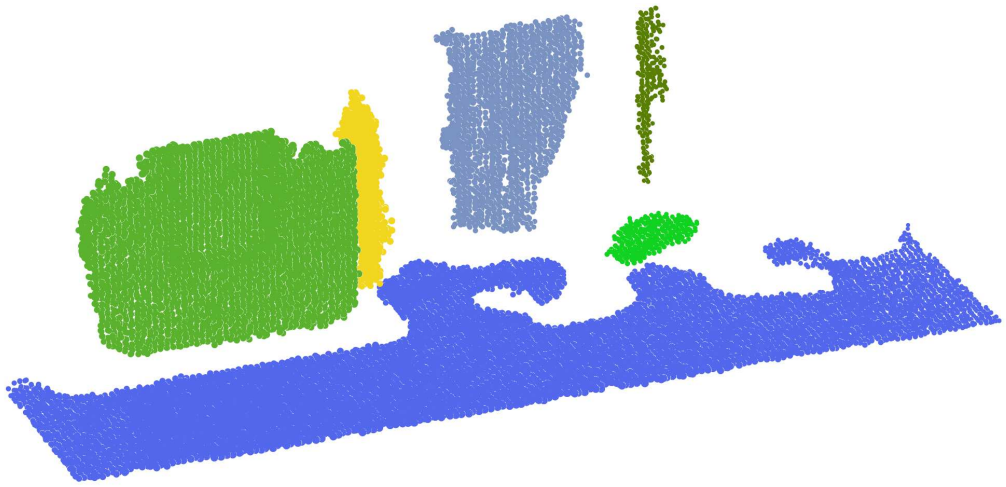


Figure 4.6: An exemplary result of the plane segmentation in an indoor scene. Different colors indicate different segments.

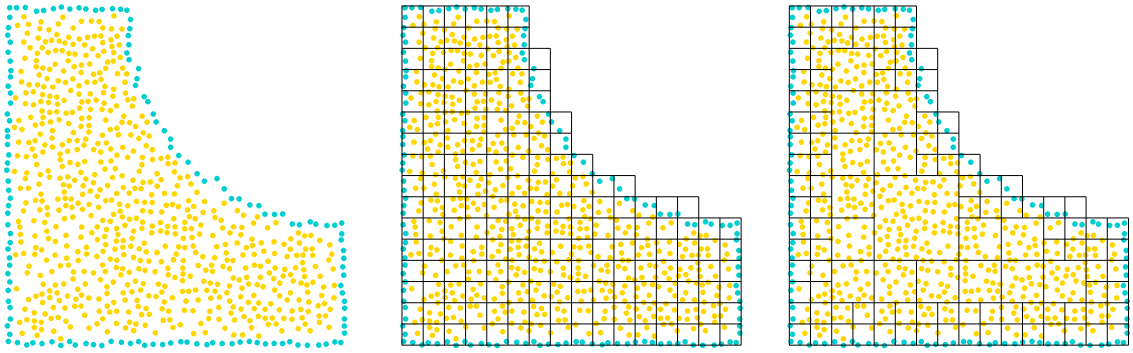
4.2.2 Simplification of Planes

Each extracted plane consists of many points. If it is known that the points all lay in one plane, each point of a dense point cloud contains little geometric information. Therefore, it is reasonable to reduce the number of points to increase the information density per point. With regard to the geometric information, the boundary of the plane would be sufficient to interpolate all points within a plane. In the subsequent derivation of a well-defined triangle mesh, however, the inner angles of the triangles used should be as large as possible. Therefore, the simplification method presented in the following retains

information on points within the plane boundaries. When deriving a well-defined triangle mesh, however, it is important to ensure that the inner angles of the used triangles are as large as possible. The following simplification procedure considers this requirement by retaining enough points within the plane boundaries.

In order to process the planes as such, the dimensionality of the planar points is reduced from 3D to 2D. The pose of the individual planes is arbitrary, whereby the coordinates of the points correlate with each other. To be able to handle the coordinate axes individually, each plane is transformed into the x-y plane. A Principal Component Analysis (PCA) derives the transformation by analyzing the point set regarding their eigenvectors [40]. The transformation is applied to the points and later on inversely applied to the resulting plane mesh to restore the previous pose. This way, the z axis can be neglected, resulting in a 2-dimensional surface.

The interior and the boundary of the planar point cloud are considered separately. Using the alpha shape as described in Section 2.2.3, the boundary of the plane is extracted. In Figure 4.7 (a) an example of a 2-dimensional point cloud with its extracted boundary is depicted.



(a) Planar point cloud with detected boundary in blue.

(b) Initialized quadtree at lowest level.

(c) Combined quadtree cells.

Figure 4.7: Simplification of an example point cloud (Adapted from [39]).

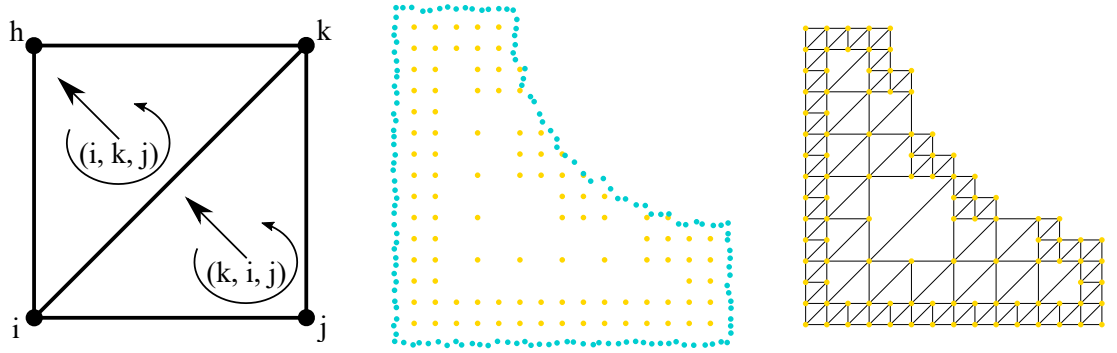
For the simplification, a modified quadtree structure as described in [39] is used. In addition to the points, this structure contains at the individual nodes information on whether the respective node is inside, outside, or on the boundary of the represented point set. The depth of the tree is defined in advance and influences the accuracy of the simplification towards the boundary of the plane. In the first step all points are inserted into the quadtree defining all used nodes as interior nodes. The extracted boundary points are inserted afterwards, defining all touched nodes as boundary nodes. By doing so, each

node, that contains nodes that belong to the boundary in a lower level, is therefore defined as a boundary node. All nodes that are not reached are defined as exterior and thus not considered. The example depicted in Figure 4.7 (a) results in a quadtree with a grid at the lowest level as shown in Figure 4.7 (b).

The simplification is included in the structure of the quadtree. By creating the structure in the described manner, all nodes that contain interior nodes but are no boundary nodes are themself interior nodes. Each node that is defined as an interior node represents all points below it sufficiently. The covered area of each node is as large as possible regarding the quadtree, so that the following meshing will be successful. The grid of the point cloud example thus can be simplified to be as depicted in Figure 4.7 (c). To retain a minimum of color information for a better visualization each node gets the average color of the underlying points assigned.

4.2.3 Mesh Derivation

After completing the simplification, the quadtree structure allows the derivation of a triangle mesh based on the simplified grid. The remaining cells of the quadtree are used to define the vertices and faces of the mesh representing the planar surface. Each node contains the information about its position within the plane and, based on its level, information on the size of the area it covers. These information allow the definition of four points spanning a rectangle as seen in Figure 4.8 (b). A rectangle can be represented by two triangles in a triangle mesh with these four points serving as vertices. By connecting the top right, top left and bottom left as well as the top right, bottom left and bottom right point two triangles are defined counterclockwise (see Figure 4.8 (a)). All points get the color of the represented node to color the face. This is done for all interior nodes of the quadtree resulting in a mesh as for example represented in Figure 4.8 (c). For simplification the modeling of the interior nodes is sufficient, for a detailed model the boundary can be triangulated additionally with a suitable method such as a constrained delaunay triangulation [41]. After meshing all nodes the simplification is finalized by transforming the planar mesh back into its original pose by applying the inverse transformation calculated by the PCA as described in Section 4.2.2.



(a) Triangles representing a node.

(b) Extracted vertices from the quadtree.

(c) Triangulated and simplified mesh.

Figure 4.8: Derivation of a triangle mesh from the quadtree structure (adapted from [39]).

4.3 Model Derivation of Non-Planar Objects

The extraction of planes removes a large set of points from the point cloud. All remaining points represent areas with a coarse structure, such as objects of smaller scale, edge areas around objects, and measurement inaccuracies. These point sets are simplified by approximating bounding boxes that substitute areas with similar properties in a process as shown in Figure 4.9. To retain enough spatial information, points that seem to belong to one object are combined. However, this enforces a trade-off between accuracy and efficiency of the simplification. The segmentation of related points is described in Section 4.3.1. The resulting segments are further analyzed with respect to their spatial extent and bounding boxes covering the respective points are derived. The boxes are to be represented in the same way as the planes as triangle mesh. Therefore, a meshing of the bounding boxes is necessary. This process is described more in detail in Section 4.3.2.

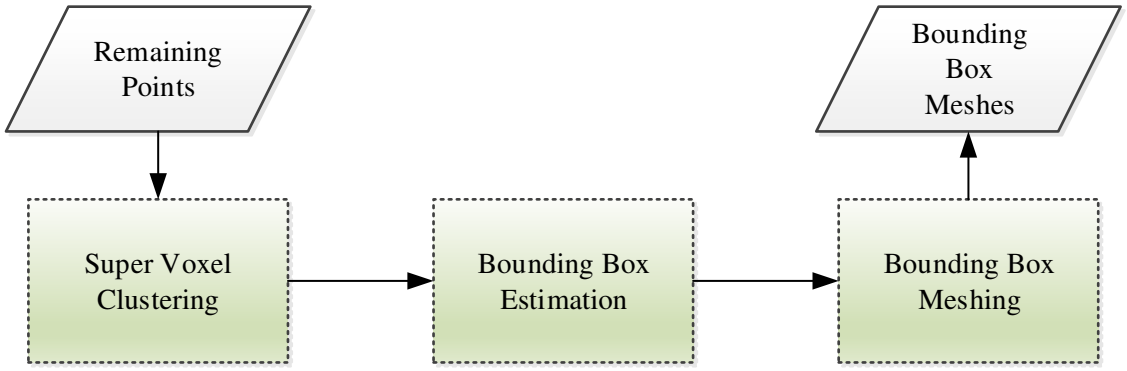


Figure 4.9: Process of the model derivation of non-planar objects.

