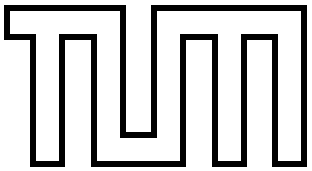


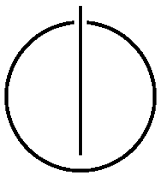
FAKULTÄT FÜR INFORMATIK

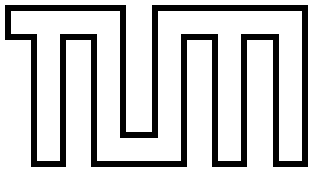
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

Fusion of Time-of-Flight Plane Features with Point Features for Camera-Pose Tracking

Sebastian Marsch





FAKULTÄT FÜR INFORMATIK

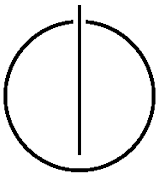
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

Fusion of Time-of-Flight Plane Features with Point Features for Camera-Pose Tracking

Bestimmung der Kameratrajektorie anhand von Ebenenmerkmalen einer Time-of-Flight-Kamera in Kombination mit Punktmerkmalen

Author:	Sebastian Marsch
Supervisor:	Prof. Gudrun Klinker, Ph.D.
Advisor:	Dipl.-Inf. Christian Waechter
Date:	September 15, 2011



I assure the single handed composition of this master thesis only supported by declared resources.

Munich, September 15, 2011 Sebastian Marsch

Acknowledgments

My gratitude goes to my thesis advisor Christian Waechter. He always had an open ear for my concerns.

Furthermore, I want to thank Axel Stamm for the employment as software engineer at Synatec GmbH during my academic studies.

Thanks go to Jürgen Sotke and Julia Reizlein for advices on writing.

Finally, I want to thank my parents for offering me support whenever I need it.

Thank you!

Abstract

The objective of this master thesis is to estimate the pose change of a time of flight (TOF) camera between consecutive frames in all six degrees of freedom. To gain visual high-resolution information of the scene, the TOF camera is combined with an RGB camera. The proposed algorithm fuses 3D geometric and 2D visual features. Planar surfaces are extracted directly from the 3D point cloud while SURF is applied to the 2D projections of these surfaces. Feature positions and plane equations are both used to estimate all six degrees of freedom of the camera motion. The algorithm outperforms fast coarse pose registrations, that do not combine the 3D geometry with visual projections in accuracy while it is suitable for online processing. Fine registrations, that use the complete point cloud are more precise but also much more time consuming.

Contents

Outline of the Thesis

CHAPTER 1: INTRODUCTION

This chapter presents the purpose of the thesis. Furthermore, a short comparison of different 3D imaging technologies is given.

CHAPTER 2: RELATED WORK

Two state of the art algorithms for point cloud registration and various related, recently published research is presented here.

CHAPTER 3: TIME OF FLIGHT (TOF) CAMERAS - WORKING PRINCIPLE AND PROPERTIES

The physical working principle and the properties of Time of Flight cameras are outlined.

CHAPTER 4: MECHANICAL SETUP AND CALIBRATION

The sensor attachment and its coordinate systems are described first. How this coordinate systems are registrated to each other, is described in the calibration part of this chapter. Determination of 3D points from the depth map is also descibed here.

CHAPTER 5: PLANE RECOGNITION

The underlying representation of planes and the estimation of its parameters is the topic of this chapter, as well as the detection and segmentation of planes in the 3D range data.

CHAPTER 6: VISUAL FEATURE EXTRACTION ON THE TEXTURES OF THE PLANES

Two popular feature detectors are compared. It is described, how the feature search can be restricted to the textures of the detected planes and how the 3D positions of the visual features can be refined by these planes.

CHAPTER 7: POSE ESTIMATION

Two matching techniques to find correspondences of feature descriptors between two frames are presented here, as well as the identification of false correspondences. The estimation of the pose using 3D feature point correspondences and using a fusion of these correspondences and the plane equations is described.

CHAPTER 8: EVALUATION

The evaluation methods and the results are listed. Rotation angle, translation distance and processing time have been evaluated and its results are discussed.

CHAPTER 9: SAMPLE APPLICATIONS

Two sample applications have been implemented and are demonstrated here: 3D map generation and the augmentation of a cube.

CHAPTER 10: CONCLUSION AND OUTLOOK

This chapter points the features of this work out.

1 Introduction

The objective of this work is to track the motion of a sensor, that consists of a time of flight (TOF) camera and an RGB webcam. That is the pose change of the sensor between two consecutive frames. Concretely, the aim is to estimate the pose ${}^{TOF_{i-1}}T_{TOF_i}$ of the measurement coordinate system TOF_i of the time of flight camera of frame i in the coordinate system TOF_{i-1} of frame i-1.

The estimation of a cameras motion is one of the most significant issues of Computer Vision. Since computers became affordable for research facilities and companies, they have been used to process digital 2D images. And since this time, the 3D movement of sensors consisting of one or more 2D cameras became an important topic of research, as such information opened new fields of application. Be it the examination of trajectories, be it the 3D reconstruction of scenes, or be it augmented reality just to mention a few of them. Until the market launch of 3D imaging technologies, the basis to solving for pose changes in six degrees of freedom (that implies three for rotation and three for translation) had been the 2D projections of the 3D world. Until such 3D range imaging devices are available, the way round to 2D had to be accepted inevitably. Beside TOF cameras, there exist several other devices that generate range data: Light sectioning sensors, 3D laser scanners and stereo or multiple camera systems with or without fringe pattern projection and light coding cameras.

The first mentioned technology is just measuring a distance profile of the scene part that intersect a measurement plane. To generate a depth map and thus a mapping of the 3D profile of a scene, this device has to be moved precisely on a known trajectory to capture one frame. 3D laser scanners are also in fact just 2D laser scanners, that measure the intersection of the scene with a plane. They are rotated for data acquisition. Stereo cameras (as well as three or more cameras), that use a fringe pattern as artificial landmark, have to stay on a fixed position for image acquisition, as long as the projector needs to generate all patterns. Just camera systems without fringe pattern projections and light coding cameras are able to capture the scene in one measuring step and thus can be moved arbitrarily during the acquisition process.

All technologies, that use multiple cameras or a combination of a camera with a projection device do not produce full depth maps due to occlusion. Stereo camera systems also cannot reconstruct homogeneously textured parts of the scene. Depth maps, that are generated by these devices contain regions with no range information, so called gaps. Interpolation is often used to fill this gaps but nevertheless it is missing information. Only 3D laser scanner produce depth maps without gaps. Table 1.1 gives an overview of the attributes of above mentioned devices.

Every technology has its popular applications but as can be seen from table 1.1, just TOF cameras generate full depth maps and can be used in applications with moving sensors. A further big advantage is their high frame rate compared to the other described systems.

On the other hand their resolution and precision is rather low. Thus it is a challenging

	depth map with gaps	depth map without gaps
not movable	stereo cameras with fringe projector light sectioning sensors	3D laser scanner
movable	stereo camera without fringe projector light coding cameras	TOF cameras

Table 1.1: Comparison of 3D imaging technologies.

task to develop algorithms, that work on this data. But it allows to realize new applications, that failed due to the restrictions of the other above mentioned sensor systems as for example gapless 3D mapping with a free-hand moved sensor.

Algorithms, that use the complete 3D point clouds (like ICP) instead of a few 3D interframe point correspondences that have been identified via visual features, are very precise. But they require a good initial pose (see section 2.1.1). This can be granted in two ways: By a small pose change between the point clouds. Thus this algorithm is popular for systems, that capture data with high frame rates and do the processing offline (see section 2.2). Or by applying a preregistration of the point clouds which can be done for example with 4PCS (see section 2.1.2). These preregistrations are very time consuming due to their high complexity and would require some seconds per frame. Thus such techniques are just applied in systems, that capture data offline and process it afterwards.

On the other hand there exist multimodal systems, that combine visual features with depth data. They can be used to estimate large pose changes with low processing times but also with low accuracy. They detect and match visual features in 2D images and determine the pose change by correspondences between the 3D coordinates of the found features. This feature detection is done regardless of the 3D geometry of the objects and because of the low accuracy of TOF cameras the so determined pose changes also lack accuracy.

I propose a novel algorithm, that uses planar features of the 3D point cloud as well as 2D visual features and their 3D positions to estimate the pose. Planes have been chosen, because they are fast to detect. Indoor environments usually contain planes like for example the flour, walls or table plates. Because these are 2D surfaces, they encompass many points which leads to a compensation of the measurement noise of single depth values of the TOF camera. Thus the algorithm reaches a higher accuracy than other coarse registrations using pure 3D positions of visual features and does not need much more processing time. Even the algorithm itself benefits from the use of planes, since this reduces the number of iterations of the outlier detection by RANSAC enormously as can be seen in section 7.4. This makes the algorithm suitable for applications like online 3D mapping (see section 9.1) or augmented reality (see section 9.2).

2 Related work

2.1 Algorithms for point cloud registration

Two algorithms, that registrate 3D point clouds, are presented in this section. The ICP algorithm is a popular method for fine registration. High accuracy can be reached, but it is only capable for small pose changes. On the other hand, 4PCS is used to registrate two point clouds with arbitrary pose. It can be used as preprocessing step to ICP.

2.1.1 The ICP algorithm

The iterative closest point (ICP) algorithm is a popular algorithm to registrate two point sets P and Q. An outline can be found in Besl et al. [7].

The algorithm is an iterative process, that refines the registration of the two point sets with every iteration. It works as follows:

1. Find a nearest point $q \in Q$ for every point $p \in P$.
2. Estimate the transform that converges P to Q by the method of Horn [15]. Apply this transform to the points of P.
3. Find point correspondences like in step 1. If the average distance of the corresponding points is above a threshold, go back to step 1. Else stop the iteration. The overall transform is composed by all estimated transforms of step 2.

If the sets P and Q do not overlap to 100% step 1 finds wrong point correspondences for the non-overlapping areas, because for every point of P a point of Q will be assigned. Some modifications to the ICP algorithm exist to overcome this problem. The so called Vanilla-ICP [23] discards the correspondence (p_i, q_j), if the euclidean distance between p_i and q_j exceeds a threshold. Another modification is described in section 2.2.

There exist also some modifications to the ICP algorithm to accelerate it. One, that yields good results is published by Jost et al. in [19]. It accelerates step 1 of the ICP algorithm by limiting the search region for corresponding points as follows: if a point correspondence was already found in the neighborhood of p_i in P, q_j is searched in the neighborhood of this correspondence in Q. In the best case, this modification improves the complexity of ICP to $O(n)$. If ICP would have found a different corresponding point for p_i, this modification of the algorithm yields just approximative results to ICP. This fast version is only profitable, if the neighbors of points can be found fast. For point sets that have been derived from range images, the 3D neighborhood of a point can be approximated by the points that occur from the neighboring pixel in the 2D range image.

If the initial pose difference between P and Q is too high, step 1 does not find correct point correspondences. This leads to a wrong transform estimate in step 2 which let the two point sets drift apart or let the algorithm stuck in a local minimum.

2.1.2 The 4PCS algorithm

The 4-points congruent sets (4PCS) algorithm was published by Aiger et al. [2]. In contrast to the ICP algorithm, it is able to registrate two point clouds that have completely arbitrary poses. It can be used as preprocessing step to ICP, to find a good initial transformation, because the registration accuracy of ICP is better than the accuracy of 4PCS.

Consider four coplanar but not collinear 3D points A, B, C and D that are aligned as illustrated in figure 2.1. The intermediate point E is the intersection of the lines AB and CD. 4PCS is based on the fact, that the two ratios

$$r_1 = \frac{\|A-E\|}{\|A-B\|},\ r_2 = \frac{\|C-E\|}{\|C-D\|} \tag{2.1}$$

are invariant under affine transformations.

The algorithm searches four coplanar points in the set P and calculates the ratios r_1 and r_2. Then it calculates the possible two intermediate points

$$e_1 = p + r_1(q-p),\ e_2 = p + r_2(q-p) \tag{2.2}$$

for each possible pair (p,q) of all points $p \in P$ and $q \in Q$. Every couple of point pairs with coincident intermediate points (one resulting from r_1, the other from r_2) can be assumed as affine transformed version of the four coplanar points of P. The transformation of P to Q can than be calculated with a closed form solution as described by Horn et al. in [15].

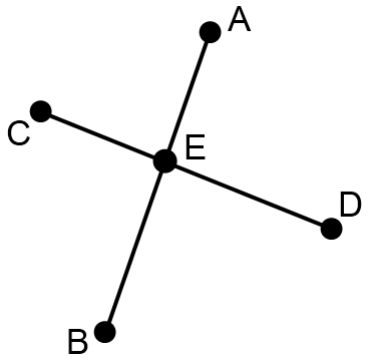


Figure 2.1: Physical components of a time of flight camera

With the use of RANSAC the algorithm becomes robust to outliers. The complexity of 4PCS is $O(n^2)$ and it is only suitable as preprocessing step to ICP as mentioned before. Computing times of the 4PCS for point sets of different sizes on a common PC have been published in [2]. From this listing can be derived, that the processing time of 4PCS for the registration of range data which is used in this work would be at least one second. Thus the algorithm is not suggested for online registration of point clouds.

2.2 Pose estimation of Time Of Flight cameras

Several algorithms have been developed to estimate poses of TOF cameras for mainly two applications: map building and controlling of trajectories of mobile robots. Both are

existing, systems using a TOF camera only and multimodal systems. A short overview of recently published research is presented in the following.

Droeschel et al. [24] designed a sensor system for measuring the trajectory of a mobile robot in 3 degrees of freedom: The robot is able to move on the ground in x- and y-direction and to rotate around its z-axis. They used a TOF camera in combination with sensors for linear and rotational acceleration. The data of this accelerometers is used to improve the estimated pose change between two consecutive frames. The pose change itself is calculated as the transformation between two sets of the 3D coordinates of corresponding feature points. The feature points are found with the SIFT algorithm and matched with the features of the previous frame. They demonstrated that the use of accelerometers enhances the accuracy of the pose change estimate.

A path reconstruction system for a mobile robot using a TOF camera was published by Droeschel et al. in [23]. It was also developed for a robot that moves just in x- and y-direction and rotates around its z-axis. Data is captured during the move of the robot and afterward it is evaluated offline. Thus data can be captured with high frame rates. That results in small pose changes between successive frames which enables them to apply the ICP algorithm directly on the 3D point clouds of the TOF camera. They compared two different versions of ICP: the Vanilla-ICP (see section 2.1.1) and the Frustum-ICP which discards points that are not in the frustum of the TOF camera in both poses. The frustum of the TOF camera has to be updated in every iteration step of the ICP.

Ohno et al. [26] presented a system for trajectory estimation of a mobile rescue robot and map construction using a TOF camera and an rotational accelerometer. Here the robot moves on the ground, too. The data was transmitted to a PC via LAN during the movement of the robot. The authors modified the ICP algorithm to make it usable for online frame-to-frame registration. They applied the Prewitt operator to the amplitude image of the TOF camera and used only the range data of the related pixel with edge responses in the amplitude image. This leads to an enormously reduction of point correspondences in step 1 of the ICP algorithm (see section 2.1.1) and would lead to an increase of its iteration number. They overcame this problem by delivering an inertial guess to the ICP. The inertial rotation is calculated from accelerometer data and the initial translation between the range data of the previous frame and the rotated range data of the current frame by the shift of their centroids. They reached frame rates of 4 fps. The frame-to-frame errors of rotation are 1% to 15% and of translation 15% to 17%.

A system that uses only a TOF camera for pose registration and is able to handle large pose changes is demonstrated by Swatzba et al. in [29]. The authors divide the point cloud registration in to steps: a coarse registration followed by a fine registration. In the first step, visual features are extracted in both, the amplitude image and the depth map of the TOF camera. The 3D positions of this features are used for the coarse registration, which is calculated by a closed form solution using singular value decomposition. The coarse registration delivers a initial pose for the second step. The fine registration is then performed using a modified version of the ICP algorithm. The step 1 of the ICP is done on the points that emerge from a region of interest of the depth map. This region is determined using the intrinsic parameters of the TOF camera and projecting the 3D point, for which a corresponding point is searched, into the depth map of the other frame. Also correspondences to points that are multiple assigned are rejected. This restrictions on the correspondence search can cause the algorithm to not converge any more. In this case,

the algorithm is stopped after many iterations and the registration accuracy is low. The algorithm was evaluated with two different TOF cameras, one with 64x16 and one with 120x160 pixel resolution and they evaluated just for translation errors. Depending on the scene and the type of the camera they reached average translation errors between 5mm and 85mm. Using the Camera with the high resolution they measured processing times of 3041 ms per frame for the normal ICP and 242ms per frame for the accelerated version of the ICP.

Huhle et al. introduced a multisensor system in [17] that is directly comparable with this work because it uses a combination of TOF camera and RGB camera. Instead of a low resolution RGB webcam they used a 1600x1200 industrial color camera. They are using the RGB image for SIFT feature detection. The detected features are used to calculate a coarse registration. For fine registration they are minimizing the sum of two energy functions: One derives from the normal distributions transform (NDT) which uses gaussian point distributions of local structures of the point cloud for registration. The other is a measure of the distances between the 3D feature positions of the 2D SIFT features under a certain transform. Both energy functions can be differentiated analytically which allows the use of a nonlinear optimization method to solve for the transform parameters. Unfortunately they did not publish an evaluation of the accuracy of this algorithm. The processing time of one frame was between 2 and 3 seconds. Thus this algorithm is not suggested for online registration.

3 Time Of Flight (TOF) Cameras - working principle and properties

A brief overview of the working principle of time of flight cameras is given in this chapter. There are two kinds of time-of-flight cameras. One uses continuous modulated light and the other uses light pulses. A TOF camera with continuous modulated light has been used in this work, so this technology is explained in detail here. The presented overview includes a description of the physical components and their alignment as well as a description of their output data, properties, restrictions and artifacts. The camera model SR4000 is explained in detail, because it is the underlying hardware for the presented work.

3.1 Physical components of time of flight cameras

Figure 3.1 illustrates the physical setup of a time of flight camera. It consists of an infrared camera with adjusted infrared bandpass filter, an infrared led illumination and a processor to calculate the depth map.

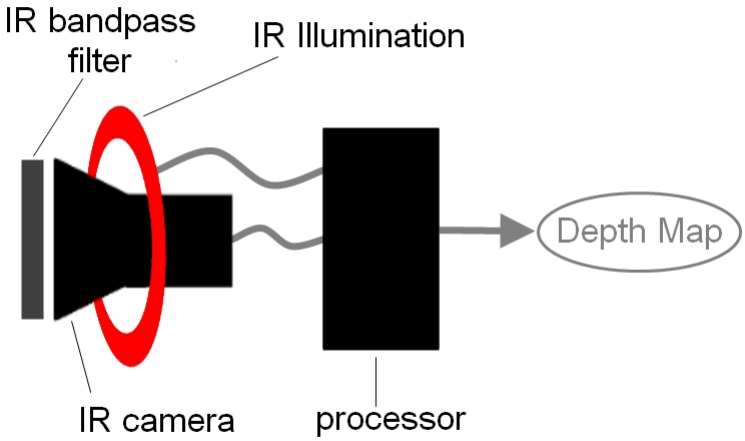


Figure 3.1: Physical components of a time of flight camera

3.2 Working principle of TOF cameras with continuously modulated light

The illumination LEDs produce infrared light of a single constant frequency f_m. This light wave is emitted into the scene and reflected by its objects (see figure 3.2).

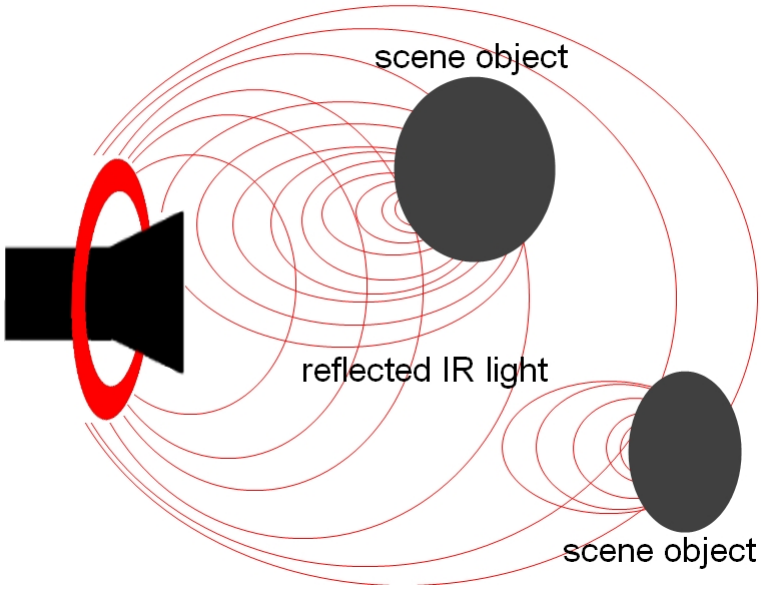


Figure 3.2: Working principle of a TOF camera: reflected light

The reflected light needs less time for near objects to reach the infrared camera than for far away objects. The camera captures four sample images $S_i ... S_{i+3}$ in one modulation period $1/f_m$ of the emitted light (see figure 3.3). Each sample image is the result of an integration of the photoelectric current of a time interval $\Delta t \leq 1/f_m$. To increase the signal to noise ratio, the samples are the integration of several thousand modulation periods [9].

For every pixel the reflected light wave is reconstructed by these four sample points in the processor. In this way the phase shift between the transmitted and received light is calculated for every pixel which means for every sight ray. The phase shift is proportional to the flight time of the light from the illumination source to the camera image plane. So for every pixel the distance between the illumination-camera setup and the nearest object crossing the sight ray can be calculated by

$$D = \frac{c}{4\pi f_m}\phi \tag{3.1}$$

where c denotes the speed of light, f_m denotes the modulation frequency of the illumination and ϕ the phase shift. As per [9] the distance can be calculated directly from the sample images $S_0 ... S_3$ by

$$D = \frac{c}{4\pi f_m} \cdot atan\left(\frac{S_1 - S_3}{S_0 - S_2}\right). \tag{3.2}$$

As illustrated in figure 3.3 the amplitude A' of the received light wave is less than the amplitude A of the emitted light wave. This is because of absorption of light on the scene's

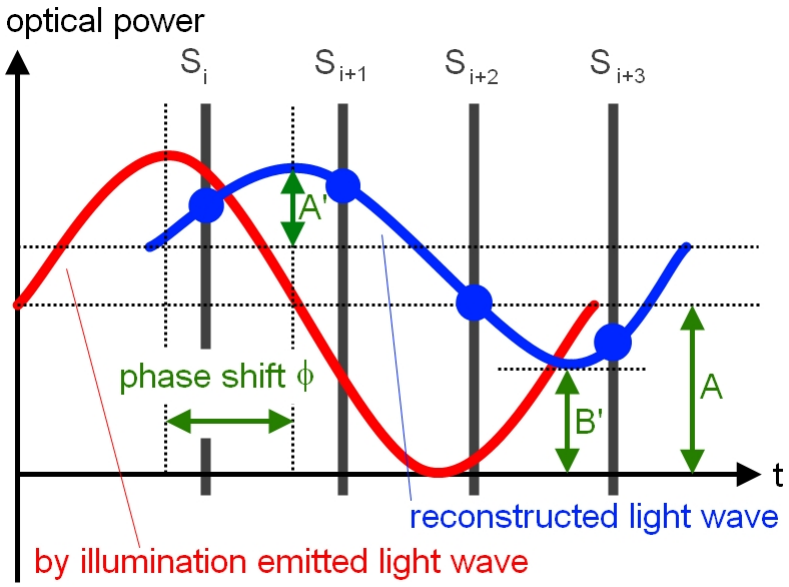


Figure 3.3: Working principle of a TOF camera: phase shift measure

objects and because of light diffusion and attenuation. Also there is an offset B' in the received signal due to background illumination. As the value A' represents mainly the absorption on material, it can be used to generate a gray value image of the scene. It can also be used to measure the quality of the distance measurement, since the bigger A' is the bigger is the signal to noise ratio. A high amount of background light and a intense reflected signal cause over-saturation regarding the conversion process which can also be dedicated from the values A' and B'. If B' is too high, the background illumination is too bright (for example due to sun light) and should be reduced. If A' is too high, objects are too specular or reflectable like mirrors or metallic surfaces. The camera can use this information to derive a confidence map, which indicates for every pixel, whether the signal to noise ratio or the saturation is out of scope.

3.3 Technical restrictions of time of flight cameras

Due to the physical working principle of TOF cameras there are effects and restrictions in imaging which should be known when using this cameras. This section describes all important issues about these limitations and gives implications for a correct usage of this type of cameras. Only TOF cameras with continuously modulated light are considered.

3.3.1 Limited range

Depth estimation is done by calculating the phase shift between transmitted and received light. Therefor it is not possible for the camera to obtain the depth correctly, if objects are more than half of the wavelength away from the TOF camera. Otherwise the light would need more than the period time to flight back and the processor would not be able to

assign the correct phase shift. Measured phase shifts are always in the scope of $[0, 2\pi]$ and if the real phase shift is greater, it is folded back to this interval. By this reason it should be assured, that there are no scene objects more far away than this range. The range is calculated by

$$D_{max} = \frac{c}{2f}. \tag{3.3}$$

For example, if a modulation frequency of 30MHz is used, the distanced between camera and objects should not transcend 5 meters.

3.3.2 Movement artifacts

Each of the four samples described in section 3.2 is taken as separate capture. To estimate the phase shift, it is important, that the reflection point did not move during the exposure of this four samples. If an object or the camera moves very fast, systematic errors are introduced to the measurement. Thus it is important, not to move the camera or objects fast.

3.3.3 Dependence of the measurement accuracy on object properties

The confidence and accuracy of the measurement depends on the object surface properties. It can be distinguished between 2 surface types, which cause a quality reduction of the depth values.

- Absorbing surfaces:
  Dark surface absorb a big amount of the transmitted infrared light. Hence the amplitude of the received light (A' in figure 3.3) and the signal to noise ratio are low. A low signal to noise ratio causes a low repeatability of the measured values.

- Specular surfaces:
  Surfaces with high reflectivity cause the reflected light to have a high amplitude (A' in figure 3.3). This causes over-saturation and the range of conversion data is out of the domain of the analog to digital converter. Over saturated pixel are indicated by a confidence map. They should not be considered in the algorithms.

3.3.4 Problems caused by background illumination

Background illumination in the infrared band cause a shift (B' in figure 3.3) in the received signal. In this way the level of the measured analog values is increased, which makes measurement data sensible to over saturation. For bright surfaces, depth values might then not be estimated.

3.3.5 Multiple reflections

Depth value reconstruction works under the assumption, that the light moves directly from the light source to the object and back to the camera. In practice, light rays are reflected several times, until they reach the image plane of the camera (see figure 3.4). Because paths along multiple reflections are longer, the phase shift of this light will be

greater as well as the recognized depth value. One single reflection gives the optimum depth value accuracy. To avoid multiple reflections, situations as illustrated in figure 3.4 should be avoided.

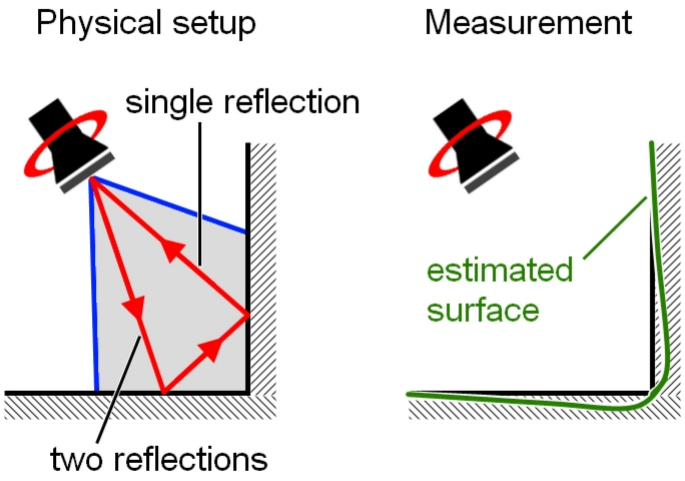


Figure 3.4: Artifacts due to multiple reflections

3.4 The SR4000

Multiple vendors are supplying TOF cameras, most of them are using continuously modulated light. The leading vendor is the swiss company Mesa Imaging. In this work, the Mesa SwissRanger SR4000 (see figure 3.5) has been used. This section gives a quick survey of the details of this camera, which are important to this work. The here presented dates are manufacturer informations and taken from the data sheet.

Figure 3.5: The SwissRanger SR4000: rear and front view

3.4.1 Hardware characteristics and image acquisition details

- Maximum frame rate:
  The SR4000 achieves a maximum frame rate of 54 fps.

- Pixel resolution:
  176 x 144

- Available modulation frequencies and the resulting ranges:
  The camera allows the use of eight different modulation frequencies. Equation 3.3 defines the maximum range for each of this frequencies. The available frequencies reach from 10 MHz to 60 MHz and the relating ranges from 2.5 m to 15 m respectively. They are listed in table 3.1.

modulation frequency	maximum range
60 MHz	2.5 m
31 MHz	4.84 m
30 MHz	5 m
29 MHz	5.17 m
15.5 MHz	9.68 m
15 MHz	10 m
14.5 MHz	10.34 m
10 MHz	15 m

Table 3.1: Available modulation frequencies and the related ranges of the SR4000.

- View angle:
  The standard type of the SR4000 has a view angle of 43.6° in X-direction and 34.6° in Y-direction of the TOF coordinate system. This angle is very small compared to standard RGB cameras, which results in the fact, that when combining these two camera types, the TOF camera sees only a subset of the frustrum of the RGB camera.

- Absolute accuracy:
  Mesa declares the absolute accuracy of the SR4000 model as $\pm 10mm$ or $\pm 15mm$, depending on the camera subtype. This values are only valid in a calibrated range of 0.8 m to 8 m. Also there are temperature drifts in depth measurement of up to $1.5mm/°C$.

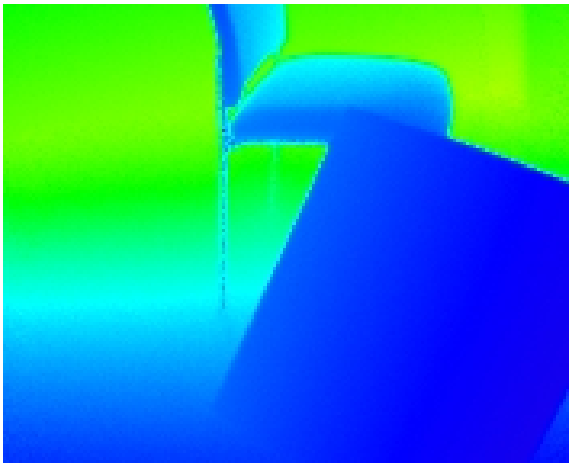
- Repeatability:
  The specified repeatability depends on the measurement region. If the considered pixel is near to the principal point, a value of $\sigma \leq 120\%$ of the maximal depth value is specified. Else $\sigma \leq 200\%$ is given.

3.4.2 Sensor measurements

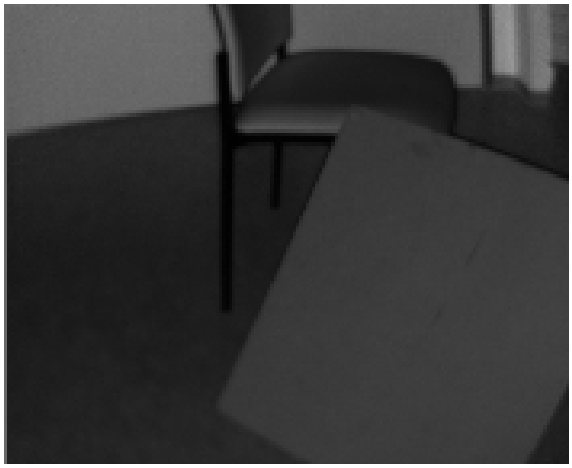
Figure 3.6 shows the three measurement images of the Mesa Swissranger SR4000. The scene is illustrated in figure 3.6(a). The depth map (figure 3.6(b)) is printed with false color

rendering. If object points are near to the camera they are blue, if they are in the middle of the range of the TOF camera they are green and if they are at maximum range, they are red. Also the confidence map (figure 3.6(d)) is presented with false colors. Pixel with low confidence (see section 3.2) are painted in red. The amplitude image is shown in figure 3.6(c).