4.3.1 Super Voxel Clustering

The segmentation of the point cloud containing all non planar objects utilizes a similar approach to the planar segmentation. A region growing process combines sets of points with similar color, normal direction and position in groups by analyzing their neighborhood. This is done by extracting so called *Super Voxels* in a super voxel clustering as introduced by [42]. The algorithm is parameterized by weights that regulate the assignment of points to clusters based on their color, normal direction and distance to the seed point. Additionally, a maximum cluster size is defined which prevents an undersegmentation of the point cloud. This would otherwise result in large bounding boxes that do not represent the underlying points properly. The segmentation starts with the uniform distribution of seed points in the point cloud. Each seed represents a potential super voxel. Iteratively each super voxel is extended by its k nearest neighbors when they are within a distance D. The distance D is calculated by weighting the spatial distance D_s, the distance in color space D_c and the angle between the normal vectors D_n based on the predefined weights w_s, w_c and w_n:

$$D = \sqrt{w_c D_c^2 + \frac{w_s D_s^2}{3R_{seed}^2} + w_n D_n^2} \tag{4.4}$$

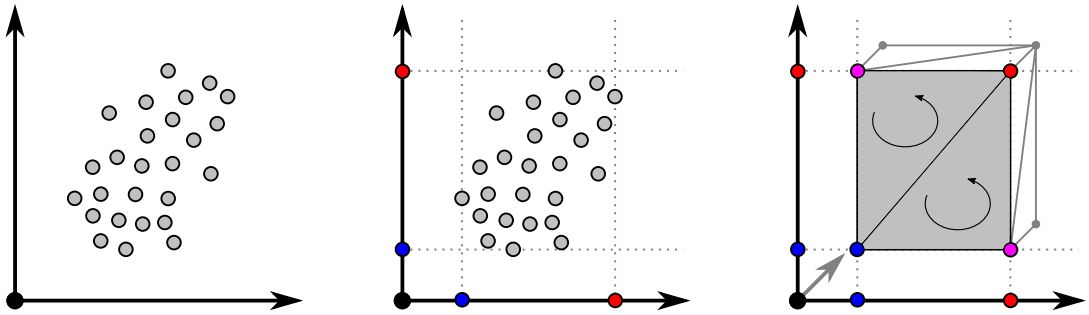
This step is repeated with increasing distance D from the seed point until all points of the point cloud are assigned. The distribution of seed point in the point cloud is parameterized by the cluster resolution. It states the distance between the starting points for the region growing process. In this work, the cluster resolution is set depending on the cluster size by $clusterResolution = 1/2 * clusterSize$. This experimentally derived dependency leads to compact clusters with few outliers in most cases. The described segmentation technique results in clusters of similar size. An example is shown in Figure 4.10.

4.3.2 Bounding Box Estimation

In the last step of the simplification, the extracted clusters are approximated by bounding boxes. Bounding boxes represent the smallest cuboid that encloses all points of a point set. In this thesis, AABBs are used to approximate the segments. With this type of envelopes, the axes of the enclosing cuboid are parallel to the respective coordinate axis. Thus, the enclosed volume is not as small as possible, as with an oriented bounding box, but the generation and comparability is much simpler. To generate a triangle mesh that

Figure 4.10: Example of super voxel clustering in a point cloud. Each color indicates one cluster.

can be represented in the same model as the extracted planes, each cluster extracted in the segmentation is processed separately as shown in Figure 4.11.



(a) Extracted cluster.

(b) Minimum and maximum determination.

(c) Bounding Box Meshing.

Figure 4.11: Process of bounding box estimation.

The bounding box is defined by 8 points representing the corners and 6 sides spanned by 4 corner points each. For a triangle mesh, the sides are additionally divided into two faces. The coordinates of the corner points of the enclosing box correspond directly to the largest and smallest value of the considered point set along the respective coordinate axis. By extracting these minimum and maximum values, the vertices for the bounding box are defined. All vertices are extracted in the same order, so that the faces can be derived directly without the need for an additional meshing process. The faces are defined counterclockwise so that the normal faces point outwards.

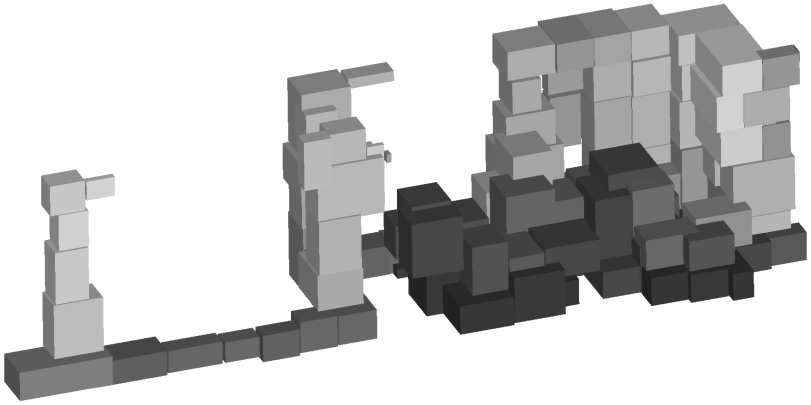


Figure 4.12: Exemplary results of the bounding box estimation.

4.4 Change Detection

The purpose of change detection is to recognize in which areas of the point cloud changes have occurred. Objects can be added and removed. These possibilities are treated separately as shown in Figure 4.13. The approach of detecting change is based on the extracted AABBs of both compared models. The planes of both models are not taken into account since the constraints assume that the changes are primarily caused by coarse structures and are thus represented as bounding boxes. Both models are directly compared by checking the bounding boxes of one model to the boxes of the second model in its region for intersection. By using AABBs, the evaluation of intersections is reduced to the direct comparison of the coordinates in the respective axis of compared bounding boxes. Bounding boxes of the compared model M_i that intersect with the bounding boxes of the reference model M_0 are considered as unchanged, non intersecting boxes as changed. By comparing both models vice versa removed objects are classified.

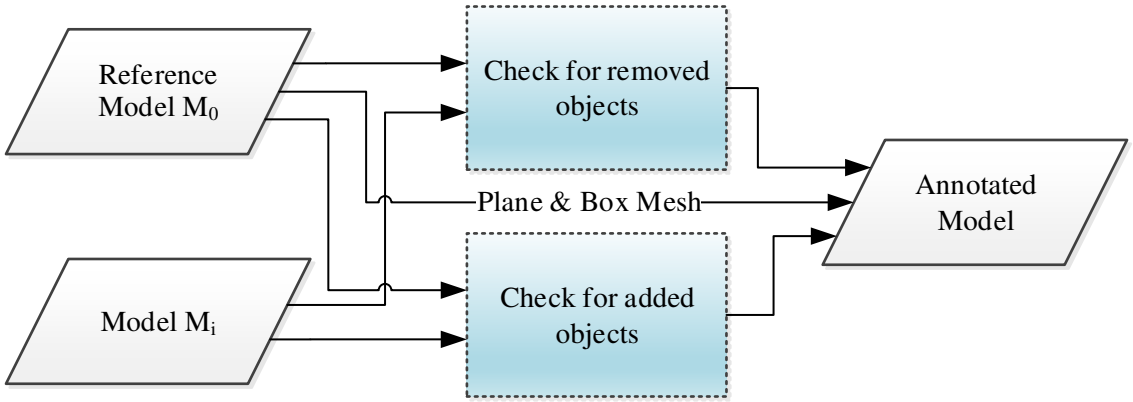


Figure 4.13: Process of the change detection based on the non-planar objects.

5 Evaluation

The proposed methods for registration, model derivation and change detection can be adjusted by different parameters and are subject to various influencing factors. Therefore, they operate with different precision depending on the application area. This has to be considered and quantified. In addition to the purely qualitative assessment of the results with regard to meaningfulness and visual quality, it is also necessary to consider quantitative metrics. These allow an objective and automated analysis of the processing results.

The evaluation is based on datasets, described in Section 5.1, that have been acquired under consideration of different application and evaluation scenarios. The results of the evaluation are structured according to the different proposed methods. A qualitative evaluation of the registration of multiple point clouds with AprilTags is presented in Section 5.2. The model derivation is evaluated in Section 5.3. The quality of the change detection depending on the simplification of the point cloud is explained in Section 5.4.

5.1 Datasets

In order to evaluate the proposed methods, four datasets were acquired with the IPS, each representing a different use case and its requirements. Since the intended application areas of the method are primarily indoor environments, three of the four datasets are designed for this purpose. A dataset created with the help of a simulation tool allows the evaluation of the change detection under controlled conditions. The simulation framework and the aquired dataset is described in Section 5.1.1. This dataset is refered to as *Indoor - Simulation*. The following datasets are recorded in a series of measurements with the IPS and thus represent real data. They are summarized in Section 5.1.2. The dataset *Indoor - On a Vehicle* resembles the simulated data and shows how the method behaves when processing real data. In the simulation as well as in the real replica no automatic registration is necessary. In order to investigate this, the dataset *Indoor - Handheld* is added. Multiple handheld runs with the IPS are registered and combined using AprilTags distributed in the changing scene. In order to open up further application possibilities, the dataset *Outdoor - Car mounted* represents a street scene as an outdoor environment examined for changes.

5.1.1 Simulation

The recording of real data with the help of the IPS is subject to fluctuations with regard to the areas observed in a scene. These fluctuations create measurements in non-overlapping areas, which are detected as changes in the proposed method and thus make a quantitative examination of the capabilities more difficult. In addition to capturing real measurements with the IPS, the department also has a simulation environment for the entire data acquisition chain of the system at its disposal. This simulation tool enables the dedicated investigation of influencing factors on the visual and inertial odometry in the further development of the IPS [43]. It allows the free definition of a measuring sequence in an artificial 3D scene. By repeating the measuring sequence in the same scene with changes introduced, the change detection can be observed independently of measurement fluctuations. An introduction of the framework and its configuration as well as a description of the featured scene is given in the following.

Simulation-Framework

The simulation framework allows the recording of realistic measurements by defining a digital twin of the IPS in a simulated environment. It requires a definition of the IPS consisting of the camera specification of the left and right camera, as well as the relative positions of the cameras and the IMU. The produced artificial images are similar to realistic data due to the simulation of various camera effects [43]. The framework supports common 3D formats and allows the creation of arbitrarily complex 3D scenes. This trajectory can either created by defining artificial control points, reproducing a real run by loading a previously recorded trajectory or by freely moving the simulated setup in the virtual 3D scene. The output of the virtual IPS matches the format of a real one and can be processed in the same way. By defining the trajectory, the same measurement can be repeated as often as desired with different parameters.

Scene Data

An indoor scene is used for the evaluation of the method in the simulation. A digital replica of the corridor used in the real datasets is available for this purpose. This replica represents an approximation, since no exact measurements of all details of this scene are given. However, the general dimensions are identical, so that trajectories recorded in real measurements can be passed into the simulation without further processing. By

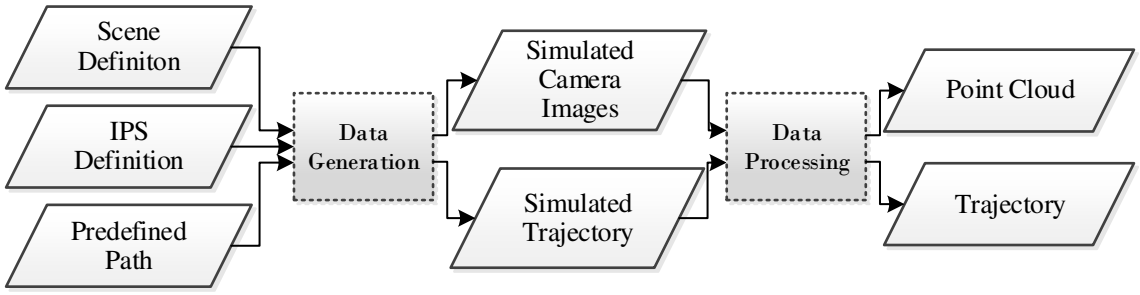
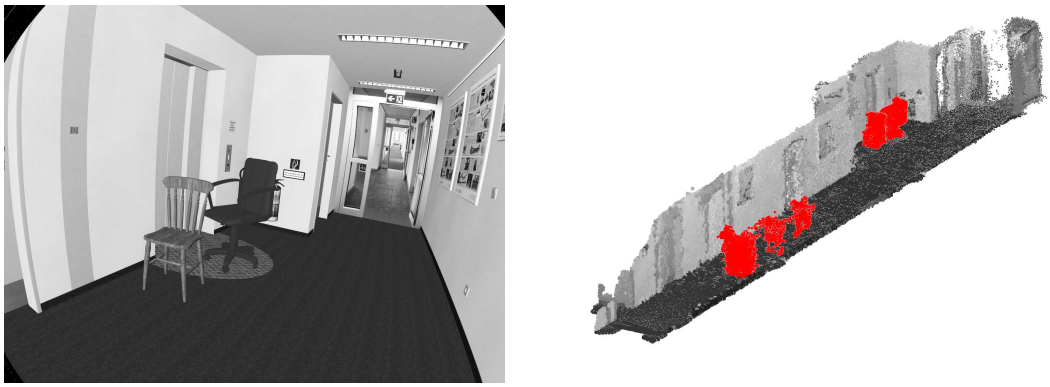


Figure 5.1: Data acquisition pipeline of the simulation framework [43].

using a realistic trajectory and defining similar light sources an accurate simulation can be achieved as depicted in Figure 5.2 (a) The same trajectory is used in two different runs in the simulation. In the second run three office printers and two chairs are introduced in the virtual 3D scene as changes. An example of the resulting point cloud is given in Figure 5.2 (b).



(a) Example image *Indoor - Simulation*.

(b) Point cloud *Indoor - Simulation*.

Figure 5.2: Exemplary data of the dataset *Indoor - Simulation*.

5.1.2 Real World

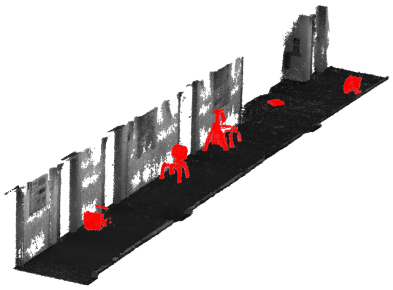
To evaluate the method on real data, three datasets were acquired under different conditions with the aim of covering a variety of applications. The first two datasets are captured in a corridor with a configuration of the IPS with a base length of 20 cm, which is appropriate for indoor applications, while an outdoor scene is captured with another configuration with a base length of 50 cm. In the following, these three different datasets are described.

Indoor - On a Vehicle

This dataset is intended to reproduce the scenario from the simulation by capturing a similar measurement in the real world. An empty corridor is recorded in a first run. In this corridor changes are introduced and a second measurement is carried out. The modifications are boxes of different sizes and chairs.

While a defined trajectory can be specified in the simulation, an exactly identical path in a real environment can not be guaranteed. In order to approach this, the IPS is placed on a mobile pedestal and slowly pushed through the corridor that is shown in Figure 5.3 (a). This fixated the measuring height to the height of the vehicle and prevents measurement inaccuracies due to unsteady movements or fluctuating viewing directions. By defining a fixed stop as a common starting point, the measurements ought not to require registration. Since the length of the measurement was about $20\ \mathrm{cm}$, orientation changes of half a degree results in an offset of $\tan 0.5^\circ \cdot 20\ \mathrm{m} = 0.17\ \mathrm{m}$. Therefore the registration of the runs was corrected manually afterwards. In order to minimize lighting changes, the measurement runs were carried out at night so that only artificial non changing lighting illuminates the scene. In the selected configuration, this dataset can also be seen as the exemplary measurements of a patrolling robot and thus represents a possible application scenario. The resulting point cloud in which changed areas are highlighted in red is shown in Figure 5.3 (b).

(a) Example image with placed objects.



(b) Point cloud (no wall for visualization).

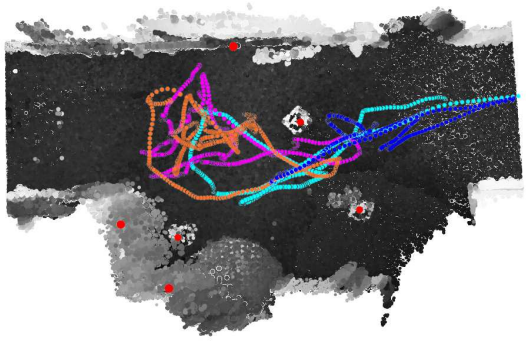
Figure 5.3: Examplary data of the dataset *Indoor - On a Vehicle*.

Indoor - Handheld

In many cases, the IPS is carried by hand through the environment to be inspected. These runs often have unsteady paths and are influenced by the operator's handling. In addition, the starting pose changes in many cases even if this is not intended (see dataset Indoor - On a Vehicle). If the resulting point clouds of two runs are to be evaluated and processed, there is a risk that non-overlapping areas of the scene were observed in both runs. This causes areas that cannot be compared. To counteract this, it is advisable to capture point clouds that are as comprehensive as possible. The coverage can be increased by combining individual point clouds from runs where different areas are captured.

This is demonstrated by this dataset, where a corridor was captured in varying constellations. It consists of seven measurement runs, each with different start and end positions. The real trajetories of the runs are visualized in an abstracted model of the corridor as seen in Figure 5.4 (b). To demonstrate the registration, AprilTags were distributed in the scene. They are placed in such a way that they can be seen from a similar direction early during the runs, regardless of the starting position (see Figure 5.4 (a)). In addition, changes were made in between the runs in order to also use the dataset for the evaluation of the change detection. The first three runs capture the corridor without any modification. Run 4 and 5 capture one modified version of the corridor containing several changes varying from boxes and bins, with a minimum extent of about 25 cm at the shortest side, up to a chair and boxes similar the ones used in Indoor - On a Vehicle. In run 6 and 7 a small table that was inserted as change, is removed.

(a) Example image *Indoor - Handheld*.



(b) Four trajectories in the corridor. Red dots indicate AprilTags.

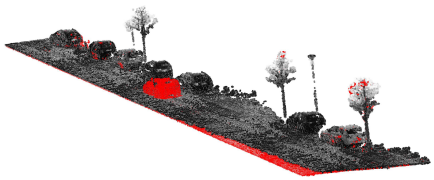
Figure 5.4: Trajectories and placement of the AprilTags in the dataset *Indoor - Handheld*.

Outdoor - Car mounted

In addition to the indoor scenarios, an outdoor scenario is also considered in order to evaluate the proposed method in other application areas. In outdoor areas, the captured point cloud is often more sparsely occupied than in indoor areas. Additionally, both application areas differ with regard to the presence of planar surfaces. Usually, the ground plane makes up large parts of the scene so in long runs the captured ground can become very extensive.

In this dataset a wide baseline configuration of the IPS is mounted on a transporter capturing the area in front. The car drives in an industrial area around the block so that start and end position are close to each other. The scene consists of the street and trees and parking cars on the left and right side of the street as seen in Figure 5.5 (a). This path is repeated four times. Originally this dataset is captured to evaluate the performance of the position estimation of the IPS. In between the respective rounds some cars left the scene and some parking spots got occupied by new cars. For the evaluation of the change detection one part of the street is separated and two runs selected. In this area one car appears in the middle of the road and one parking car on the left. These two parts of the scene are captured from similar points but the viewing direction is not equal resulting in large non-overlapping areas. To minimize this error, the left part of the road that is not seen in both rounds equally is removed. The resulting point cloud can be seen in Figure 5.5 (b). Despite capturing the area in one run, a scaling error in the trajectory leads to a drift in the point cloud. A minor manual registration was necessary

(a) Example image *Outdoor - Car mounted.*



(b) Point cloud *Outdoor - Car mounted.*

Figure 5.5: Exemplary data of the dataset *Outdoor - Car mounted.*

5.2 Performance of the Registration

In this section the registration of several point clouds in one coordinate system is qualitatively evaluated. A quantitative evaluation, for instance regarding the robustness of the detection or the influence of the distribution of the markers, is not necessary, since the registration itself is not the main focus of this work and merely serves the completion of the processing chain. For this reason the registration is demonstrated exemplary by an example similar to future applications and the results of this example are reviewed.

The evaluation is based on the dataset *Indoor - Handheld*. The presented seven different runs in an environment with minor changes are to be registered. The first run is set as reference independent of the quality of the marker detection in order to come as close as possible to a realistic application where no selection is done. The following six runs are registered towards this run. For this, the markers are detected in each run as described in Section 4.1 and the best position estimate for each marker is selected based on the accuracy of the detection. Figure 5.6 shows the six to be registered point clouds of the respective runs including the trajectories in blue and the best detected position of the markers in red. The image axes are parallel to the x and y axes of the respective coordinate system. The start poses can be similar in places but have differing orientations.