(a) Scene



(b) Depth map



(c) Amplitude image

(d) Confidence map

Figure 3.6: Sensor measurements of the SR4000

4 Mechanical setup and calibration

4.1 Mechanical setup

A combination of a Time-Of-Flight camera and a RGB camera is used in this work. To evaluate the accuracy of the estimated pose, a system of the company ART was provided to track rigid bodies. Because every device defines at least one coordinate system and the registration of these coordinate systems is calibrated and must stay fixed after calibration a mounting was used which enables a solid attachment of this three devices.

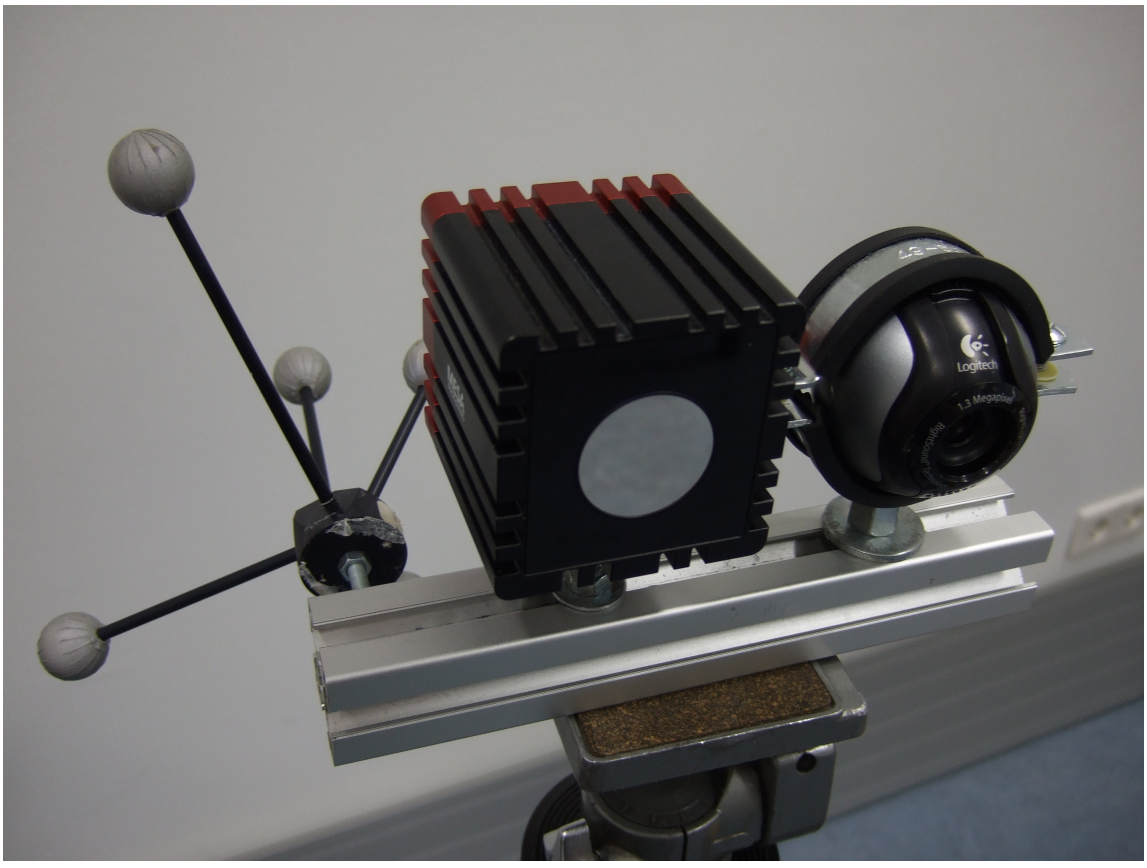


Figure 4.1: Mounted components: front view

Figure 4.1 shows how the devices are assembled to each other. On the back side (see figure 4.1) the rigid body consisting of infrared light reflecting balls of the ART measurement system can be seen.

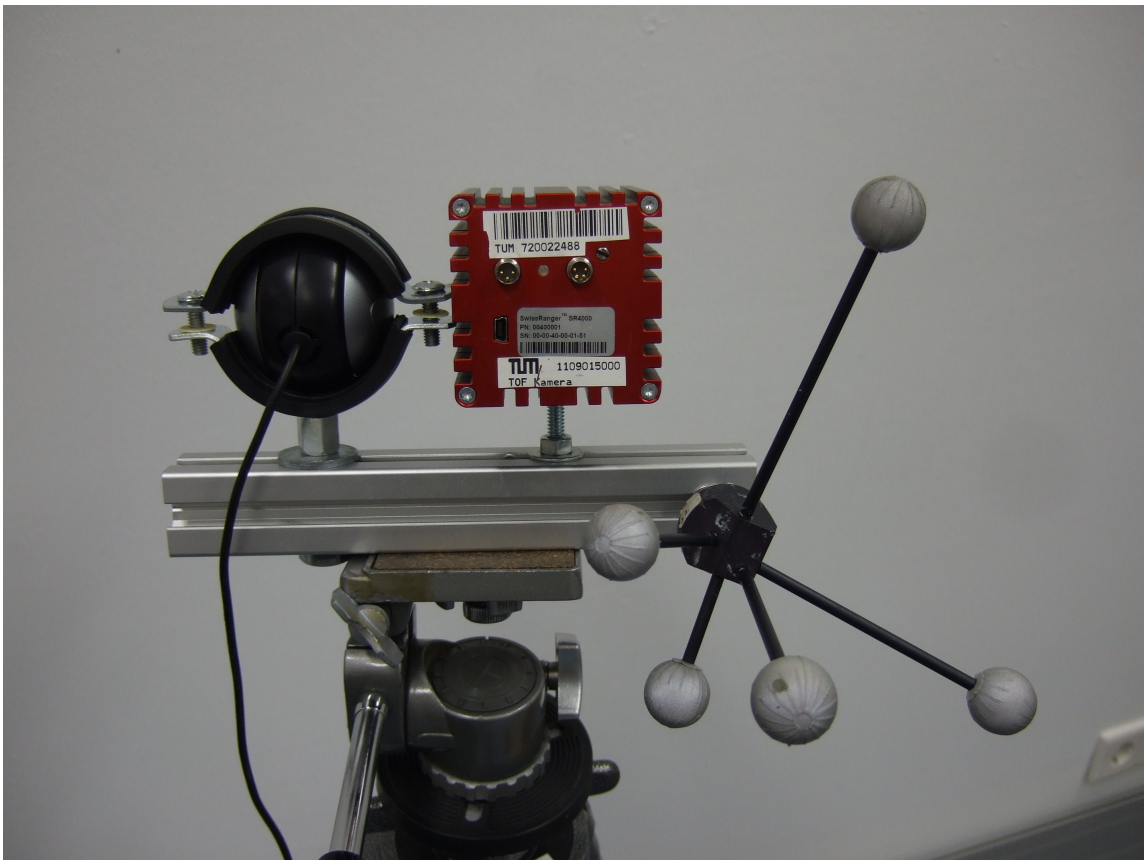


Figure 4.2: Mounted components: rear view

4.2 Coordinate systems

Every device defines at least one coordinate system. Actually the TOF camera brings in two coordinate systems: The camera coordinate system of the amplitude image denoted by AMP and the point cloud coordinate system denoted by TOF. The RGB camera defines the camera coordinate system CAM. The rigid body for evaluation defines $BODY$ and its pose is measured in the world coordinate system ART of the ART tracking system. Coordinate systems are denoted in italic letters in the following. It is necessary to register all coordinate systems to each other. Figure 4.3 shows the mentioned coordinate systems and the performed registrations (the arrows in figure 4.3). Every registration is explained in detail in section 4.4.

4.3 Intrinsic calibration

Intrinsic calibration for the RGB and the TOF camera was performed (see camera calibration by Zhang [31]). A chessboard was used as calibration pattern. The amplitude image of the TOF camera is very noisy, by this reason it was necessary, to use a chessboard pattern with just 5x4 corners.

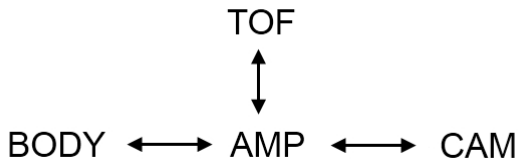


Figure 4.3: Coordinate systems and the calculated registrations

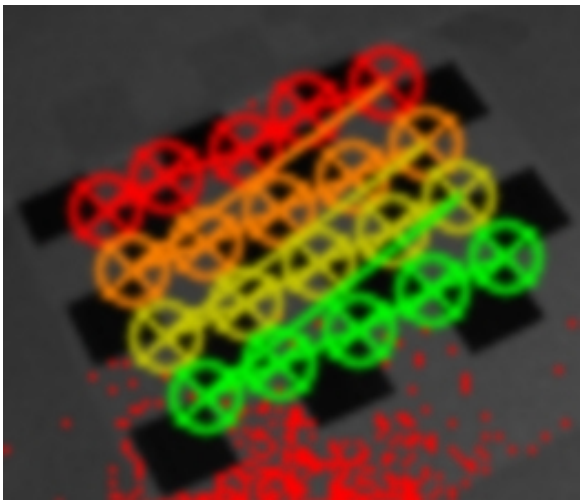


Figure 4.4: Corner detection of a chessboard pattern for calibration

Due to the bad signal to noise ratio of the amplitude images, the chessboard detection of OpenCV cannot measure the corners precisely (as can be seen in figure 4.4). Therefore at least 30 images where used for the intrinsic calibration. Nevertheless the results of the intrinsic calibration varied. The camera matrices and distortion coefficients of two consecutive calibrations using 30 amplitude images each, are shown below. It can be seen, that there is a clear drift of the principal point. Also the focal lengths and the measured distortions differed much.

$$M_1 = \begin{bmatrix} 261.450 & 0 & 85.167 \\ 0 & 260.111 & 69.406 \\ 0 & 0 & 1 \end{bmatrix}$$

$$radial_1 = [-0.67783, -0.52733]\,,\ tangential_1 = [0.00747, 0.02374]$$

$$M_2 = \begin{bmatrix} 255.833 & 0 & 88.975 \\ 0 & 256.598 & 64.561 \\ 0 & 0 & 1 \end{bmatrix}$$

$$radial_2 = [-0.89458, 0.95098]\,,\ tangential_2 = [0.00506, -0.00187]$$

The extrinsic calibrations which are described in the next sections are influenced directly by the intrinsic calibration.

4.4 Extrinsic calibration

This section gives a short introduction to coordinate system registration. Two methods have been implemented and compared. They are explained in detail in the following sections.

4.4.1 Registration of coordinate systems by corresponding points

One possibility to registrate two coordinate systems is to use corresponding points. There are several algorithms to perform the registration. Four major methods are presented in Eggert at al. [11].

At first, an algebraic least squares algorithm, that was introduced by Lowe [21], was adapted to 3D and implemented. By examination of the calculated transforms it turned out, that this algorithm does not produce orthonormal rotation matrices. Thus, the calculated rotation matrices had to be orthonormalized (see below).

The orthonormalization of the found rotation matrix is just the approximation with an orthonormal matrix and this method is algebraic and independent of the underlying geometry and thus distorts the results. A further method has been implemented that is considered as standard method for absolute orientation calculation. It was introduced by Horn [15] and uses quaternions.

Least Square method

Lowe described an algorithm for 2D coordinate system registration by corresponding points in [21], which was adapted to 3D for this work.
For one point correspondence the transformation is

$$p' = Rp + t = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.1}$$

After solving this equation for the transformation parameters it can be rewritten as

$$\begin{bmatrix} x_1 & y_1 & z_1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & z_1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 0 & 0 & 1 \\ & & & & & \cdots & & & & & & \\ & & & & & \cdots & & & & & & \\ x_n & y_n & z_n & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & z_n & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_n & y_n & z_n & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \\ m_9 \\ t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ z'_1 \\ \vdots \\ x'_n \\ y'_n \\ z'_n \end{bmatrix}. \tag{4.2}$$

As each point correspondence brings in three equations and the equation system consists of twelve equations, at least four point correspondences are needed to provide a solution. Equation 4.2 can be written as

$$Ax = b. \tag{4.3}$$

Using the pseudo inverse matrix $A^+ = \left(A^T A\right)^{-1} A^T$ of A, this equation can be solved by

$$x = \left(A^T A\right)^{-1} A^T b. \tag{4.4}$$

This solution minimizes the sum of squared distances from transformed model points to the image points. It works also for an overdetermined equation system (if more than four point correspondences are available).

The drawback of this method is, that the calculated rotation matrices are not orthonormal. This is due to the fact, that the algorithm chooses all values of the rotation matrix and the translation vector independently in such a way, that the least square error is minimal. The algorithm has no mechanism to ensure orthonormality of the rotation matrix.

To enforce orthonormality of the rotation matrix, a method described by Zahng [31] was implemented. This method works by minimizing the Frobenius norm of the difference matrix between the found and the approximated rotation matrix. It is clear, that this method produces a better rotation matrix in the sense of orthonormality, but this is done without regard to the geometric error, that the found transformation should minimize.

Registration using unit Quaternions

This section gives a short outline of the algorithm of Horn [15], which is considered as the standard method for point cloud registration. A complete description of the algorithm would go beyond the scope of this exposure. Unlike the least squares method described before, rotation and translation are calculated separately in this algorithm.

1. Rotation estimation:
   At first the centroids $\overline{c}$ and $\overline{c}'$ of the two point clouds are calculated by $\overline{c} = \frac{1}{n} \sum_{i=1}^{n} p_i$,

$\overline{c}'$ respectively. This centroids are subtracted from all points to get $\overline{p_i} = p_i - \overline{c} = (\overline{x}, \overline{y}, \overline{z})^T$ and $\overline{p_i}' = (\overline{x}', \overline{y}', \overline{z}')^T$ respectively.

For each pair of corresponding points the nine possible products of their coordinates $\overline{xx}'$, $\overline{xy}'$, ..., $\overline{zz}'$, are calculated and summed up to obtain $S_{xx} = \sum_{i=1}^{n} \overline{x_i x_i}'$, S_{xy}, ..., S_{zz} respectively. These nine values form the 4×4 matrix N:

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

The eigenvector e corresponding to the most positive eigenvalue of N contains the rotation quaternion q in the form

$$q = (e_1, e_2, e_3, e_0).$$

The corresponding rotation matrix can be calculated directly by the quaternion as described in [16].

2. Translation estimation:
   The translation vector is calculated as the difference between the centroid of the first point cloud and the rotated centroid of the second point cloud.

Horns method calculates a quaternion which is converted to a rotation matrix. Unlike the least squares method described in section 4.4.1, this method gives orthonormal rotation matrices.

The impact of noise and the number of points on the accuracy of this method have been evaluated. For this reason, two tests have been made. 3D points with random positions inside a cubical volume of $range^3$ m^3 have been created. This points P_i have been transformed by the transformation T_{real} to the points P_i'. Noise with a gaussian distribution has then been added to this points which gives points P_i'' with the Box-Muller algorithm [14]. With Horns method, the transformation $T_{estimated}$ is determined, which transforms points P_i to the points P_i''. The points P_i' are then transformed by $T_{estimated}^{-1}$ to yield points P_i'''. The mean length of the vectors $P_i - P_i'''$ is then calculated, which serves as accuracy measure for the transformation. The lower this mean length is, the higher is the accuracy of the transformation estimation.

The first test analyzes the impact of noise on the accuracy of the transformation. Points P_i are generated in a cubic of $2.0m^3$. 70 points where used for this test. The sigma of the gaussian noise was set from 0.0 m to 0.1 m. For every sigma value, the test was repeated 3000 times to get an averaging of the accuracy. Figure 4.5 shows the test result. The more noise is added to the points, the less is the accuracy. The accuracy is not falling very fast with increasing sigma, it is a linear relationship.

The second test shows the influence of the number of points to the accuracy. For this test, points where generated in a range of 10.0 m^2 and sigma was set to 0.08 m. The accuracy was measured for sets of four to 250 points. Every measure was again done 3000 times and then averaged. Figure 4.6 shows the results. The accuracy of the algorithm is very low with a small number of points but is increasing very fast.