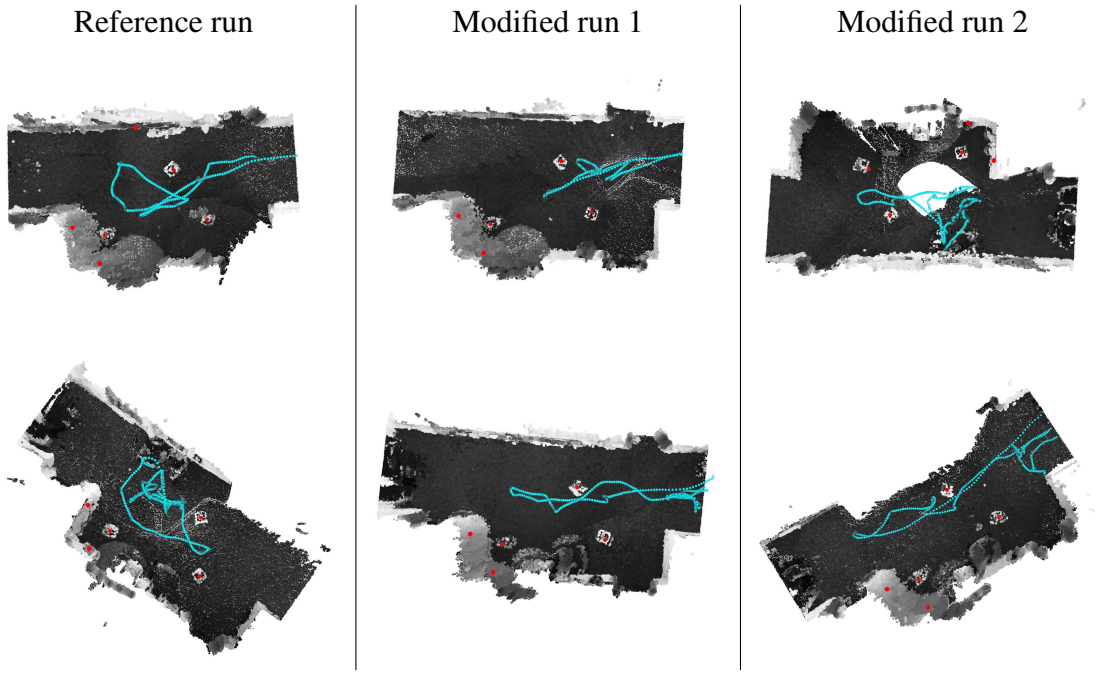


Figure 5.6: Point clouds of the dataset *Indoor - Handheld* with visualized AprilTag positions and trajectories.

Table 5.1 shows the accuracy for each run for each of the most certain and therefore used AprilTag positions. The standard deviations of the respective positions allow the calculation of the accuracy $a = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}$. It turns out that the accuracy is similar for most AprilTags and around $0.13\ \mathrm{m}$. AprilTags that are captured from a closer distance are more certain. In the last row of the table the resulting standard deviations of the respective transformation of the point cloud into the reference coordinate system is given. The left column describes the position in meter and the right column the rotation uncertainty in radian. It can be seen that the uncertainty of the translation is approximately $0.05\ \mathrm{m}$ and the rotation uncertainty between 0.05 and $0.2rad$. Figure 5.7 shows the point clouds of all

Table 5.1: Uncertainties of the AprilTag positions and the resulting transformation in m.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
$AT22$	0.147	0.139	0.146	0.130	0.028	0.132	0.129
$AT25$	0.140	0.127	0.139	0.130	0.066	0.124	0.122
$AT35$	0.099	0.087	0.088	0.087	0.120	0.068	0.061
$AT36$	0.146	0.136	0.146	0.136	0.076	0.133	0.130
$AT38$	0.108	0.089	0.108	0.163	0.132	0.074	0.090
σ_x σ_ω	0.04 0.08	0.05 0.08	0.06 0.10	0.07 0.10	0.05 0.08	0.05 0.09	0.04 0.09
σ_y σ_ϕ	0.04 0.16	0.04 0.16	0.04 0.20	0.04 0.05	0.04 0.05	0.03 0.15	0.05 0.16
σ_z σ_κ	0.04 0.14	0.04 0.13	0.05 0.13	0.05 0.12	0.04 0.09	0.04 0.10	0.04 0.10

seven runs in one coordinate system. The first three runs are shown by their captured gray values, while the runs 4 and 5 are shown in red and 6 and 7 in yellow. The colored display shows that the transformation error affects the overlap of the point clouds. Especially in the right area the corners are not clearly overlaid. This error can be traced back to the error of the transformation as well as the scaling error of the IPS itself, which increases in particular with faster movements, such as when scanning a corner more precisely.

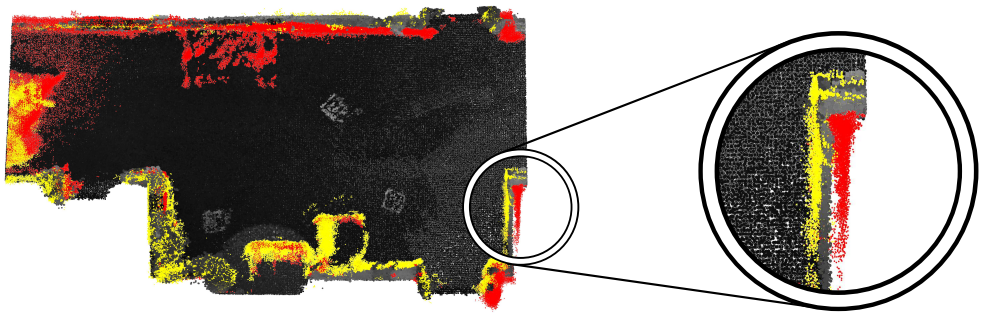
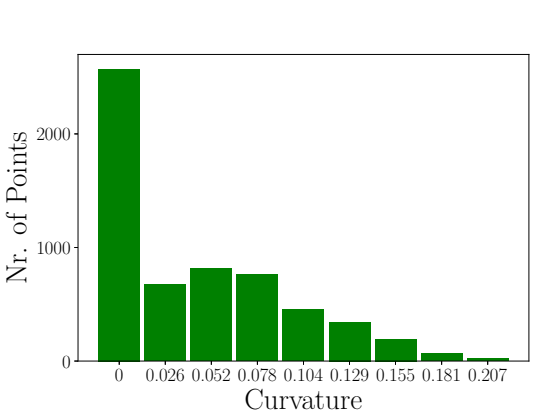


Figure 5.7: The point clouds of all six runs combined in one frame of reference with exemplarily highlighted inaccuracy. Colors indicate different runs.

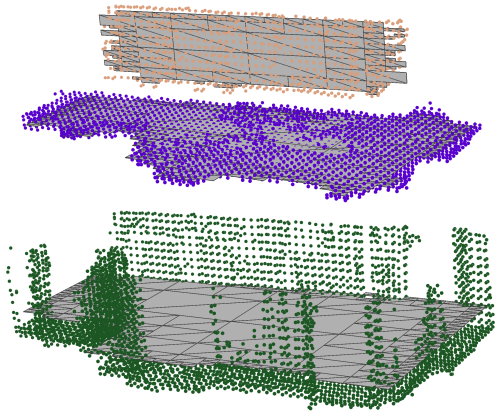
5.3 Performance of the Model Derivation

The quality of the model derivation depends on the detection and modeling of the planes and the non-planar objects. Both procedures are examined in this section with regard to their performance with respect to several different parameters. These parameters influence both the visual quality of the results as well as the accuracy of the modeled parts. The first are considered qualitatively, the latter quantitatively.

Alongside the quality of the measurement data, correctly selected thresholds for the plane detection play an important role in modeling the planes in a point cloud. In the following, the influence of the curvature threshold on the detection is examined based on the dataset *Indoor - Handheld* downsampled to a resolution of $10\,\mathrm{cm}$. Figure 5.8 (a) shows the distribution of the curvature values of the points in this dataset. Since the curvature calculation is based on the number of neighbors and not on a fixed search radius, it is independent of the point cloud resolution. Based on the histogram, different thresholds are applied and the results of two remarkable thresholds are depicted in Figure 5.8 (b). It shows an example of the segmented planes and the resulting plane meshes. Points of one plane are indicated by the same colors. While up to a threshold of $0.10\ \frac{1}{\mathrm{m}}$ the planes are well separated as shown in the top figure, it is not possible to differentiate the planes at a threshold of $0.15\ \frac{1}{\mathrm{m}}$. Thus all points are combined and modeled as one plane. As can be seen in the bottom figure, the position of the ground plane is shifted by the influence of the falsely segmented points of the close wall. Experiments in other datasets have shown that a threshold of $0.05\ \frac{1}{\mathrm{m}}$ often leads to good results independent of the investigated scene.



(a) Histogram of the curvature values in $\frac{1}{\mathrm{m}}$.



(b) Planes detected with a curvature threshold of $0.10\ \frac{1}{\mathrm{m}}$ (top) and $0.15\ \frac{1}{\mathrm{m}}$ (bottom).

Figure 5.8: Exemplary plane detection in the dataset *Indoor - Handheld*.

The next processing step approximates the extracted planes by triangle meshes. The degree of abstraction as well as the visual quality depend mainly on the depth of the respective quad tree. The following Figure 5.9 shows the ground plane of the dataset *Indoor - Handheld* modeled with different Quad Tree Depths (QTDs) as an example.

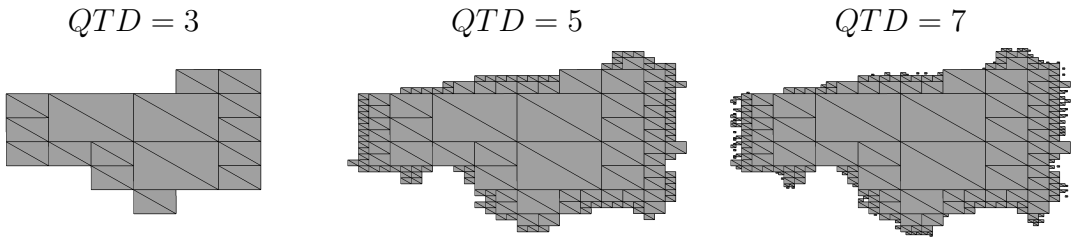


Figure 5.9: Ground Plane of dataset *Indoor - Handheld* modeled with different QTDs.

The greater the depth of the tree, the smaller rectangles at each level can be generated and the more accurately the point cloud is approximated. This increases on the one hand the accuracy and on the other hand the visual quality to a certain extent. In addition the number of generated rectangles increases with the QTD quadratically. Figure 5.10 shows the simplification rate for three datasets depending on the QTD. The simplification rate represents the number of plane points per generated vertex as $SimplificationRate_{Planes} = \frac{|Points|}{|Vertices|}$. The graph shows that the simplification rate decreases with increasing QTD.

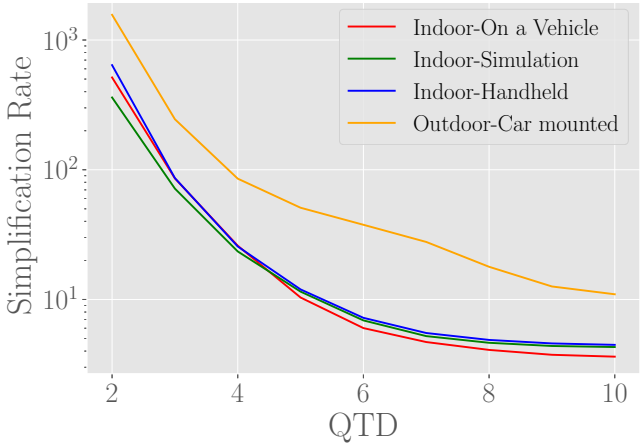


Figure 5.10: Influence of the QTD on the simplification rate of the planes.

Depending on the application, a trade-off between simplification and visual quality of the modeled planes must be found. However, the greatest usable QTD can be specified.

Depending on the minimum extend along one axis and the known leaf size for downsampling the maximum usefull depth can be calculated by:

$$QTD = \log_2 \frac{extent}{leafSize} \tag{5.1}$$

Greater QTDs are possible in principle, but are of no use, since the area per rectangle then becomes smaller than the resolution of the point cloud and thus only individual points would be added as rectangles with no extent.

Another parameter that influences the quality of the modeling is the selected alpha value, parameterizing the accuracy of the concave hull around the respective plane. Figure 5.11 shows three different plane models from one planar point cloud with $QTD = 5$. An alpha value of $\alpha = 0.1$ leads to a visually pleasing result. The process is susceptible to boundary detection that is too coarse. If points in the quad tree fall at the edge of the plane, but are not detected as boundary points, the corresponding nodes can be mistakenly marked as interior. This results in noisy plane modeling as shown with $\alpha = 0.01$. If the alpha value is very small as in the example at $\alpha = 0.001$, no node of the quad tree is extracted as boundary, but classified as interior node. This results in the maximum possible simplification whereby the plane is abstracted by its two dimensional AABB.

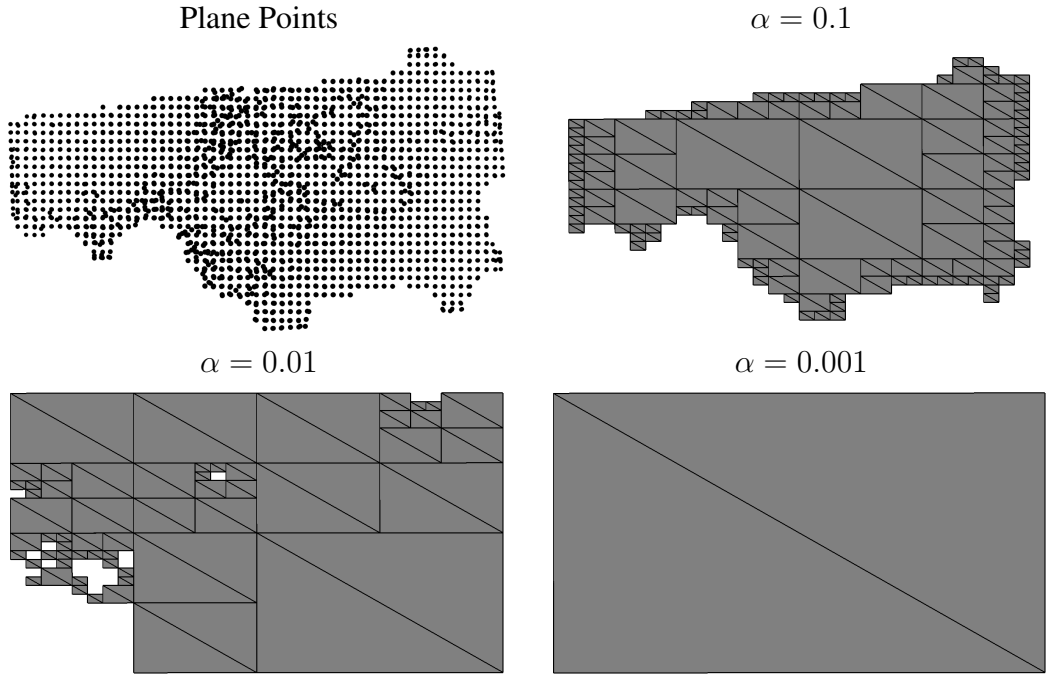


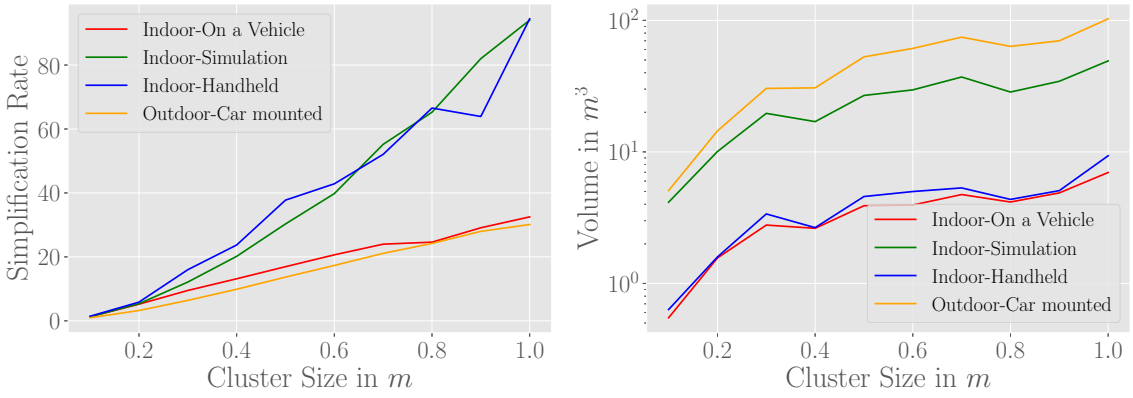
Figure 5.11: Influence of the alpha value on the plane modeling using the ground plane of *Indoor - Handheld*. $QTD = 5$, $LeafSize = 0.1m$.

After the extraction of the planes, all further points of the point cloud are clustered and the clusters are approximated by AABBs. The maximum size of the clusters is the decisive influencing factor. For the evaluation different cluster sizes are chosen and the reduction of the data volume by the approximation with AABBs is considered. The influence of the cluster size on the amount of points modelled is negligible. In any case, between 80 and 100 percent of all points are approximated. However, the cluster size affects the simplification rate that is obtained from:

$$SimplificationRate_{Boxes} = \frac{|Points|}{|Boxes| \cdot 8} \tag{5.2}$$

Each bounding box consists of eight vertices. Increasing the cluster size leads to larger bounding boxes, modelling more points. Furthermore, the number of boxes decreases. Fewer and larger AABBs lead to a higher simplification rate as can be seen in Figure 5.12 (a). If the cluster size is very small and reaches the resolution of the point cloud, degenerated bounding boxes are derived. Here the boxes are very small and consist of more points than the number of underlying points.

Larger boxes, however, tend to include empty areas as well, since for example larger clusters can also expand around corners. Figure 5.14 shows the summed up volume of all AABBs at one cluster size per dataset. For a reasonable modeling a cluster size is recommended, which allows a high simplification and at the same time encloses a small volume.



(a) Simplification rate of the AABBs depending on the cluster size.

(b) Covered volume of all AABBs per cluster size and dataset.

Figure 5.12: Influence of the cluster size on the modeling of non-planar objects.

5.4 Performance of the Change Detection

In each of the presented datasets there is some kind of change introduced. The accuracy of the change detection in these given point clouds is examined in this section. The datasets each consist of one reference run and at least one modified run as described in Section 5.1. The point clouds of the actually added objects are extracted from the modified runs and stored individually for each object and dataset as ground truth.

In order to limit the scope of the evaluation, the parameters for the extraction of the planes are fixed and only the non-planar objects are considered. Furthermore, the evaluation is based on the down sampled point clouds of the datasets as extracted in the plane detection and extraction. The down sampling is also applied to the ground truth point clouds. This ensures a uniform ratio between the set of points in the extracted ground truth and the modeled reference point cloud. In the evaluation, the points of the individual areas are checked for their correct assignment. This is done by examining each point, if and in which bounding box of the model or the annotation of changes the respective point lies. Thus the evaluation of the detection can be considered as a classification problem as shown in Figure 5.13 and the point cloud can be divided in several steps into different subsets P. The points of the reference point cloud and those of the changed ground truth objects as well as the change detection model consisting of the reference model and the change annotation serve as input.

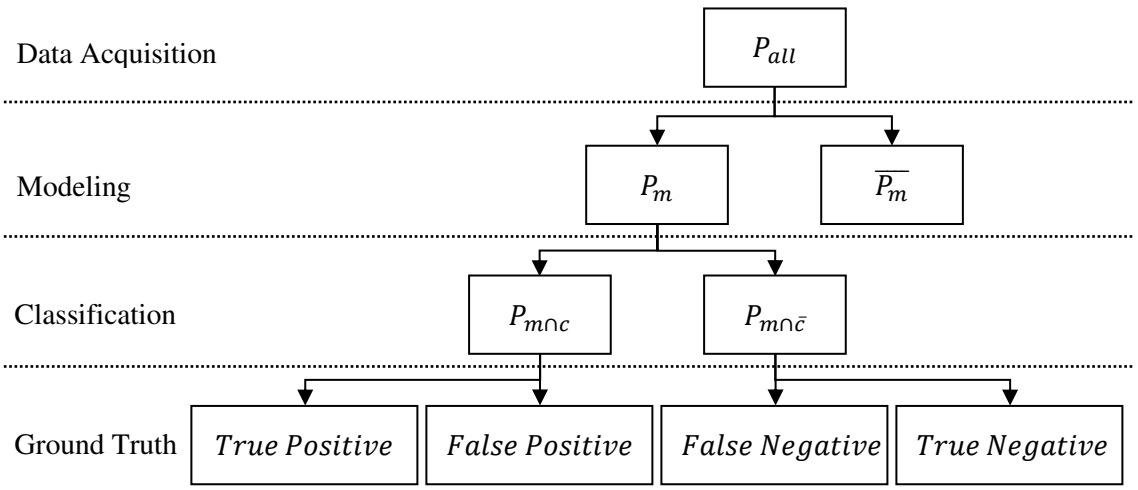


Figure 5.13: Assignment of the points of the point cloud and the detected changes on the basis of a classification problem.