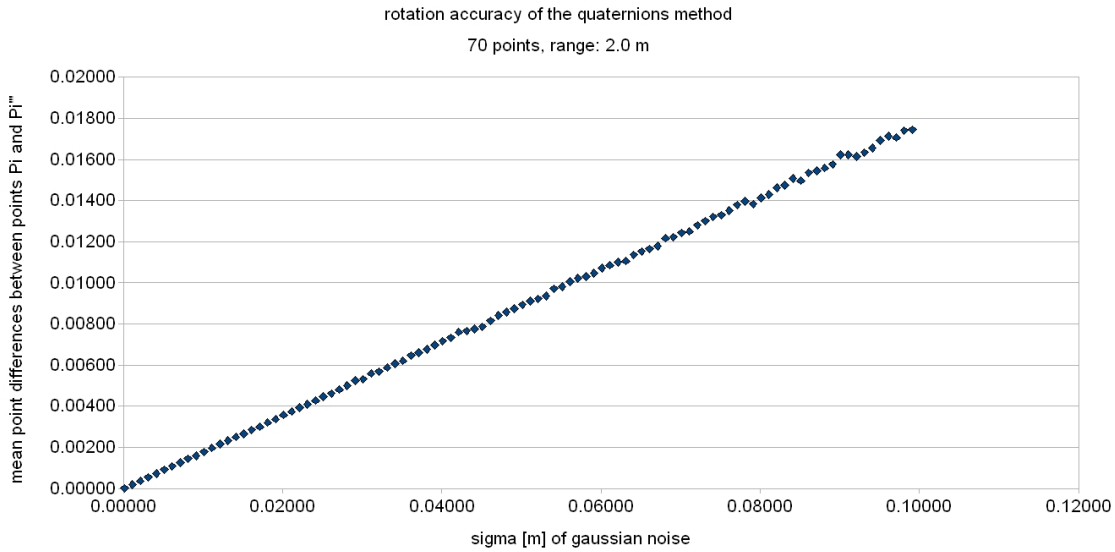


Figure 4.5: Test of the quaternion method : increasing noise

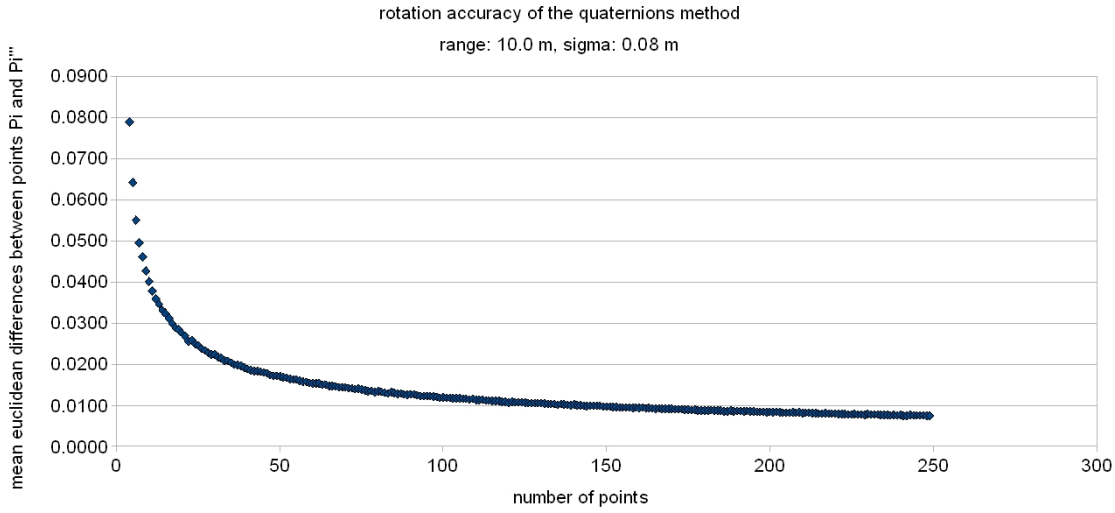


Figure 4.6: Test of the quaternion method: increasing number of points

4.4.2 Estimation of ${}^{CAM}T_{AMP}$

In this calibration step, the transformation of the RGB camera coordinate system to the camera coordinate system of the TOF camera is determined. Both are monochrome cameras.

After the intrinsic calibration, the camera matrix and distortion coefficients are known. The camera pose in the calibration target coordinate system can then be estimated by minimizing the reprojection error of the calibration points. OpenCv provides this functionality by the function cvFindExtrinsicCameraParams2. This function calculates the pose of the

TOF amplitude image coordinate system in the chessboard pattern coordinate system

$$^{CHESS}T_{AMP},$$

and the pose of the RGB camera in the same chessboard pattern coordinate system

$$^{CHESS}T_{CAM}.$$

The pose of the RGB camera in the TOF camera coordinate system can then be calculated as

$$^{CAM}T_{AMP} = (^{CHESS}T_{CAM})^{-1} \cdot {}^{CHESS}T_{AMP}. \tag{4.5}$$

A transformation consisting of rotation R and translation t in the form

$$p' = Rp + t$$

is inverted as follows:

$$R^{-1} = R^T, \;\; t^{-1} = -R^T t. \tag{4.6}$$

4.4.3 Estimation of $^{TOF}T_{AMP}$

As described in section 3.4.2, the camera driver calculates the 3D Cartesian Coordinates from the depth map. This is provided by the function SR_CoordTrfFlt of the software driver. The returned points are expressed in meters and define the coordinate system TOF. The Z-axis of this coordinate system increases along the optical axis, Y is increasing vertically upwards and X is increasing to the left.
It is important to know the transformation between TOF and AMP. For example, if the projected image point of an arbitrary object point in 3D measurement space is desired, this coordinate system transformation is required. This is the case for every algorithm, that works on the 3D points and the amplitude image simultaneously.
Section 4.4.1 describes, how coordinate systems can be registrated by using corresponding points. For the registration of TOF to AMP it is possible to get corresponding point coordinates. In section 4.4.2 it is shown how to determine the cameras pose relative to a calibration pattern. The 3D object points of the calibration rig in the cameras coordinate system are calculated by the use of this pose. Because the calibration target is a flat chessboard pattern, several measurements with different camera poses are taken (see figure 4.7). So the points are non planar distributed over the 3D space. They are the measurement points in the camera coordinate system AMP.
To obtain the same measurement points in TOF the 3D TOF points are taken on the positions of the detected chessboard corners in the amplitude image.

These two point sets are then registered by one of the two algorithms described in section 4.4.1.

Another method to obtain the 3D points from the TOF camera is by calculating them from the depth map as described in section 4.5. This method is preferred, because the lens distortion can then be taken into account and a registration of the coordinate systems TOF and AMP is then not longer necessary, as this coordinate systems are the same.

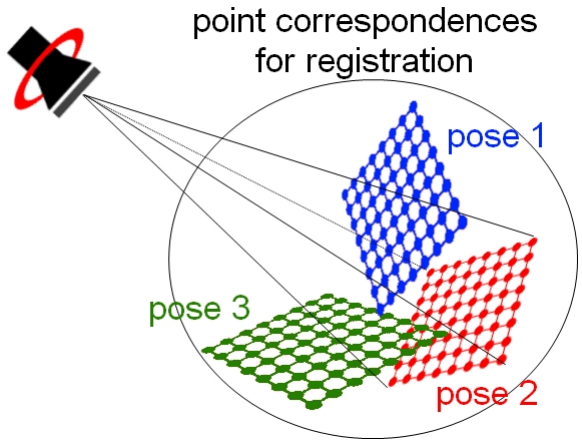


Figure 4.7: Corresponding points for the estimation of ${}^{TOF}T_{AMP}$

4.4.4 Estimation of ${}^{AMP}T_{BODY}$

The reference system for evaluation is a tracking system of the company ART. To compare the determined pose difference with the pose difference estimated by ART, it is important to know the transformation between the TOF camera coordinate system AMP an the coordinate system $BODY$ of the rigid body.

To obtain this transformation the following steps are proceeded:

1. Perform an intrinsic camera calibration of the TOF camera.
2. Place the TOF camera before a chessboard in the ART measurement field, so that the chessboard is fills the amplitude image completely.
3. Register the chessboard pattern in ART world coordinates to obtain ${}^{ART}T_{CHESS}$.
4. Measure the pose of the rigid body in ART world coordinates to obtain ${}^{ART}T_{BODY}$.
5. Estimate the cameras pose to the chessboard pattern to obtain ${}^{CHESS}T_{AMP}$.
6. Calculate ${}^{AMP}T_{BODY}$ by

$$ {}^{AMP}T_{BODY} = ({}^{CHESS}T_{AMP})^{-1} \cdot ({}^{ART}T_{CHESS})^{-1} \cdot {}^{ART}T_{BODY} \tag{4.7} $$

4.4.5 Estimation of ${}^{TOF}T_{CAM}$ and ${}^{TOF}T_{BODY}$

The transformation ${}^{TOF}T_{CAM}$ is calculated by

$$ {}^{TOF}T_{CAM} = {}^{TOF}T_{CAM} \cdot ({}^{CAM}T_{AMP})^{-1}. \tag{4.8} $$

${}^{TOF}T_{BODY}$ is calculated by

$$ {}^{TOF}T_{BODY} = {}^{TOF}T_{AMP} \cdot {}^{AMP}T_{BODY}. \tag{4.9} $$

These both transformations are required later in this work.

4.5 3D point determination using camera parameters and distance measure

The calibration of a TOF camera as described in section 4.3 gives the camera matrix

$$P = \begin{bmatrix} k_u f & 0 & p_u \\ 0 & k_v f & p_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

and the distortion coefficients for radial and tangential distortion r_1, r_2, t_1, t_2. The projection equation which transforms an object point $Q = (X, Y, Z)^T$ in camera coordinate system into an image point $q = (u, v)^T$ is

$$u = \frac{k_u f \cdot X}{Z} + p_u \tag{4.11}$$

where X and Z are object coordinates, $k_u f$ denotes the focal length expressed in pixel-related units along the u-axis and p_u denotes the u-coordinate of the principal point. The calculation of the pixel coordinate v is analogous with

$$v = \frac{k_v f \cdot Y}{Z} + p_v. \tag{4.12}$$

D is the radial distance between image point q and Object point Q (see figure 4.8) given by equation 3.1 or 3.2. It can be expressed by

$$D = \sqrt{X^2 + Y^2 + Z^2}. \tag{4.13}$$

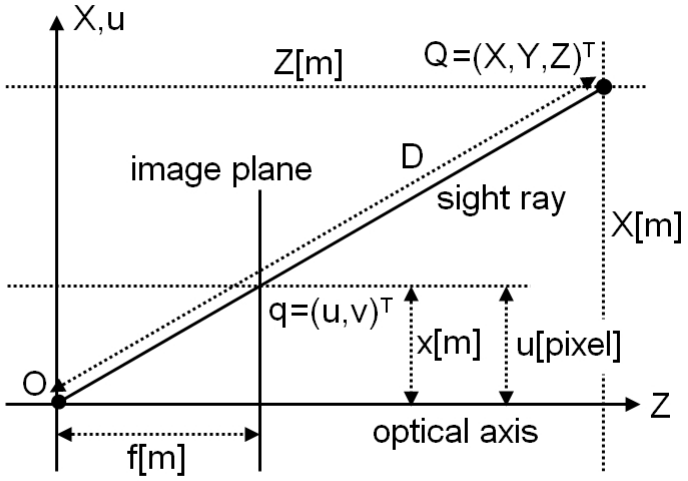


Figure 4.8: Object to image point projection

In disregard of the lens distortion, equations 4.11 and 4.12 can be rewritten as

$$X = aZ,\ Y = bZ \tag{4.14}$$

with

$$a = \frac{u - p_u}{kuf},\ b = \frac{v - p_v}{kvf}.$$

To yield Z equations 4.14 are substituted into equation 4.13. Solved for Z results in

$$Z = \sqrt{\frac{D^2}{a^2 + b^2 + 1}}. \tag{4.15}$$

The Z value can than be used to calculate X and Y by equation 4.14. In regard of the lens distortion, the depth map should be undistorted before calculating the object point coordinates (see [22], [12], [8]).

5 Plane recognition

Finding the pose difference of two point clouds makes it necessary to detect features in both point clouds, that can be matched. Features are a description of shapes or portions in a point cloud and can present different kind of information. For example there are features which yield 3D positions in space while other features yield orientations. Features can be geometric primitives as well as subsets of the points which have special characteristics.

Common 3D indoor scenes usually contain geometric standard primitives like planes or 3D edges.

One approach to calculate the pose update between two point clouds is to find significant geometric primitives in both point clouds, find the correct mapping of this features in both coordinate systems and to calculate the coordinate transformation.

5.1 Ability of different geometric primitives as features

This section states the advantages and disadvantages of different geometric primitives for pose calculation. It is done in regard to the properties of time-of-flight measurements.

5.1.1 3D edges

Edges occur at the intersection of surfaces of different orientations. Therefore one possibility to yield 3D edges is to segment surfaces first, to get their boundaries and thus the 3D edges. Another alternative is to apply a gradient filter on the depth map of the 3D camera. Figure 5.1 illustrates the problem, that occurs in both cases: Indeed the 3D measurement points lie on the surfaces itself, but not on their boundaries (blue lines). Thus, the reconstructed edges (green line) would not hit the real edges exactly. The first described possibility to detect edges would allow a more precise detection of edges between two surfaces which are seen by the camera because the surface intersection can be calculated with sub-pixel precision. The disadvantage is that the segmentation and the analytic representation of this surfaces must be calculated. If just one adjoining surface is visible to the camera (like in figure 5.1), this method would not yield an exact edge. The second described possibility has the disadvantage, that the results of a gradient filter are biased if on one side of the edge are higher gray values in the depth map than on the other side.

5.1.2 Planes

Indoor scenes contain typically many planes. This can be walls, the floor, the ceiling, table plates or other objects.

Planar surfaces are 2D structures and therefore include much more points than 1D structures like edges. As more points are used as more precise the model parameters can be calculated due to the averaging effect. On the other hand, planes are 2D structures with just a

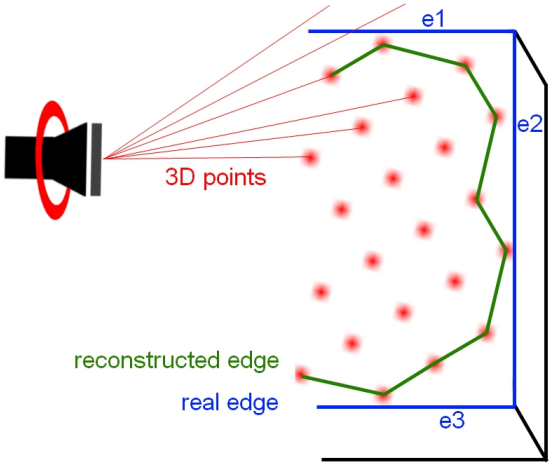


Figure 5.1: 3D edges as geometric features: sample artifacts

few parameters. By applying sophisticated algorithms, they can be segmented easily and fast in 3D point clouds. If plane points are segmented successfully, there exist standard algorithms to estimate the parameters of the plane.

There is one big advantage of planes in respect to this work: Every plane has a normal vector. It can be used to calculate parts of the rotation of the pose difference.

Using planes as features for pose estimation brings also two disadvantages. With respect to metric transformations planes possess three invariances: Two translation invariances along the plane axes (the green arrows in figure 5.2) and one rotation invariance for rotations around the normal vector (the blue arrow in figure 5.2). Thus, these parts of a pose change cannot be detected by using one single plane as feature.

It is not enough to use just one plane for pose estimation. At least three planes with different normal vectors are required for definite pose estimation. The planes must than be matched between one frame and the next frame.

Time of flight cameras have a small viewing angle and thus mostly there are just few planes in an image. Therefore, other features must be used additionally.

5.1.3 More complex primitives

More complex geometric primitives like spheres or conics could also be used as features. But the higher the complexity of an model, the more parameters must be estimated.