In the first step each point of the set P_{all} is examined whether it lies within one of the bounding boxes of the model or not. The set is divided into the set of modeled points P_m and the set of not modeled points $\overline{P_m}$. All points in P_m are then categorized according to their classification by the annotated model as *modeled and detected as change* or *modeled and not changed*. This step results in the point sets $P_{m\cap c}$ and $P_{m\cap\overline{c}}$. Since the points that actually belong to changes are given as ground truth, the classification of the respective points can be checked for their correctness. All points in $P_{m\cap c}$ that are part of the ground truth are correctly classified and thus called *True Positives* (TP). All others are misclassified and thus *False Positives* (FP). The points in $P_{m\cap\overline{c}}$ that are not in the ground truth are modeled and correctly classified as unchanged and thus *True Negatives* (TN), all others are *False Negatives* (FN). The latter would have to be marked as changed. In order to prevent that no changes are modeled in case of bad modelling (e.g. one large cluster containing all points), all points that are not modeled but have to be classified as changed are additionally evaluated as *Not modeled False Negatives* (NMFN).

The numbers of TP, FP, TN, FN and NMFN are calculated for different cluster sizes and for all four datasets. The comparison of these evaluated classifications allows the quantitative interpretation of the change detection with regard to its quality. In the following, the sensitivity of the change detection is evaluated. The sensitivity results from the *True Positives* and the *False Negatives* as well as the *Not modeled False Negatives*. It describes the ratio of how many of the changed points were actually detected. A value of 1 corresponds to 100% meaning that all points were detected correctly.

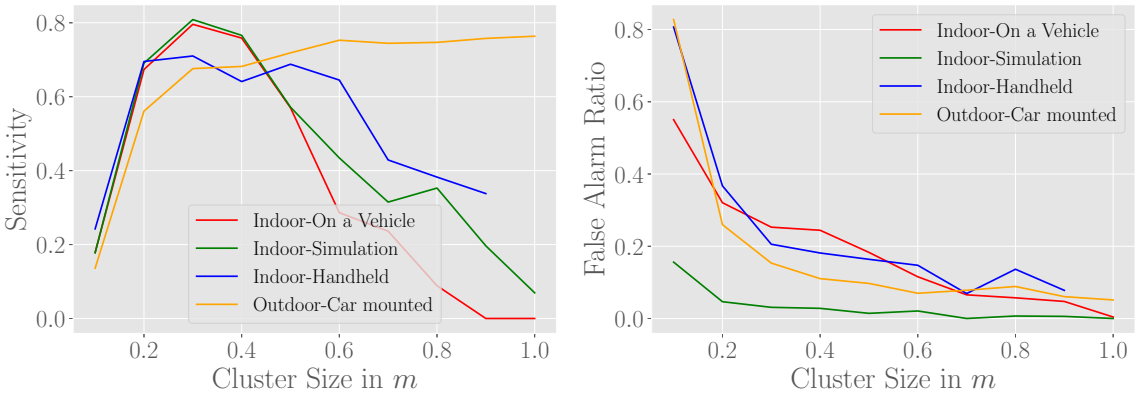
$$Sensitivity = \frac{TP}{TP + FN + NMFN} \tag{5.3}$$

In Figure 5.14 (a) the dependency of the sensitivity on the cluster size is plotted. The maximum sensitivity for the three indoor datasets is in the range $20 - 40cm$. With larger clusters the sensitivity decreases. This cluster size corresponds to about the half of the size of the objects that were inserted as changes. In the dataset *Outdoor - Car mounted* no decrease occurs. This is also not observed with larger cluster sizes $> 1m$ and can be justified by the sparse distribution of the objects in the scene. The AABBs never grow larger than the actual object size due to the correctly detected and removed ground plane. Since there are no other objects in the vicinity of each modeled object, the clustering algorithm stops when an object is completely modeled.

Another significant ratio is the *False Alarm Ratio*. It indicates how many of the detected changes are false and is derived from the *False Positives* and the *True Negatives*:

$$FalseAlarmRatio = \frac{FP}{FP + TN} \tag{5.4}$$

In Figure 5.14 (b) the *False Alarm Ratio* is plotted as a function of the cluster size. With very small cluster sizes there are many false detections. The number of false detection decreases with an increasing cluster size. This indicates that with large cluster sizes the certainty of the detected changes is higher than with small clusters. The graphs show that the change detection depends on the cluster size, which in turn depends on the size of the changed objects and the structure of the observed scene.



(a) Sensitivity depending on the cluster size.

(b) False Alarm Ratio depending on cluster size.

Figure 5.14: Influence of the cluster size on the modeling of non-planar objects.

5.5 Runtime

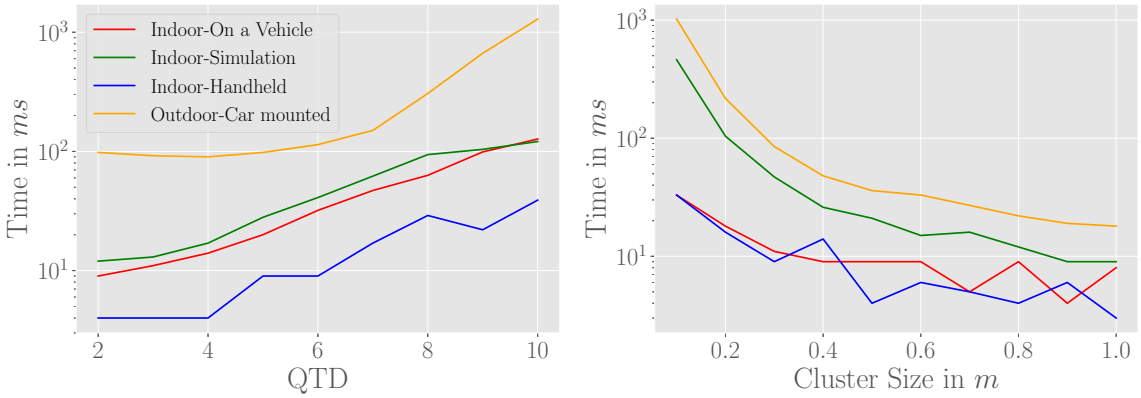
The runtime of the proposed method is an important aspect of the evaluation, since it is a limiting factor in real-time systems. The quad tree depth and the cluster size directly affect the runtime since they influence the amount of data points generated for the model. This chapter provides a review of the runtime of the modeling and the change detection.

The initialization requires a significant part of the processing time. In this step, the already loaded point cloud is sampled down to the desired resolution (10 cm in the depicted example) and the normals and curvatures are calculated. In Table 5.2, the time for initialization and plane detection is given for the individual datasets. The initialization is not

Table 5.2: Initialization and plane extraction including the boundary extraction.

	Indoor Simulation	*Indoor On a Vehicle*	*Indoor Handheld*	*Outdoor Car mounted*
Nr. Points	287165 Points	1750024 Points	690727 Points	513420 Points
Subsampling to $10cm$ and Normal Estimation	$161ms$	$171ms$	$75ms$	$302ms$
Nr. Remaining Points	27905 Points	14315 Points	5905 Points	113698 Points
Plane Extraction	$154ms$	$89ms$	$31ms$	$887ms$
Nr. Points in Planes	16335 Points	12065 Points	3509 Points	96466 Points

only dependent on the number of points, but also on the characteristics of the scene. The scene in *Indoor - Simulation* contains large planar areas and consists of less noisy points than the similarly arranged real *Indoor - On a Vehicle* scene, but more points. The plane extraction in contrast, is linearly dependent on the number of plane points as indicated in the line *Plane extraction*. Figure 5.15 (a) shows the influence of the quad tree depth on



(a) Influence of the QTD on the runtime.

(b) Influence of the cluster size on the runtime.

Figure 5.15: Influence of the QTD and cluster size on the runtime of the modeling.

the runtime of the plane modeling. The increasing depth obtains a higher runtime when aiming for a higher resolution of the model. Both increase quadratically to the quad tree depth. The modeling of the non planar objects, however, depends on the cluster size. The more clusters that need to be modelled, the greater the time required. This is shown in Figure 5.15 (b). A larger cluster size has an decreasing effect on the runtime. The overall runtime of the modelling is comparatively short and for the indoor datasets in a configuration with $QTD = 5$ and $clusterSize = 0.3$ m in any case less than 100ms.

The more clusters are extracted in the modeling, the more bounding boxes are compared in the change detection. The implementation in this work compares all clusters with

each other so the dependency between cluster number and execution time is quadratic and thus decreases with increasing cluster size. This dependency can be seen in Figure 5.16.

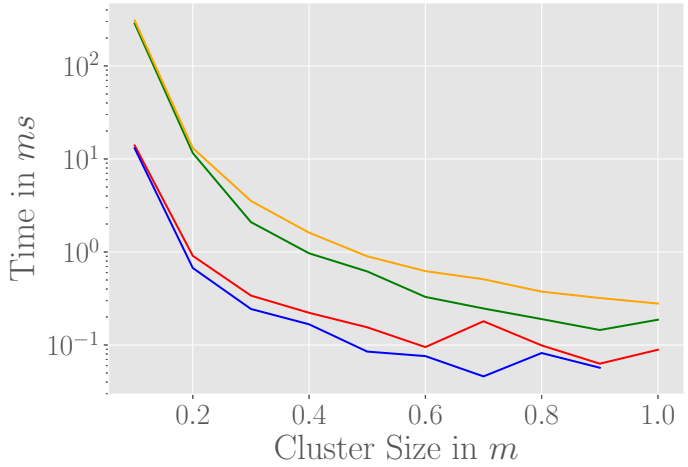


Figure 5.16: Influence of the cluster size on the runtime of the change detection.