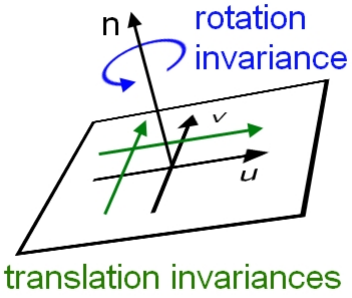


Figure 5.2: Transform invariances of planes

5.2 Mathematical fundamentals of planes

5.2.1 Representation of planes

Several representations of planes exist. The common ones are described in [27]. The representation of a plane by its normal vector and one point of it was chosen. The mathematical formulation is

$$\vec{n} \cdot (\vec{r} - \vec{r}_0) = 0, \tag{5.1}$$

where $\vec{n}$ denotes the normal vector, $\vec{r}$ a running point and $\vec{r}_0$ a fixed point on the plane. This representation has the advantage, that it is possible to achieve the distance of a point $\vec{r}_Q$ from the plane very fast by calculating a scalar product. With equation 5.2 this distance is calculated.

$$d = \frac{\vec{n} \cdot (\vec{r}_Q - \vec{r}_0)}{\|\vec{n}\|} \tag{5.2}$$

In the processing steps described later (for example the region growing), this point-plane distance has to be calculated very often. This plane representation has also the advantage, that only few values are needed to define the plane: Three point coordinates for $\vec{r}_0$ and three vector elements for $\vec{n}$. In this work, $\vec{r}_0$ is not arbitrary chosen but the intersection point between the plane and the line which is perpendicular to the plane and goes through the origin. This point is denoted as foot point of the plane in the following. So the vector length of $\vec{r}_0$ is the distance of the TOF camera to the plane. Also the normal vector is scaled to an unit vector to fasten later processing. Only four values are needed then to define the plane: Three vector elements for the normal vector $\vec{n}$, and one value for the distance α between camera and plane. The plane representation is then

$$\vec{n} \cdot (\vec{r} - \alpha\vec{n}) = 0, \tag{5.3}$$

with $\alpha = \frac{1}{\|n\|}\vec{n} \cdot \vec{r}_a$, where $\vec{r}_a$ is an arbitrary point on the plane. Thus the point-plane distance calculation is simplified to

$$d = \vec{n} \cdot (\vec{r}_Q - \alpha\vec{n}). \tag{5.4}$$

5.2.2 Intersection of a plane with a line

In this work, the intersection points of planes with sight rays are required in some cases. Sight rays are straight lines which pass through its corresponding image point on the cameras CCD and the optical center of the camera. If the intrinsic parameters of the camera are known by calibration, this line equations can be formed. A line is given by the equation

$$\vec{r} = \vec{r}_1 + \lambda\vec{v} \tag{5.5}$$

and the plane is given by Equation 5.3. To determine the intersection point, λ is calculated by insertion of equation 5.5 into equation 5.3. Using this λ in equation 5.5 yields the intersection point $\vec{r}_s$:

$$\vec{r}_s = \vec{r}_1 - \left(\frac{\vec{n}\cdot(\vec{r}_1 - \alpha\vec{n})}{\vec{n}\cdot\vec{v}}\right)\vec{v}. \tag{5.6}$$

5.3 Estimation of plane parameters from 3D points

Two methods to estimate the parameters of a plane from its generating points are presented and compared in this section.

5.3.1 Multiple regression

The multiple regression model (see [28]) assumes, that the z value depends on x and y and a noise d is added to the correct z value for example by measurement error. Every measurement point that forms the plane can then be written as

$$z_i = c_0 + c_1 x_i + c_2 y_i + d_i. \tag{5.7}$$

All points together form the equation system

$$z = X\gamma + \delta \tag{5.8}$$

with

$$X = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{pmatrix}$$

as design matrix,

$$z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}$$

as vector of the function values,

$$\gamma = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix}$$

as vector of the plane parameters and

$$\delta = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

as noise vector.

As shown in [28], the plane parameters c_0, c_1 and c_2 can be estimated with the pseudo inverse of X:

$$\gamma = (X^T X)^{-1} X^T z. \tag{5.9}$$

This is a least square estimation of a plane, which minimizes the distances to the points along the z direction.

5.3.2 Principal component analysis

The principal component analysis is a powerful tool to find directional distinctness of high dimensional point clouds. It is often used to reduce the data to projections on the point clouds main directions. Physicians use it to calculate the main rotation axes of objects, which are available in form of a point cloud. These main rotation axes or rather the main directions of the point cloud are called the principal components.

In detail, for a n-dimensional point cloud, the PCA delivers n n-dimensional vectors. The first vector is the direction, along which the point cloud possesses the highest variance. The second vector is the direction along which the variance of the point cloud is maximum under the restriction, that it is orthogonal to the first vector. Thus the PCA estimates a set of pairwise orthogonal vectors, which point to the directions along which the point cloud has the most spread.

For Every direction it delivers also the variance of the point cloud along it.

PCA is often used to classify objects or rather to give classes, to which an object can not belong. For example, the point cloud of a cube has three rotation axes along the diagonal of the cube. The variances along this directions are all the same, because of the symmetry of the cube. The same relation between the variances applies for the point cloud of a ball. Thus with a PCA it is possible to find out, if points do not form a certain object, but a unique identification of the objects, which the points are forming is not possible. It can be used to reduce the set of possible classes for object identification.

Another important fact is, that if an object is identified, its orientation is also known by the principal components.

The principal component analysis consists of tree steps (see [18]):

1. Average subtraction
   The average of all points is calculated

   $$\hat{P} = \frac{1}{n} \sum_{i=0}^{n-1} P_i. \tag{5.10}$$

   This average is subtracted from every point

$$P_i' = P_i - \hat{P}. \tag{5.11}$$

2. Calculation of the covariance matrix
   The next step is to gain information about the point clouds main directions. For this reason it is important to gather the correlation between all coordinates of the measurement. This is done by calculating the covariance matrix

$$C = \begin{pmatrix} cov(x,x) & cov(y,x) & cov(z,x) \\ cov(x,y) & cov(y,y) & cov(z,y) \\ cov(x,z) & cov(y,z) & cov(z,z) \end{pmatrix} \tag{5.12}$$

   with

$$cov(a,b) = \frac{\sum_{i=0}^{n-1}(a_i - \hat{a})(b_i - \hat{b})}{n-1}.$$

3. Calculation of the eigenvectors and eigenvalues of C
   The principal components are then the eigenvectors of C (see [18]). The eigenvector according to the largest eigenvalue is the direction along the variance of the point cloud is maximum. The eigenvalues are a measure of the variance in the direction of the related eigenvector.

To estimate plane parameters, it is necessary to know how the principal components of a plane are aligned. A plane in 3D is a structure with a big spread in two orthogonal directions and flat distribution in the third orthogonal direction. Thus a point cloud of a plane has two large eigenvalues and one small. The eigenvector corresponding to the small eigenvalue is orthogonal to the main directions of the plane and it is therefore the normal vector of the plane.

To estimate plane parameters with a PCA, step one reveals a point on the plane: $\hat{P}$. It is not affected by measurement noise as it is neutralized by averaging (equation 5.10). Step three gives the normal direction of the plane. It is the eigenvector corresponding to the smallest eigenvalue.

5.3.3 Comparison of multiple regression and PCA

Figure 5.3 visualizes the difference between PCA and multiple regression for finding plane parameters. Multiple regression minimizes the distances along the z direction of the measurement points to the plane. PCA minimizes the orthogonal distances of the measurement points to the plane. As multiple regression is a least square approximation, this method should not be used for planes that are nearly parallel to the axis, for which the regression is minimizing the distances (in the above representation of the plane it is the z-axis). In this case, noise causes a much higher variance of the distances between the generating points and the plane along this direction than in the direction which is orthogonal to the plane. Thus noise has a much stronger influence on planes in this direction than on planes orthogonal to it. Both methods have been implemented but only PCA is used because of the influence of the plane direction on the multiple regression method.

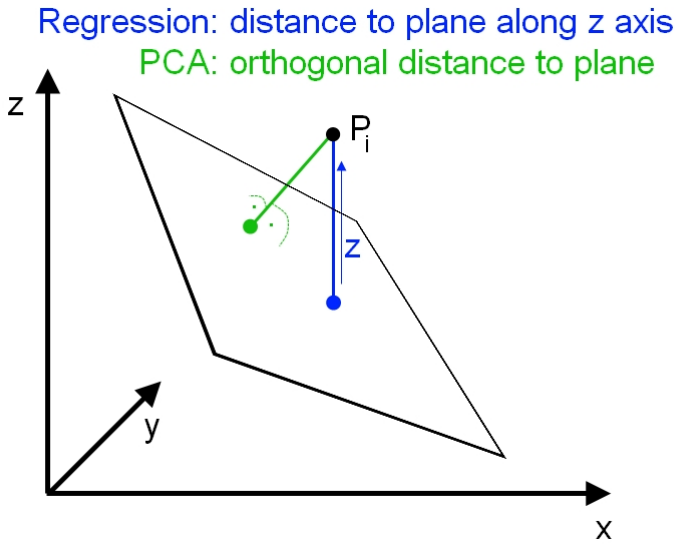


Figure 5.3: Distances that are minimized by PCA and multiple regression

5.4 Segmentation of planes in 3D images

Before plane parameters can be calculated, the planes have to be segmented. This means, to find out, which pixel of the depth map belongs to a plane. This is a somewhat difficult task, as it would be too time consuming to test all combinations of points, whether they are forming a plane by brute force. Nevertheless, planes can be segmented from depth images very fast by a more sophisticated technique. For this goal it is important to know, that an continuous surface of an object appears also as continuous region in the depth map, if the object is not occluded by another object.

Two methods have been developed and compared in this work. One is a clustering method, the other uses the above mentioned property of the regions of the depth map, that belong to a plane.

5.4.1 Clustering method

The first idea to find planes in depth images fast was a clustering technique. The idea behind this is, that a plane is defined by its normal vector and its distance to the origin. It can thus be defined by its foot point (see section 5.2.1). This works for every plane that does not pass through the origin, which is always the case in this work, because a plane passing through the origin would be parallel to the sight rays an thus invisible to the camera.

The depth image was divided into small sub regions (in the example which is shown by figure 5.4 sub regions of the size 10x10 pixel where used). Every of this regions defines a separate point cloud, for which plane parameters have been computed by the regression method described in section 5.3.1 and the perpendicular foot point was printed into figure 5.4. The scene consisted of two planes only (the floor and a shelf). This two planes produce two clearly distinguishable clusters in the cluster space.

To detect planes, clusters in the 3D image must be found. As can be seen in figure 5.5,

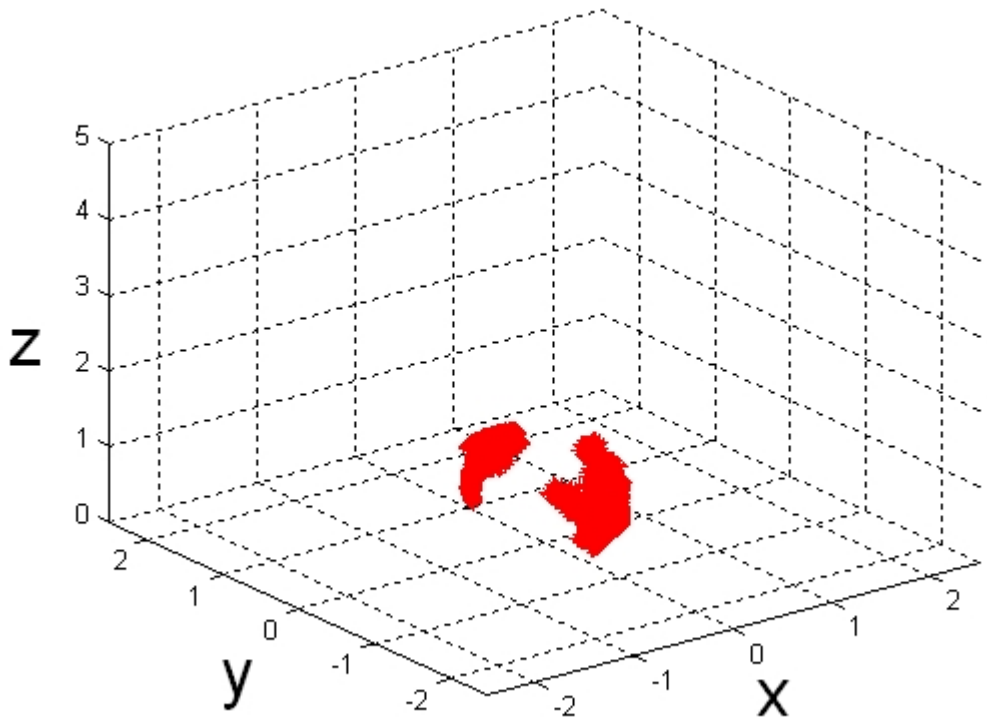


Figure 5.4: Clustering of the foot points of two planes

this clusters have a special form. They are not punctual but have some spread. This special form probably occurs, because of effects caused by multiple reflections, that are described in section 3.3.5.

The method using foot point clusters entails the problem of finding this clusters. This requires time consuming algorithms. So this method was not implemented further. Instead, the region growing method described in the next section was developed.

5.4.2 Finding small plane patches and link them by region growing

Dividing the point cloud into sub sets and determining the plane characteristics of each:

The method to determine plane parameters, which was used in section 5.4.1 has the disadvantage that it does not give a measure about the fact, how good the points really form a plane. For this purpose, the distances of the single points to the estimated plane must be calculated afterwards. The algorithm described in section 5.3.2 gives the plane equation and additionally gives a measure for the spread of the points along the normal direction of the plane by the ratios of the eigenvalues $\lambda_0, \lambda_1, \lambda_2$.

As plane confidence value the ratio between the third and the second largest eigenvalue

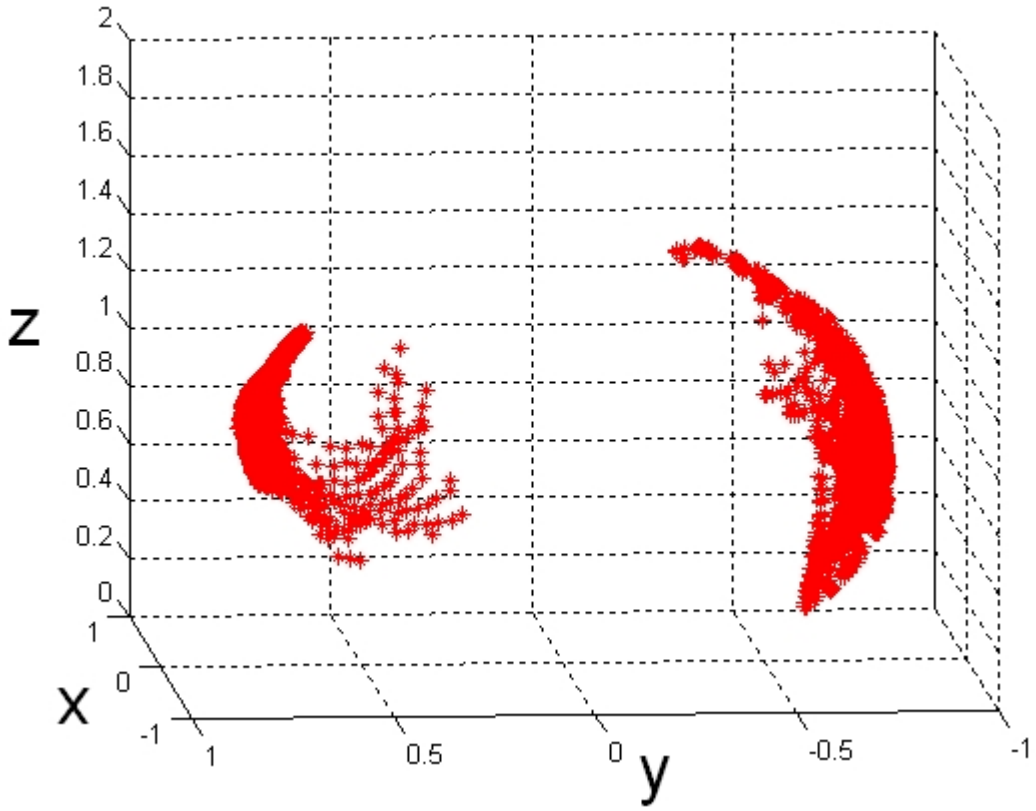


Figure 5.5: Spread of the clusters of foot points

was used:

$$c_{2,1} = \frac{e_2}{e_1}. \tag{5.13}$$

If $c_{2,1} <= c_{threshold}$ this point cloud is considered as plane and regarded in further processing steps. Some attempts have been made to use also the first eigenvalue for defining the plane confidence, but it turned out that results can be achieved by $c_{2,1}$ only. This is due to the fact, that a plane has much less spread along its normal vector than along any other perpendicular direction.