To conclude, the runtime for a complete modeling of a reference dataset and a checked dataset as well as their comparison is examined. Table 5.3 shows the accumulated runtime of the whole change detection process in all four datasets. The runtime consists of the time for initializion and modeling of two point clouds and the time for the comparison. It is noticeable that the time for the change detection is very short compared to the time spent on the modeling.

Table 5.3: Accumulatd runtime at $ClusterSize = 0.3m$.

Runtime	*Indoor Simulation*	*Indoor On a Vehicle*	*Indoor Handheld*	*Outdoor Car mounted*
P_0 (Init + Modeling)	$155ms + 195ms$	$170ms + 124ms$	$92ms + 49ms$	$335ms + 1083ms$
P_1 (Init + Modeling)	$138ms + 189ms$	$197ms + 124ms$	$61ms + 43ms$	$304ms + 952ms$
Change detection	$2ms$	$0.3ms$	$0.2ms$	$4ms$
Accumulated runtime	$679ms$	$615.3ms$	$245.2ms$	$2678ms$

6 Conclusion

In this thesis a method for the recognition of changes, based on the modeling of point clouds, is investigated. For this purpose, a method for 3D modeling of point cloud data was designed and implemented. This modeling enables a compact representation of the data and a fast comparability based on it. In the following, Section 6.1 summarizes the developed processing steps. Section 6.2 then takes a critical look at the procedures based on the evaluation results. The results of this work show a potential way to further develop the IPS framework. These and possible application scenarios are presented in Section 6.3.

6.1 Summary

Within the scope of this work, three consecutive steps are elaborated and implemented with the aim of identifying and modeling geometric changes in 3D data. The processing chain consists of the registration of point clouds, the derivation of a triangle mesh based model as well as the recognition of changes in these models. For the registration AprilTags are used. In combination with the trajectories of the respective runs, these markers allow the calculation of a transformation between two local coordinate systems. By determining the position of the markers in the respective coordinate systems of the trajectories, no preceding assessment of the tag position is necessary. The functionality is summarized in an independent implementation. The registration can be applied in a preprocessing step if required or used independently from the change detection for other purposes as for example combining point clouds covering only partly overlapping areas.

The point cloud is sampled down to a uniform distribution for faster processing and the normals and curvatures are calculated for the remaining points. A region growing process uses this additional information to divide the point clouds into planar and non-planar areas. The former are modeled as planes through simplification and meshing using a modified quadtree structure. The non-planar points are clustered by supervoxels and the extracted clusters are approximated by Axis Aligned Bounding Boxes. Both planar and non-planar modeling is performed for all datasets which are to be compared. The bounding boxes of one dataset can be easily examined for their intersection with a second

dataset, due to their axis alignment. An exemplary result of the model derivation and change detection is shown in Figure 6.1.

Four different datasets are created for the evaluation of the methods. Each dataset contains at least two measurements with changes and the corresponding ground truth.

The evaluation indicates that the maximum cluster size to be chosen should be based on the size of the changes to be detected and the content of the scene. In indoor scenes, the best detection rates are given at block sizes of half of the object size to be detected. In scenes, where the objects are well separated as for example urban outdoor scenes, the cluster size can also be larger to allow faster modeling and comparability.

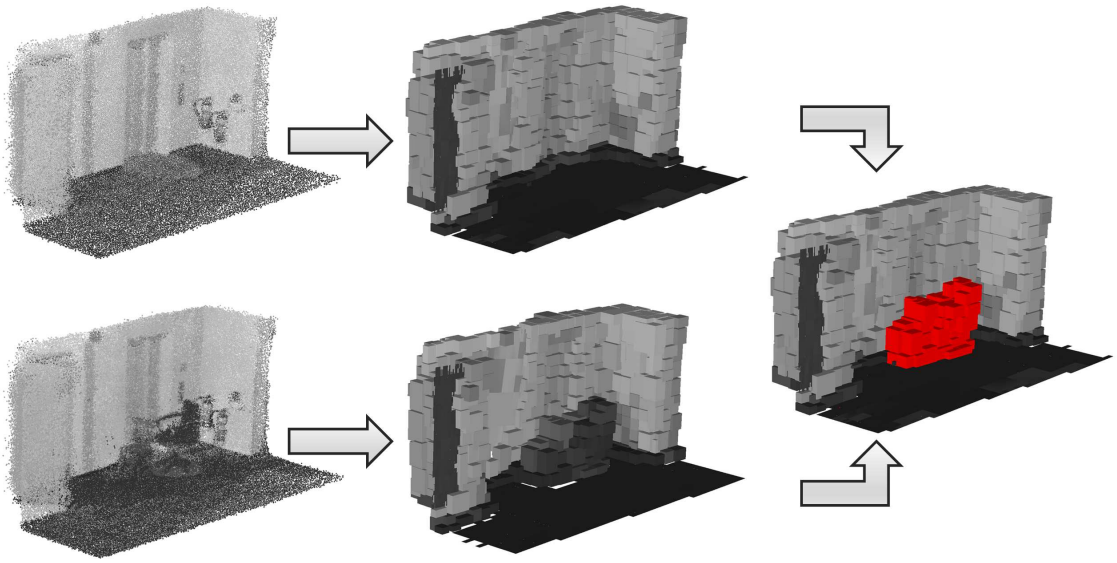


Figure 6.1: Example of a full model-based change detection.

6.2 Discussion

The evaluation demonstrates that the individual processing steps work differently depending on the field of application and are affected by each other. One significant influence on the detection of changes is the accuracy of the registration, as the 3D entities cannot be compared correctly in case of insufficient superposition. This leads to incorrect detections. The described method for registration based on AprilTags is suitable to transfer local coordinate systems into each other, independently of the change detection as presented in 5.2. It is recommended to get as close as possible to the respective tags with the IPS so that the tag can still be seen in both images. This allows a more accurate

transformation estimation by minimizing the influence of detection errors caused by the uncertainty of the pose. The examined application scenarios are measurements with trajectories of less than $10\ \mathrm{m}$ length. Therefore, no general statement on the performance can be made. An Euclidean transformation is calculated as approximation for the transformation. Therefore, an accumulated scaling error between the approached tags is not handled properly. A sectional improvement of the trajectories on the basis of redetected AprilTags may be suitable here. In general, the method is suitable for deeper integration into the processing chain of the IPS. In this case, however, a better evaluation is necessary to consider previously unobserved influences, such as the distribution of tags.

The next point focuses on the results of the model derivation. The model derivation is based on the separate consideration of planar and non-planar regions in the point cloud and thus designed for urban environments. The point clouds of the IPS are suitable for recognizing planes and simplifying them by the presented method. The use of the quadtree structure for simplification is advantageous for the modeling of rectangular surfaces. This cannot always be guaranteed, since the ground plane in particular can become extensive during longer runs. In this case, the definition of a maximum size of individual plane sections would be advisable. In the indoor datasets the proportions of the planes are sufficiently square. The evaluation shows that a QTD of 5 is adequate for a visually appealing result.

All areas that are not modeled as planes are approximated by bounding boxes. The results depend mainly on the selected cluster size and the structure of the input data. The super voxel clustering used for segmentation creates equally distributed clusters of similar size. However, this method does not completely avoid outliers, which can result in overlapping clusters and therefore overlapping bounding boxes. For a low-cost approximation, the bounding boxes are axis-aligned. An elongated area in a point cloud, which is not aligned, can cause artifacts in the shape of stairs. It is therefore important to ensure that the structure of the scene is as far as possible adapted to the coordinate system during processing. This can also be supported by aligning the IPS parallel to a wall during the measurement. Structures that are at an angle of $90°$ are thus well modeled. This phenomenon does not occur in plane modeling, since these are viewed independently of the initial coordinate system by means of the PCA transformation. The approximation using AABBs is not suitable for a pure optically appealing modeling, because the information about the structure of objects is lost, especially with larger cluster sizes. However, this design decision enables a fast change detection as presented in this work.

Examining the bounding boxes from one dataset for overlapping with the boxes from another dataset allows a fast classification. The mere occurrence of overlaps is sufficient to classify boxes as unchanged areas. Since even small overlaps are sufficient, an incorrect assignment and thus more false negatives occur more easily. Here the consideration of the volume or the intersection over union of the overlapping boxes may lead to an improvement. Exceeding a minimum volume as a threshold can reduce the number of false negatives, but also increase the number of false positives.

The neglect of the planar surfaces leads to runtime savings in the change detection, since in urban environments, indoor as well as outdoor, large parts do not have to be considered. This is based on the assumption that no large planar areas change. Application scenarios in which such changes occur cannot be checked with the described approach. Furthermore, measurements that differ spatially are not considered. If areas are scanned in a run that are not present in the reference run, these areas are classified as changes. This case is not under investigation. It is also possible that areas due to noisy measurements are modeled as planes in one dataset and as non-planer objects in the next dataset. Using the described method, change would be detected in this case. This can in principle be corrected by checking the intersections of the objects with the planes, but this is not a comprehensive solution either.

The evaluation of the execution time has shown that the approach is in principle real-time capable regarding specifications of the IPS. However, the current implementation is only applicable to the captured data at the end of a measurement run, since a complete point cloud is expected as input. An incremental derivation of the model on the basis of the measured points per time step or in a range is reasonable. Compared to the initialization time and model derivation, the time for change detection is negligible. This section-wise generated model can then be compared in real time with a previously loaded model.

The boxes are from a subjective perspective visually not appealing. The plane modeling is very promising and can be extended. Under certain circumstances, a separate use is advisable. The plane modeling can still be used for a visual and geometric simplification, while the boxes are only used for change detection. For a more visually appealing modeling of non-planar objects other methods are preferable.

6.3 Outlook

The developed methods show various possibilities for the further development of the IPS framework and its application areas. In addition, further potential tasks have emerged in the processing that could inspire future work. The plane modeling has turned out to be very promising and may be integrated into the processing chain as an independent implementation. Therefore, the approach described in [44] is an interesting starting point to abstract the planes incrementally and in real time. The remaining non-planar objects can be also added in sections. Points that are not modeled can be clustered and abstracted as boxes, points that fall in already existing boxes can be discarded. This depends on a negligibly small scaling error accumulated during the run.