The procedure to segment planes is, to divide the depth map into adjoining or overlapping sub regions like in section 5.4.1. For every such patch of the depth map the principal components of the related point cloud are computed. Then the plane confidence $c_{2,1}$ is calculated and if the patch can be considered as plane, a data structure is generated, containing a reference to the patch and the plane parameters. Figure 5.6 illustrates how the depth map is divided into sub regions. Every region is quadratic and has an odd size. Between the sub regions is a shift. The best results for plane segmentation have been achieved with a patch size of 22x22 points and a shift of 4 pixel. With larger patches, small plane regions are not detected any more while smaller patches cause a loss in accuracy, because less points are used to calculate the plane parameters.

Figure 5.7 shows an amplitude image and the corresponding depth map with the found

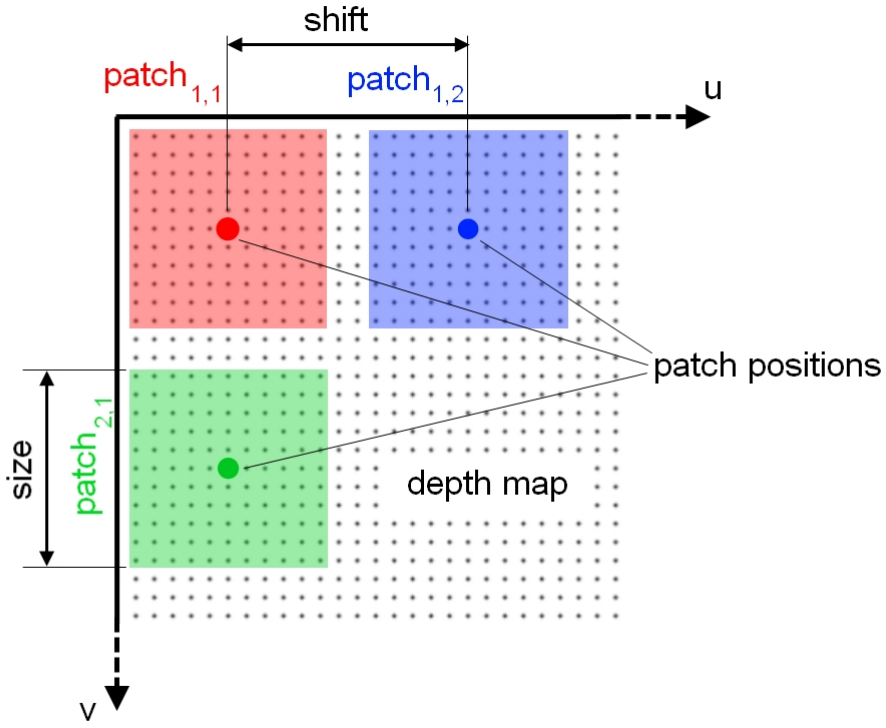


Figure 5.6: Attributes of plane patches

plane patches (green marked). The scene consisted of a cubicle on the left side and a column with low curvature beside it on the right side.

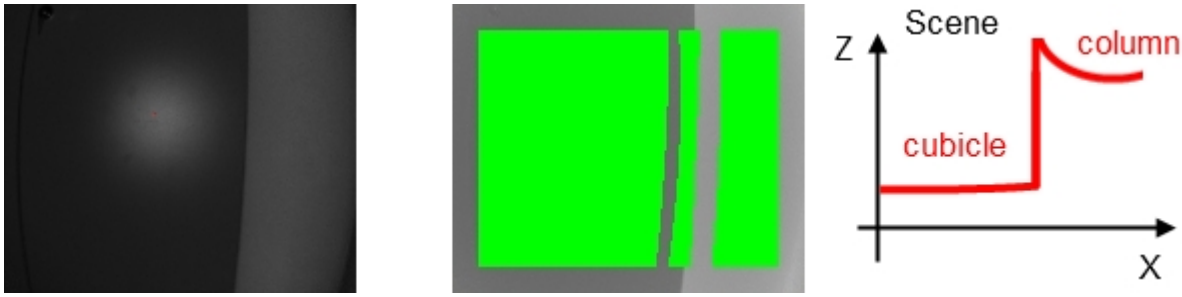


Figure 5.7: Detection of planes on depth edges

For every plane patch a green dot was printed into the image. As it can be seen, plane patches where found all over the cubicle and also all over the column. The column had a curved surface, but as the curvature was very low it is an approximation of many small planes.

As can be seen in this figure, plane patches have also been detected at the border of the

cubicle. The reason for this is illustrated in figure 5.8. If there is a big difference in depth between two adjoining objects and the 3D points of a plane patch distribute among this two objects, there is a big spread of the point cloud in two orthogonal directions. In figure 5.8 this two directions are visualized by the red arrows. Because of the fact that there is nearly no spread in the third orthogonal direction, the principal components of the point cloud fulfill the criteria of a plane (equation 5.13).

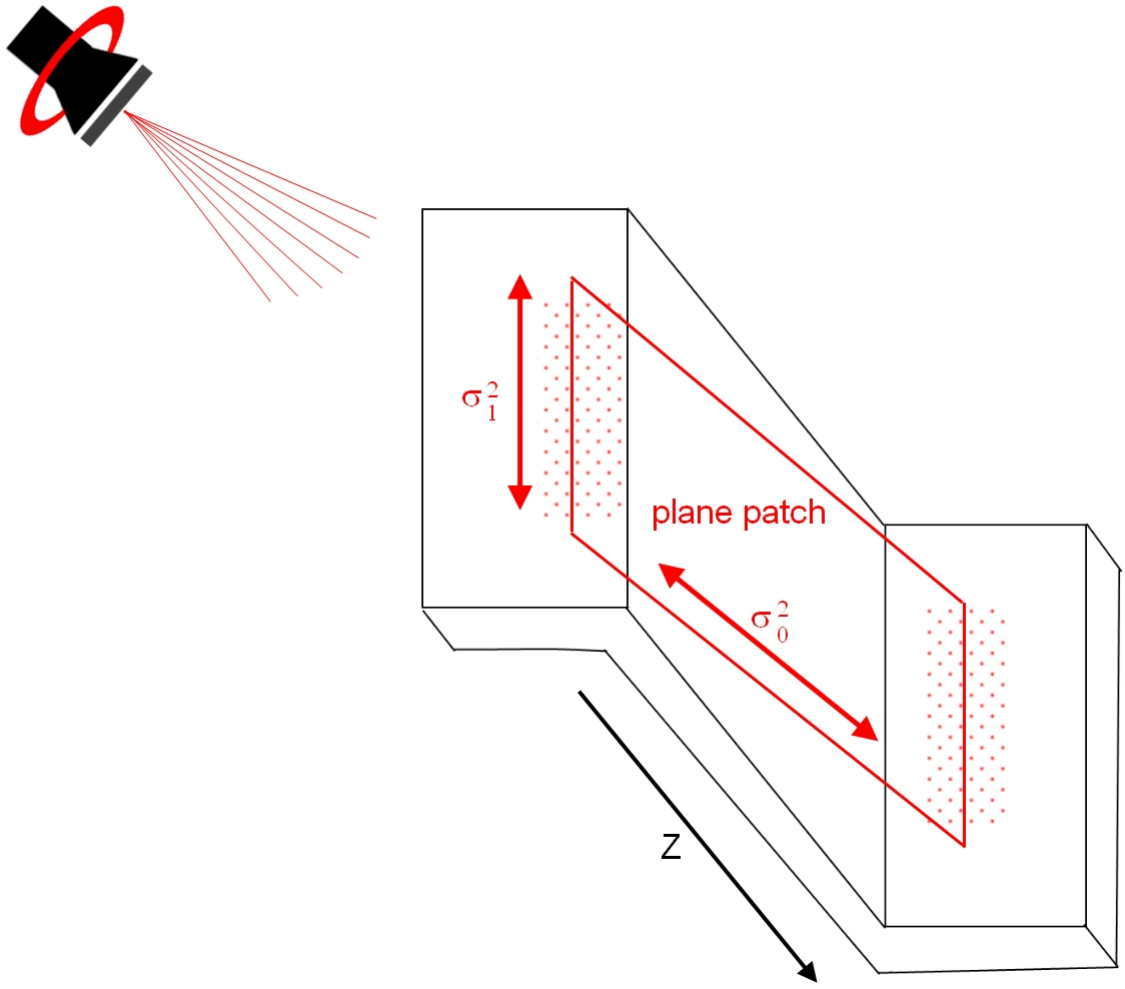


Figure 5.8: Point distributions on depth edges

To solve this problem, the intersection points of the sight rays and the estimated plane are calculated by equation 5.6. The average distance between the 3D measurement points and the calculated intersection points is then determined. If the objects surface is a plane and continuously in depth, the intersection points are near by the 3D points, as can be seen in the left part of figure 5.9. In the case of figure 5.8, the recognized plane patch does not match the objects surface and thus the 3D measurement points are far away from the intersection points (see figure 5.9, right).

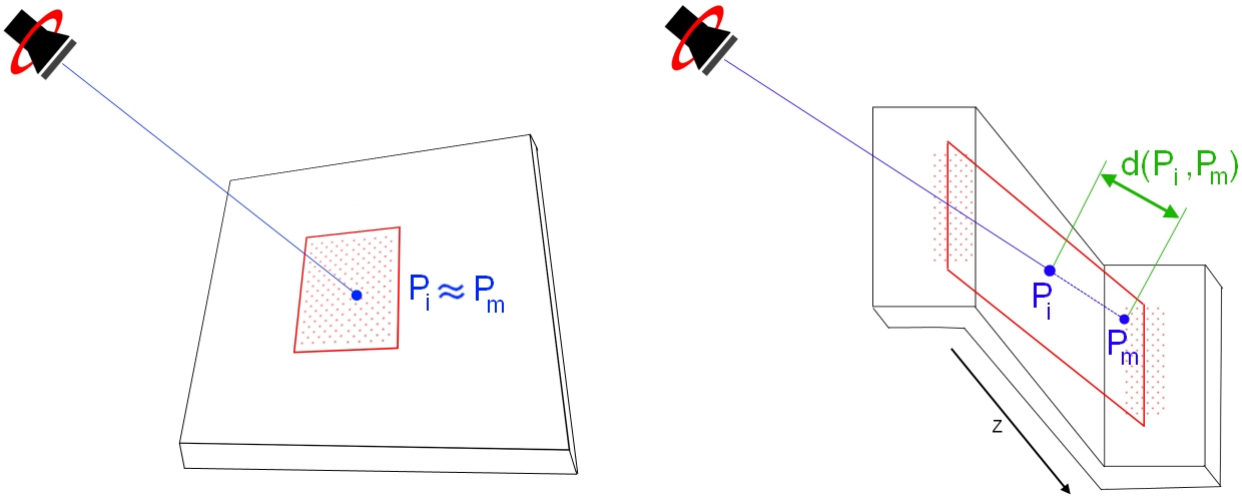


Figure 5.9: Intersection point of sight rays with detected planes

The following condition must be satisfied to assume the plane patch as an approximation of the objects surface:

$$\frac{1}{n}\sum_{j=0}^{n-1}(\|P_{intersection,j} - P_{measurement,j}\|) <= t. \tag{5.14}$$

The described detection of planes that do not exist could be eliminated successfully by applying this threshold. As can be seen in Figure 6.2, only plane patches are detected on surfaces, which really are an approximation of a plane. These patches are marked as PLANE.

Linking adjoining planar sub sets by region growing:

When all plane patches have been checked for their plane characteristics, the plane patch with the smallest value $c_{2,1}$ (see equation 5.13) is used as seed point for region growing. Region growing [1] is an image segmentation method that adds neighboring pixels of a region to it, if they are fullfilling a certain condition.

Figure 5.11 visualizes the proceeding of region growing.

1. Create an empty FIFO structure, mark all patches as NOT TOUCHED and set the plane identification number to 1.

2. Choose the seed patch and add it to the FIFO
   The seed patch is the plane patch with the smallest $c_{2,1}$ (see equation 5.13) among all patches that are marked as PLANE. Mark the seed patch as TOUCHED.

3. Initialize the plane equation of the growing plane G.
   The plane equation of the growing plane G is the plane equation of the seed patch.

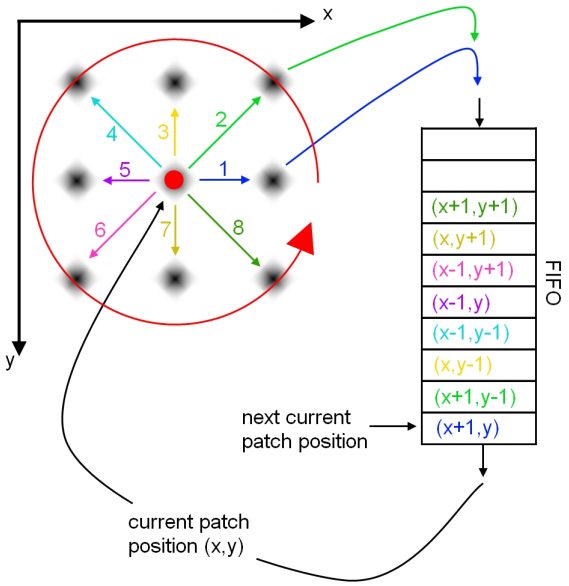


Figure 5.10: Region growing: FIFO principle

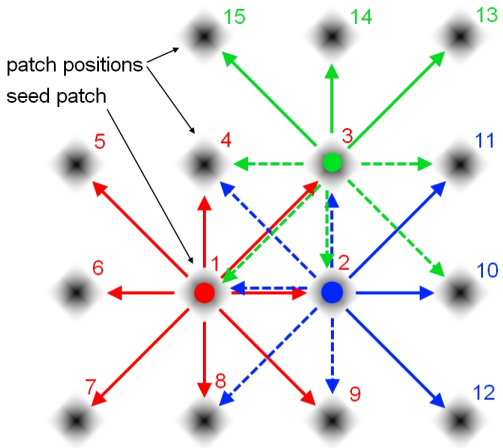


Figure 5.11: Region growing: processing order of neighboring plane patches

4. Take a patch from the FIFO
   This patch is now the current patch C.

5. Check neighbors of C
   Check adjoining patches of the 8-neighborhood of C, if they are marked as PLANE and NOT TOUCHED and fit to G. For this purpose check the neighbors in the order shown in figure 5.10. If a neighboring patch fits to the growing plane G, mark it as TOUCHED, add it to the FIFO and assign the current plane identification number to it.

A plane patch P fits to the growing plane G, if the following two conditions are fulfilled:

- The angle between the normals of G and P is below a threshold.

$$\alpha_{P,G} = acos\left(\frac{\vec{n}_G \cdot \vec{n}_P}{\|\vec{n}_G\| \cdot \|\vec{n}_P\|}\right) \leq \alpha_{threshold} \tag{5.15}$$

- The distance $d_{P,G}$ of the gravity center $\vec{m}_P$ of the plane patch to G is below a threshold.

$$d_{P,G} = \vec{n}_G \cdot (\vec{m}_P - \alpha\vec{n}_G) \leq d_{threshold} \tag{5.16}$$

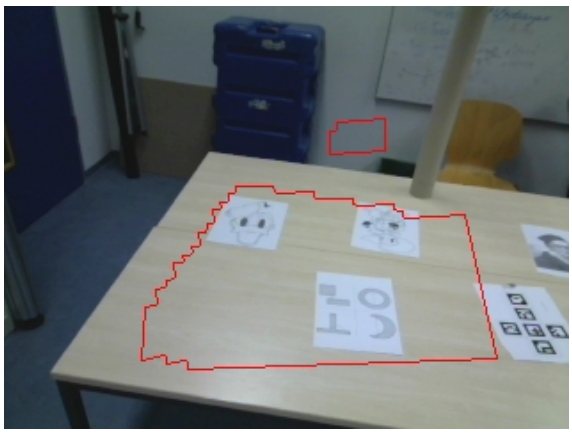
  where α denotes the distance between origin an plane G (see equation 5.4) and $\vec{m}_P$ is calculated by the n points p_i that form the patch

$$\vec{m}_P = \frac{1}{n}\sum_{i=1}^{n} p_i$$

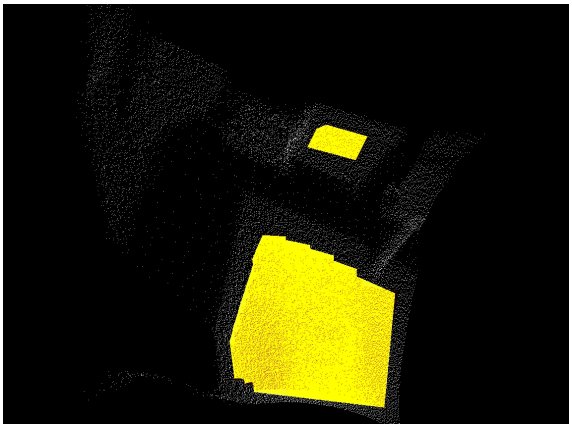
6. Update plane equation of G
   The new parameters of the growing plane G are calculated by multiple regression (described in section 5.3.1) or by PCA (described in section 5.3.2). As described in section 5.3.3,PCA is suggested. All points that are assigned to G in the current step of the region growing are considered in the calculation of the plane parameters.