The AprilTag detection and spatial localization can be used to correct the trajectory section by section between the individual tag detections. This can be done by referenced tags as well as by freely distributed tags. Another processing step to improve the model derivation is the texturing of the planes by images or by the colors of the dense point cloud. Moreover, for each plane the extent is known, therefore, as a next step in the work an adaptive Quad Tree Depth can be specified for each extracted plane. Thus, the parameter QTD can be replaced by the specification of a minimum resolution. This increases the usability, since no abstract value has to be specified anymore.

In the area of change detection, it is previously assumed that the areas considered are the same. However, this is not likely for hand-held scenarios. The implementation of a visibility map as presented in [31] may be interesting. Due to the calibration of the IPS, the acquisition frustum of the cameras is known for each image. From this, the areas under consideration can be derived.

Currently, the segmentation is based on super voxels. The integration of semantic information derived from some kind of object detection can be used to cluster points that belong to distinguishable objects. The extracted cluster can be approximated by bounding boxes as described in this thesis and therefore used for change detection. This also allows the detection of objects of certain types. Finally, the presented methods for registration, model derivation and change detection open up new application areas for the IPS that have not been considered yet. On the basis of this development, further work in these areas can make an important contribution to the development of camera based scanning systems.

Bibliography

[1] O. Vermesan and P. Friess, "EUROPEAN RESEARCH CLUSTER ON THE INTERNET OF THINGS", *IERC*, p. 142, 2015. [Online]. Available: http://www.internet-of-things-research.eu/pdf/IERC_Position_Paper_IoT_Standardization_Final.pdf.

[2] A. Börner, D. Baumbach, M. Buder, A. Choinowski, I. Ernst, E. Funk, D. Grießbach, A. Schischmanow, J. Wohlfeil, and S. Zuev, "IPS-a vision aided navigation system", *Adv. Opt. Techn*, vol. 6, no. 2, pp. 121–129, 2017. DOI: 10.1515/aot-2016-0067. [Online]. Available: www.degruyter.com/aot.

[3] M. G. Wing, A. Eklund, and L. D. Kellogg, "Consumer-grade Global Positioning System (GPS) accuracy and reliability", *Journal of Forestry*, vol. 103, no. 4, pp. 169–173, 2005, ISSN: 0022-1201.

[4] W. Xiao, "Change detection from mobile laser scanning point clouds", PhD thesis, General Mathematics Université Paris-Est, 2015. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01373359.

[5] A. Watt, *3D Computer Graphics*. 2000, ISBN: 9780521821032.

[6] D. Haupt, *Fire Extinquisher Model*, https://3dhaupt.com/3d-tools-here-you-can-check-out-some-of-my-3d-tools/fire-extinguisher-downloads (visited 09.12.2019), 2013.

[7] C. Ericson, *Real-Time Collision Detection*. 2004. DOI: 10.1201/b14581.

[8] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching", *Communications of the ACM*, 1975, ISSN: 15577317. DOI: 10.1145/361002.361007.

[9] G. Turk and M. Levoy, *3D test model created from scanning a ceramic figurine of a rabbit*, 1994.

[10] A. R. Large and G. L. Heritage, *Laser Scanning for the Environmental Sciences*. 2009, ISBN: 9781444311952. DOI: 10.1002/9781444311952.

[11] F. Remondino, "From point cloud to surface: the modeling and visualization problem", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2003. DOI: 10.3929/ethz-a-004655782.

[12] J. Berkmann and T. Caelli, "Computation of Surface Geometry and Segmentation Using Covariance Techniques", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1994, ISSN: 01628828. DOI: 10.1109/34.334391.

[13] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces", in *Proceedings of the IEEE Visualization Conference*, 2002.

[14] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane", *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, Jul. 1983, ISSN: 0018-9448. DOI: 10.1109/TIT.1983.1056714.

[15] J. F. Hughes, A. van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner, and K. Akeley, *Computer graphics: principles and practice (3rd ed.)* Boston, MA, USA: Addison-Wesley Professional, Jul. 2013, p. 1264, ISBN: 0321399528.

[16] Khronos Group, "The OpenGL Graphics System: A Specification", Tech. Rep., 2017, p. 805. [Online]. Available: https://www.khronos.org/registry/OpenGL/specs/gl/glspec45.core.pdf.

[17] O. Schreer, *Stereoanalyse und Bildsynthese*, 50. 2005, vol. 57, pp. 1–74, ISBN: 3-540-23439-X. DOI: 10.1159/000142566. [Online]. Available: http://link.springer.com/content/pdf/10.1007/3-540-27473-1.pdf.

[18] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University Press, ISBN: 0521540518, 2004.

[19] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008, ISSN: 01628828. DOI: 10.1109/TPAMI.2007.1166.

[20] I. Ernst and H. Hirschmüller, "Mutual information based semi-global stereo matching on the gpu", in *ISVC08*, Jan. 2008. [Online]. Available: https://elib.dlr.de/55655/.

[21] E. Olson, "AprilTag: A robust and flexible visual fiducial system", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2011, pp. 3400–3407.

[22] F. Bovolo and L. Bruzzone, "The time variable in data fusion: A change detection perspective", *IEEE Geoscience and Remote Sensing Magazine*, vol. 3, no. 3, pp. 8–26, Sep. 2015, ISSN: 2373-7468. DOI: 10.1109/MGRS.2015.2443494.

[23] J. Campbell and R. Wynne, *Introduction to Remote Sensing*. Jan. 2011, ISBN: 978-1-60918-176-5.

[24] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces", *arXiv preprint arXiv:1906.05797*, 2019.

[25] L. Ma, T. Whelan, E. Bondarev, P. H. N. De With, and J. Mcdonald, "Planar Simplification and Texturing of Dense Point Cloud Maps", Tech. Rep., 2013. [Online]. Available: http://www.thomaswhelan.ie/Ma13ecmr.pdf.

[26] T. Whelan, M. Goesele, S. J. Lovegrove, J. Straub, S. Green, R. Szeliski, S. Butterfield, S. Verma, and R. Newcombe, "Reconstructing scenes with mirror and glass surfaces", *ACM Transactions on Graphics*, 2018, ISSN: 15577368. DOI: 10.1145/3197517.3201319.

[27] F. A. Limberger and M. M. Oliveira, "Real-time detection of planar regions in unorganized point clouds", *Pattern Recognition*, 2015, ISSN: 00313203. DOI: 10.1016/j.patcog.2014.12.020.

[28] A. Kaiser, J. Alonso, Y. Zepeda, and T. Boubekeur, "A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data", Tech. Rep., 2018. [Online]. Available: https://perso.telecom-paristech.fr/boubek/papers/GeoPrimFitSurvey/GeoPrimFitSurvey.pdf.

[29] L. J. Manso, P. Núñez, S. da Silva, and P. Drews-Jr, "A Novel Robust Scene Change Detection Algorithm for Autonomous Robots Using Mixtures of Gaussians", *International Journal of Advanced Robotic Systems*, vol. 11, no. 2, p. 18,

Feb. 2014, ISSN: 1729-8814. DOI: 10.5772/57360. [Online]. Available: http://journals.sagepub.com/doi/10.5772/57360.

[30] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017, ISBN: 9781509046331. DOI: 10.1109/ICRA.2017.7989591.

[31] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault, "Change detection on points cloud data acquired with a ground laser scanner", in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 2005.

[32] B. J. A. Gunnar, F. Marco, G. João, P. A. David, S. Vitor, C. Bernard, K. Richard, M. Frederic, R. C. Michel, and Z. Patrick, "3d reconstruction in nuclear security", 2008.

[33] W. Xiao, B. Vallet, and N. Paparoditis, "Change Detection in 3D Point Clouds Acquired by a Mobile Mapping System", in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, Copernicus GmbH, Oct. 2013, pp. 331–336. DOI: 10.5194/isprsannals-II-5-W2-331-2013.

[34] *Trimble tx6 3d lasers scanner*, https://mepsupport.trimble.com/products/scanning/trimble-tx6-3d-laser-scanner (visited 10.09.2019).

[35] A. Taneja, "Geometric Change Detection and Image Registration in Large Scale Urban Environments", PhD thesis, 2014, ISBN: 8610828378018. DOI: 10.3929/ethz-a-010782581. [Online]. Available: https://doi.org/10.3929/ethz-a-010025751.

[36] A. Taneja, E. Zurich, L. Ballan, and M. Pollefeys, "Image Based Detection of Geometric Changes in Urban Environments", Tech. Rep., 2011. [Online]. Available: https://inf.ethz.ch/personal/pomarc/pubs/TanejaICCV11.pdf.

[37] E. Palazzolo and C. Stachniss, "Fast image-based geometric change detection given a 3D model", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018, ISBN: 9781538630815. DOI: 10.1109/ICRA.2018.8461019.

[38] H. Andreasson, M. Magnusson, and A. Lilienthal, "Has somethong changed here? Autonomous difference detection for security patrol robots", in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2007, pp. 3429–3435, ISBN: 978-1-4244-0911-2. DOI: 10.1109/IROS.2007.4399381. [Online]. Available: http://ieeexplore.ieee.org/document/4399381/.

[39] L. Ma, R. Favier, L. Do, E. Bondarev, and P. H. De With, "Plane segmentation and decimation of point clouds for 3D environment reconstruction", Tech. Rep., 2013, pp. 43–49. DOI: 10.1109/CCNC.2013.6488423.

[40] K. Pearson, "On lines and planes of closest fit to systems of points in space", *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. DOI: 10.1080/14786440109462720.

[41] L. P. Chew, "Constrained delaunay triangulations", in *Proceedings of the Third Annual Symposium on Computational Geometry*, ser. SCG '87, Waterloo, Ontario, Canada: ACM, 1987, pp. 215–222, ISBN: 0-89791-231-4. DOI: 10.1145/41958.41981.

[42] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, "Voxel cloud connectivity segmentation - supervoxels for point clouds", in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, Portland, Oregon, Jun. 2013.

[43] P. Irmisch, D. Baumbach, I. Ernst, and A. Borner, *Simulation framework for a visual-inertial navigation system*, Sep. 2019. DOI: 10.1109/ICIP.2019.8803187.

[44] T. Whelan, L. Ma, E. Bondarev, P. H. De With, and J. McDonald, "Incremental and batch planar simplification of dense point cloud maps", Tech. Rep. 1, 2013, pp. 3–14. DOI: 10.1016/j.robot.2014.08.019. [Online]. Available: http://www.thomaswhelan.ie/Ma13ecmr.pdf.