7. If the FIFO is not empty, continue with step 4. If the FIFO is empty, add a new plane to plane list and all patches belonging to it. Increment the plane identification number. Search new seed point and start with step 2 until no patches are available any more, that are marked as NOT TOUCHED and as PLANE.

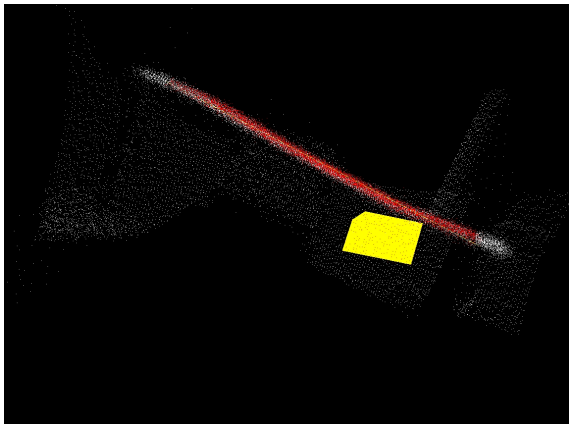
Figure 5.12 shows a 3D visualization of a scene and the recognized planes. The RGB image of the scene can be seen in 5.12(a). The non-plane points of the scene are plotted white, while points belonging to planes are red. The estimated plane is plotted yellow. As can be seen from view 2 and view 3, the calculated planes are a good approximation of its generating points.



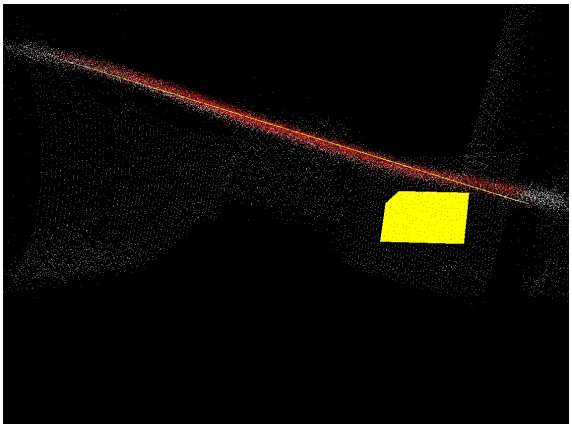
(a) Scene



(b) View 1



(c) View 2



(d) View 3

Figure 5.12: 3D visualization of detected planes

6 Visual feature extraction on the textures of the planes

In section 5 it was demonstrated, how planes can be segmented out of a depth map. Three conditions must be fulfilled, if the pose of one frame to the previous frame should be calculated by using just planes:

1. It must be known, which plane of frame i corresponds to which plane of frame i-1
2. At least three pairs of corresponding planes must be found
3. All of this three planes must have a different orientation to each other

If condition two is not fulfilled, there remain invariances in the coordinate transform as can be seen in figure 5.2 and figure 6.1. If just one corresponding pair of planes has been found in two successive frames, there remain two invariances in translation and one in rotation around the plane normal to find a coordinate transform which brings this two planes together. If two plane pairs with different orientations have been found, there still remains a translation invariance along the direction of the intersection line. Just with three or more plane pairs, a unique transform can be found, that transforms the planes of one frame into the planes of the next frame.

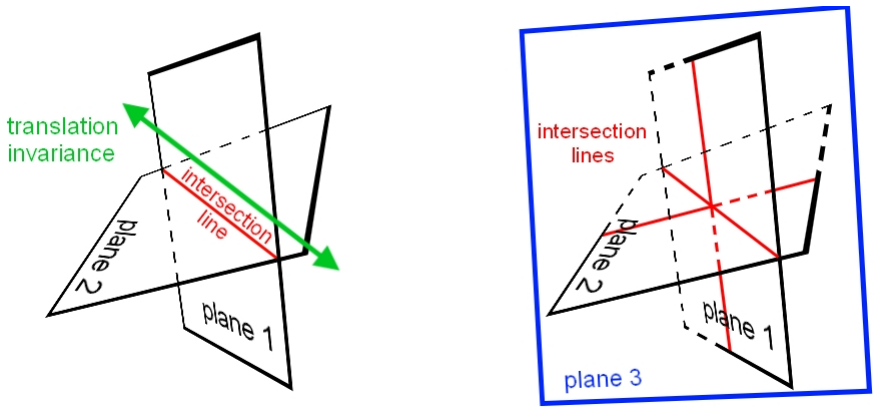


Figure 6.1: Transform invariances of intersecting planes

In indoor environments, planes are mostly all parallel or perpendicular to each other. For example a table plate is parallel to the flour as well as the surface of a cubicle is parallel to the wall behind it. Just in corners of a room can be found three planes with different orientation, for example two walls and the flour.

Due to the fact, that most planes in common indoor scenes are parallel, and thus are not sufficient to estimate a pose, visual features of the plane textures are also used in this work.

Using a TOF camera, this features are available in 3D space and not only as 2D projections. Planes of two successive frames can be associated, if visual features of their textures can be matched. In section 7 it is shown, how this features are used to calculate parts of the rotation and the translation. So it is possible to calculate the full pose with just one plane.

6.1 Extraction of the boundaries of detected planes

The feature extractor finds visual features in the 2D projection of the 3D scene. This can be the amplitude image of the TOF camera or the image of the RGB camera. Because the matching of the found features for successive frames should work if the camera is moved along its optical axis, the feature descriptors have to be scale invariant as the features can have different sizes. Only the projections of the plane textures are considered in the search for features. Thus it is important to know, which parts of an image belong to a plane and if the found feature is completely in such a part of the image.

In the region growing described in section 5.4.2, a plane identification number is assigned only to the center positions of the plane patches. Every plane patch has a size. So it is not sufficient to calculate the boundaries of the pixel sets with the same plane identification number because this would not take the size of the plane patches into account.

To calculate a planes contour in consideration of the size of plane patches, the center positions of its plane patches are drawn into a black image. This image is then dilated by an structure element of the size of a plane patch. This causes all pixels to be white, that have been used to calculate the equation of this plane in the region growing.

Then the contour is found by contour growing: The most upper right pixel is chosen from the white pixels and marked as contour pixel. The next contour pixel is found by searching its 8-neighborhood in the order shown in figure 5.11. The first white pixel that has been found is the next contour pixel. This is repeated until the first contour pixel is reached again.

6.2 Transform of plane equations and contours

A RGB webcam was additionally mounted beside the TOF camera. Thus the TOF camera can be used to find planes in the image, while the image of the RGB camera is used to find visual features on this planes. The alternative to the RGB webcam would be the amplitude image of the TOF camera, which is not as suitable because of its low resolution.

The following steps are performed after the plane recognition to obtain the plane contours for the RGB image:

1. Find the plane contours in the depth map.
   Plane boundaries are extracted from the depth map as described in section 6.1. The boundary is then stored in the data structure Contour2D. It contains all contour points.

2. Transform the 2D contours to 3D contour points.
   Each contour point is transformed to its 3D representation in the TOF coordinate system TOF. Section 4.5 describes how to perform this transformation. The intrinsic

calibration of the time of flight camera has to be known. The points are stored in the data structure Contour3D.

3. Transform the 3D contour points into the RGB camera coordinate system CAM.
   The contour points are transformed from the TOF coordinate system to the coordinate system of the RGB camera. The registration ${}^{TOF}T_{CAM}$ is used, which has been determined during the calibration (see section 4.4.5).

4. Projection of the 3D contour points into the RGB image.
   The 3D contour points in the RGB camera coordinate system are projected to the RGB image. This is done by multiplying the points with the camera matrix of the RGB camera. The perspectivic division (see equations 4.11 and 4.12) yields the 2D points of the plane contours, which have originally been detected in the depth image.

The left image of figure 6.2 shows plane contours, which have been determined from the depth map. The right image shows the RGB image and the contours, that have been transformed to it.

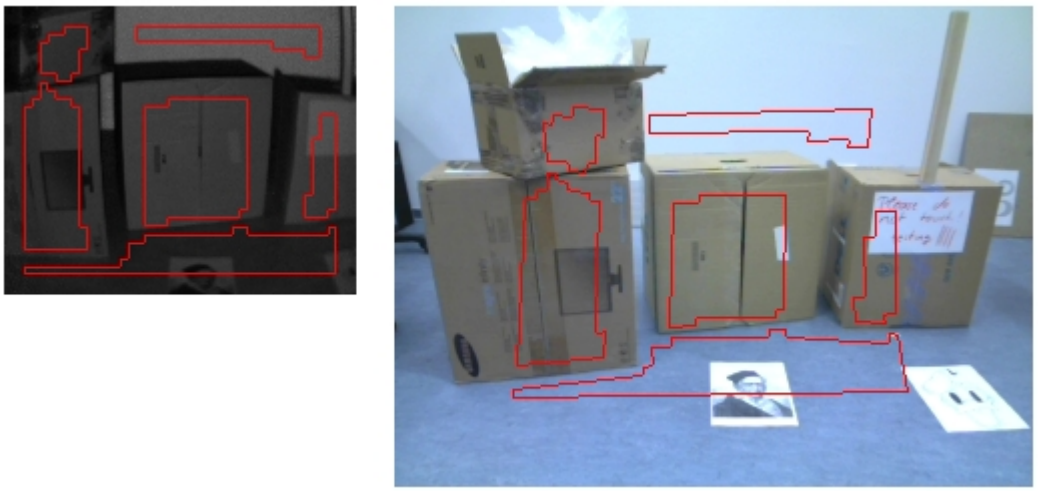


Figure 6.2: Transform of plane contours from AMP to CAM

In this work, planes are represented by equation 5.3. The transform of planes from the coordinate system TOF of the TOF camera to the coordinate system CAM of the RGB camera is separated into two steps. First, the normal vector of the plane in the TOF coordinate system is rotated by the rotation matrix of ${}^{CAM}T_{TOF}$, ${}^{CAM}R_{TOF}$

$$ {}^{CAM}\vec{n} = {}^{CAM}R_{TOF} \cdot {}^{TOF}\vec{n}. \tag{6.1} $$

Second, the fixed point in TOF coordinates ${}^{TOF}\vec{r_1}$ is first rotated by ${}^{CAM}R_{TOF}$ and then translated by ${}^{CAM}t_{TOF}$. This results in a new α in equation 5.3

$$ {}^{CAM}\vec{\alpha} = \frac{1}{\|{}^{CAM}\vec{n}\|} {}^{CAM}\vec{n} \cdot {}^{CAM}R_{TOF} \cdot {}^{TOF}\vec{r_1} + {}^{CAM}t_{TOF}. \tag{6.2} $$

6.3 Visual feature extraction and description

Visual features are used for two purposes in this work:

1. To assign planes of frame i-1 to the corresponding planes of frame i.

2. To determine the translation of the pose change from frame i-1 to frame i.

For both purposes it is important, that the features are contained completely in the plane regions. To ensure this, the features radius must be known. The minimal distance of the features position to the contour must then be higher than the features radius.

SIFT and SURF are two popular feature extraction and description techniques. Both are implemented in OpenCV. They differ primarily in the processing speed and the invariance to rotations along and away from the optical axis. In the following, a short compendium of the working principles and the characteristics of this two techniques is given. Then both techniques are compared regarding their attributes which are important to this work. For a detailed description of the techniques, it is suggested to read the original papers for SIFT [21] and SURF [6].

Both techniques work in two steps. Features are detected in the first step. In the second, these features are described by a vector, which is used for later matchings.

6.3.1 SIFT

SIFT [21] is the abbreviation for Scale Invariant Feature Transform. To achieve scale invariance, features are detected in scale space. Scale space is an image pyramid of successive blurred images. In SIFT, scale space are difference-of-gaussian images with increasing kernel sizes. An difference-of-gaussian image D is created by two slightly different blurred versions L of an image I:

$$\begin{aligned} D(x,y,\sigma) &= L(x,y,k\sigma) - L(x,y,\sigma) \\ &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= K(x,y,\sigma,k) * I(x,y). \end{aligned} \tag{6.3}$$

As can be seen from equation 6.3, instead of blurring image I two times with different gauss kernels G and subtracting the results, image I can also be convolved once by the kernel K. Depending on the scale level of the scale pyramid, this kernel can have a considerable size. Thus the convolution process is time consuming, as for a kernel size of $n \times n$, n^2 many floating point operations and additional n additions must be performed to calculate one pixel value of the result image.

Potential feature positions are found by searching local maxima in the 3D scale space. The Hessian matrix is used to determine the curvature of the feature and if it is considered as edge, it is rejected.

To achieve invariance for rotations along the optical axis, an orientation histogram of the gradient orientations around the features position is formed. The highest bins declare the features orientations, which are used to rectify the feature to norm orientation for its later representation.

The SIFT feature descriptor is build by sampling gradient orientations around the features position. A grid of 4×4 sampling regions is defined by the features orientation and scale, which have been computed in the previous steps. For every sampling region, a histogram is filled by the gradient orientations inside this region. Practically, the best results have been achieved with histograms of eight bins. So the description vector contains $4 * 4 * 8 = 128$ elements.

Matching of two feature sets is then performed by searching for each vector in one set, the vector with the smallest euclidean distance in the other set.

6.3.2 SURF

The idea behind SURF is approximately the same as behind SIFT. First, a scale space is created. Instead of blurring the image with gaussian kernels, SURF once calculates an integral image and then uses box filters. Box filters can be applied very fast and with speed independent of its filter size, as it needs just three additions to calculate the sum of the intensities over a rectangular box, with the use of an integral image. One more difference to SIFT in scale space representation is, that scale space layers are blob response maps for blobs with different scale, while SIFT uses difference-of-gaussian images. A blob response map is created by calculation of the determinant of the Hessian matrix for every pixel position. The Hessian matrix is usually formed by the results of convolving the image with derivatives of the gaussian kernel. This derivatives are approximated by box filters, which can be calculated very fast. Feature positions are local maxima in the blob response scale space. In contrast to SURF, SIFT calculates the determinant of the Hessian matrix just for local maximum positions in the DOG scale space.

To achieve orientation invariance, SURF calculates a main rotation for every feature. This is done by calculating the sum of the x- and y-gradients around the feature position. This gradients are determined by Haar filters, which can also be calculated very fast due to the use of an integral image.

The feature descriptor is then build up by sampling image gradients in a grid of 4×4 subregions around the image position. Every subregion is sampled at 5×5 sample positions. In contrast to SIFT, which is filling a gradient histogram per subregion, SURF accumulates the x-gradients, the y-gradients, the absolute values of the x-gradients and the absolute values of the y-gradients. This results in a feature vector of 4*4*4 = 64 dimensions. There is also a version of SURF available, that constructs feature vectors of 128 dimensions. In this version, positive and negative values of x- and y- gradients are considered separately.

6.3.3 Comparison of SIFT and SURF

From the design of the algorithms SIFT and SURF it is clear, that SURF is much faster, because it uses integral images and approximates the very time consuming filtering with gauss blurring filters and its derivates by box filters. Also the gradient information for orientation assignment or feature description is generated in the original image by fast haar filters.

For this work, it was important to find the weaknesses of each algorithm and the best tradeoff between the good results of slow feature extraction techniques and higher frame

rates that can be reached by using fast techniques. High frame rates cause smaller pose changes for the same movement of the TOF camera.

The following comparison relies on contemporary literature. SIFT is compared with SURF in respect to repeatability and number of matches for scale changes, rotation and view point changes. Also processing times are reflected as well as the amount of correct matches.

Some of this invariances are very important for this work. Scale change is produced by moving the camera along its optical axis. Rotation originates from rotating the camera along its optical axis, while affine transformations are caused by a rotation away from the optical axis. Effects on image blur or illumination changes are of secondary importance and thus not compared.

1. Processing time
   In [5] SIFT and SURF are directly compared with its implementations in OpenCV. For 508 image pairs, features have been detected, described and matched. In average, SIFT needs 13 times as long as SURF per image pair.

2. Scale changes
   [20] shows, that SIFT and SURF perform approximately equal for scale changes. In general, SIFT finds a few more correspondences than SURF.

3. Image rotation along the optical axis
   The most crucial weakness of SURF against SIFT is its behavior to image rotations along the optical axis. This weakness results from the approximations of continuous gaussian kernels with boxes. A box filter is anisotrop and therefore not applicable for rotations. [20] points out, that also SIFT looses up to 40 percent repeatability for rotations between 5° and 85°. SURF behaves like SIFT for the first 10°, than its repeatability decreases to 10 percent (see [20]).

4. Image rotation away from the optical axis
   Both, SIFT and SURF are not invariant to view point changes as shown in [20]. Both behave similar on view point changes. If the same scene content is seen by just a difference in angle of 10°, both SIFT and SURF loose approximately 50 percent of its repeatability. 30° view angle change causes SIFT to have approximately 20 percent repeatability and SURF 10 percent left. No matches are found by both descriptors with an view angle change of 50°.

5. Number of matches
   [30] shows, that SIFT generally finds significantly more matches. On the other hand, the ratio of correct to total matches is better with SURF (79 percent) than with SIFT (67 percent).

The very important weakness of SURF is its bad behavior to image rotations along the optical axis, but its advantage is its speed. As SURF needs less time for detection, description and matching, the frame rate is significant higher than with SIFT. This causes less pose changes between the captures of successive frames and thus compensates the weakness of SURF.

This led to the decision to use SURF.

6.3.4 Selection of relevant key points

As described in section 6.1, just the features, which lie completely on a plane are considered for pose calculation.

To realize this, the contour of the boundary of each plane is determined. Then visual features are extracted in the complete image. For each feature position it is checked, whether it lies inside a contour of a plane and the minimal distance to this contour is calculated. If the feature is inside a contour and the minimal distance is higher than the features radius, this feature is considered further while all other features are discarded.

Figure 6.3 illustrates the result of feature detection on the amplitude image (up-left) and the RGB webcam image (down-left). The SURF parameters have been the same for both feature detections. As can be seen in this figure, the number of found features is significant higher for the webcam image. 6.3 up-right and down-right show the features, that remain, because they lie completely on a plane.

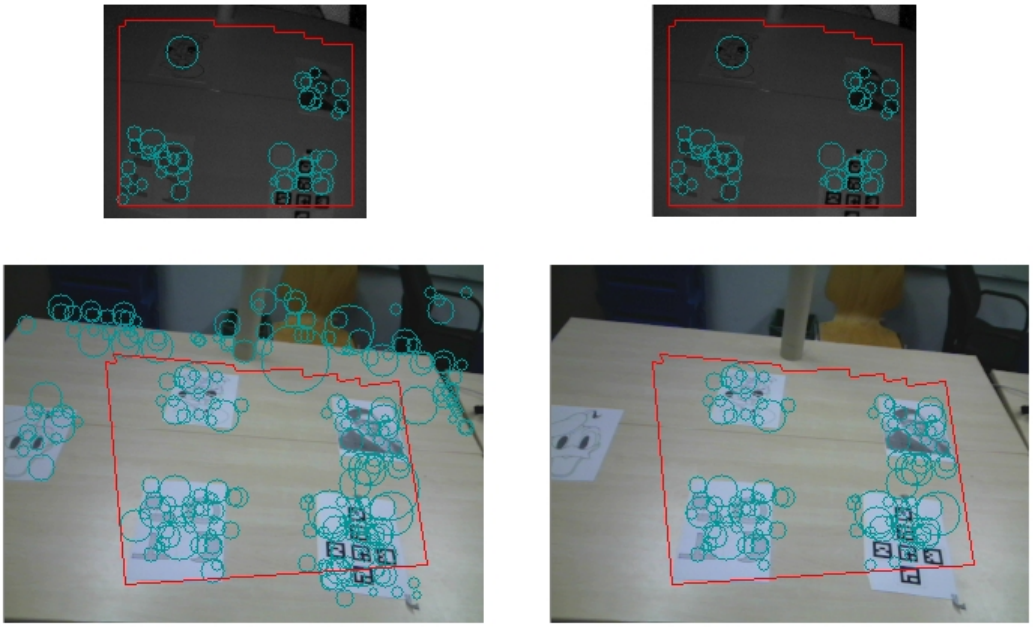


Figure 6.3: Discarding of features outside of planes

6.4 Calculation of 3D coordinates of the feature points

There are two possible ways to determine the 3D positions of the SURF features. One method is to calculate the 3D position from the depth map as described in section 4.5. The accuracy of this 3D positions accords to the measurement accuracy of the TOF camera.

The second and better way is to first determine the 3D positions as described in section 4.5 and then to project this positions orthogonally onto the estimated planes. The planes

are calculated by a large set of points which causes averaging. Thus, the projection of points onto this planes is a refinement.

7 Pose estimation

This section describes how feature point correspondences are found between successive frames and how this correspondences are used to calculate the pose change between this frames.

7.1 Descriptor matching

OpenCV provides functionality to match feature vectors with FLANN. Muja and Lowe developed FLANN [25] (Fast Library for Approximate Nearest Neighbors) to match large sets of high dimensional vectors. FLANN uses either a hierarchical k-means tree or multiple randomized kd-trees to match these two sets. It depends on the desired precision and the dataset, which of these data structures are chosen.

The drawback of FLANN is, that this trees have to be build first, before the matching can be performed. If a data set is used very often, FLANN causes a huge acceleration of the matching processes, as the tree must be build once in the beginning and can be used again for every matching. In this work, the feature descriptors of frame i-1 and frame i are matched. With the capture of frame i+1, the feature descriptors of frame i-1 are not required any more. As FLANN builds a complex data structure for every frame, which is used just a few times for matching, this method is in this context not time saving, but time consuming. Also normally there are not more than 200 key points per image, which is much too less for a profitable use of FLANN.

As there are just a few key points per image and the dimension of the feature vectors is just 64, a naive nearest neighbor (NNN) search was implemented. For a given vector, the dataset is searched for the vectors v_1 and v_2 with the smallest d_1 and the second smallest d_2 euclidean distance. If $d_1 < \tau_{threshold} \cdot d_2$, the given vector is considered distinctive and the vector v_1 is the matching. $\tau_{threshold}$ was set to 0.6.

The consumption of processing time of FLANN and the naive nearest neighbor matching have been compared in this work. Four pairs of feature descriptor sets have been matched in the test. In average, FLANN found 34.8 and the native nearest neighbor search 35.8 correspondences while FLANN needed 53.6 ms and the NNN 1.5 ms. This led to the use of a native nearest neighbor search.

7.2 Using 3D positions of the features to estimate the pose change

The 3D positions of the 2D feature points are directly computable by their depth map values and the equations 4.14 and 4.15. The matching of the 2D plane texture features between successive frames can thus be used to get two sets of corresponding 3D points. Section 4.4.1 describes, how the appropriate coordinate transformation can be calculated

by this set of 3D point correspondences. It is suggested to use Horns quaternion method instead of the least square method, as it gives a orthonormal rotation matrix.

At least four point correspondences must be available to calculate the pose change with Horns method. As can be seen from the tests in section 4.4.1, Horns method does not provide good results for few point correspondences. At least 50 point correspondences should be used to calculate the transform. The plane regions of the amplitude images of the Time-Of-Flight camera do not give enough points for the method of horn to give accurate results.

7.3 Using feature correspondences and the plane normals to estimate the pose change

Another method to calculate the pose change between two frames is to use plane normals to calculate the rotation R. Planes are calculated from normally some hundreds of points and thus provide an averaging. The idea is, to bring the normals of related planes together for each feature point correspondence. After that, a rotation along the plane normal must be done, to bring the feature positions together. These two steps determine the rotation. They are followed by the calculation of the translation by the centroids of the two point sets. The following steps are performed to estimate the pose change:

1. Project the feature positions on their planes:
   Project all 3D feature points p_i of the current frame and all 3D feature points q_i of the last frame orthogonally onto their planes. This gives the new 3D feature positions p'_i and q'_i.

2. Calculate the centroids of the feature positions:
   Calculate the centroids c_p and c_q of the points p'_i and q'_i.

3. Calculate the centroids of the feature positions on each plane:
   For every plane of the current and the last frame, calculate the centroids $m_{last,j}, m_{current,j}$ of all feature points on it.

4. Discard features, that are alone on a plane or too near together:
   Discard all feature points that are near to the centroid of the feature points of their related plane. Discard also the correspondences with the related feature point of the other frame: If $\|p'_i - m_{current,j}\| < threshold$, p'_i and q'_i are discarded. If $\|q'_i - m_{last,j}\| < threshold$, p'_i and q'_i are discarded. This is done, because if the feature point is near or equal to the centroid of the planes features, all features of this plane are too near together to calculate the correction along the plane normal described in step 5b.

5. Calculate the rotation R_i for every correspondence (p_i, q_i):
   For every feature point correspondence (p_i, qi):
   a) Rotate the normal vector $\vec{n}_{last}$ of the last frame to the normal vector $\vec{n}_{current}$ of the current frame.

- Rotation angle: The rotation angle α_i is the angle between the vectors $\vec{n}_{last}$ and $\vec{n}_{current}$.
$$\alpha_i = acos\left(\frac{\langle \vec{n}_{last}, \vec{n}_{current} \rangle}{\|\vec{n}_{last}\| \, \|\vec{n}_{current}\|}\right).$$
- Rotation direction: The direction of the rotation vector is orthogonal to $\vec{n}_{last}$ and $\vec{n}_{current}$ in such a way, that $\vec{n}_{last}$ is rotated to $\vec{n}_{current}$ via the right-hand rule.
$$\vec{d}_{1,i} = \frac{\vec{n}_{last} \times \vec{n}_{current}}{\|\vec{n}_{last} \times \vec{n}_{current}\|}$$
- Rotation vector: The rotation vector $\vec{r}_1$ has the length α and the direction $\vec{d}_1$.
$$\vec{r}_{1,i} = \alpha_i \vec{d}_{1,i} = acos\left(\frac{\langle \vec{n}_{last}, \vec{n}_{current} \rangle}{\|\vec{n}_{last}\| \, \|\vec{n}_{current}\|}\right) \frac{\vec{n}_{last} \times \vec{n}_{current}}{\|\vec{n}_{last} \times \vec{n}_{current}\|}$$
- Calculate the rotation matrix $R_{1,i}$ from the rotation vector $r_{1,i}$. The conversion of a rotation matrix to a rotation vector or vice versa is described in [10].

b) Rotate around $\vec{n}_{current}$, to bring p_i and qi together.
In this step, the vector $\vec{v} = R_1(q_i - m_{last,j})$ is rotated around $\vec{n}_{current}$, to be congruent with the vector $\vec{w} = p_i - m_{current,j}$.
- Rotation angle: The rotation angle β_i is the angle between the vector $\vec{v}$ and $\vec{w}$.
$$\beta_i = acos\left(\frac{\langle \vec{v}, \vec{w} \rangle}{\|\vec{v}\| \, \|\vec{w}\|}\right).$$
- Rotation direction: The rotation direction is
$$\vec{d}_{2,i} = \frac{\vec{n}_{current}}{\|\vec{n}_{current}\|}$$
.
- Rotation vector: The resulting rotation vector is
$$\vec{r}_{2,i} = \beta_i \vec{d}_{2,i} = acos\left(\frac{\langle \vec{v}, \vec{w} \rangle}{\|\vec{v}\| \, \|\vec{w}\|}\right) \frac{\vec{n}_{current}}{\|\vec{n}_{current}\|}$$
- Calculate the rotation matrix $R_{2,i}$ from the rotation vector $r_{2,i}$.

c) Calculate the rotation matrix for this feature correspondence by
$$R_i = R_{2,i} R_{1,i}.$$
Calculate the rotation vector r_i from the rotation matrix R_i.

6. Calculate the rotation R:
The rotation vectors r_i must be very similar, because the represent the same rotation. All rotation vectors are averaged to get the rotation vector r. The rotation matrix R is then calculated from the rotation vector r.

7. Calculate the translation t:
   The translation is calculated by the centroids of the point sets:

$$t = c_p - Rc_q$$

7.4 Removing outliers

Feature point correspondences can be wrong, especially in situations of view point changes. If all found correspondences are included to the calculation of the pose change, this would lead to bad results. Thus the wrong correspondences must be found and discarded, before the pose change is determined.

Even as the SURF feature descriptors are distinctive, wrong matches are possible. For example if there are similar structures more than once in the images. If the wrong matches are included into the pose change calculation, the estimated pose change will be wrong. Thus it is necessary to find out, which feature point correspondences are correct, that is they are inliers, and which correspondences are wrong, so called outliers.

Random Sample Consensus [13] (RANSAC) is an algorithm, that identifies outliers from a set of measurements, that comply with a model. The input parameters of RANSAC are:

- The minimum probability p, of finding the set of inliers.
- The number n of measurements, that is necessary to calculate the model.
- The ratio ϵ of the number of outliers to the number of all measurements.
- The threshold t of the distance between a measurement and the model, that declares a measurement as outlier.

RANSAC works as follows:

1. Calculate the number k of samples:
   From the input parameters, it is determined, how many samples are chosen from the measurement set. A sample consists of n randomly chosen points, where n denotes the minimum number of measurements that is necessary to calculate the model. The number of samples is independent of the number of measurements and is calculated by
$$k = \frac{\log(1-p)}{\log(1-(1-\epsilon)^n)}.$$
2. Calculate the model for every sample.
3. For each model determine the number of measurements, that support the model:
   A measurement, that supports the model has a distance less than t to the model.
4. Choose the sample with with the most supporting measurements.
5. Mark outliers:
   Mark all measurements that do not support that sample, that is their distance to the model is greater than t, as outliers.

In this work, the model is the estimated pose change and the measurements are the feature point correspondences. The number n of measurements, that is needed to calculate the model, depends on the method that is used to calculate the pose change. If the 3D positions of the feature points are used directly to estimate the pose change (as described in section 7.2), four feature point correspondences are needed. If the pose change is calculated using plane normals as described in section 7.3, only two feature point correspondences of the same plane are needed. One feature point correspondence would not be enough to calculate the rotation around $\vec{n}_{current}$ (see step 5b of the algorithm described in section 7.3).

Table 7.4 lists the required number k of samples for a probability p=99.0% and the two described methods for the pose change determination.

method	**n**	ϵ								
		10%	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
with plane normals	2	3	5	7	11	17	27	49	113	459
3D points only	4	5	9	17	34	72	178	567	2876	46050

Table 7.1: Iteration numbers of RANSAC for pose estimation

The maximum ratio ϵ of the number of outliers to the number of measurements was set to 0.5, which is a pessimistic guess. As can be seen, the method that uses plane normals to calculate the pose change needs significant less samples than the method that uses 3D point correspondences. So this method was chosen as model estimation for the RANSAC algorithm.

As just two correspondences are used instead of all correspondences, the algorithm described in section 7.3 was adapted: In step 4, all feature correspondences are discarded except the two features on the same plane, that are used as samples.

Figure 7.1 shows two successive frames with the found feature point correspondences. As the camera did not move between these frames, many correspondences have been found and the most of them are inliers. The left part of the figure shows the amplitude images and the right part the images of the webcam. 22 point correspondences have been found in the amplitude images and 57 in the webcam images. RANSAC found two outliers, which have been drawn blue.

To generate two successive frames with a high ratio of outlier correspondences, the pose was changed in all six degrees of freedom. Figure 7.2 shows these correspondences. As was shown in section 6.3.2, SURF is not robust to large view point changes. This causes nearly the half of the point correspondences to be outliers (painted in blue). These have been identified successfully with the use of RANSAC.

After the outliers have been found, the pose can be determined from all inliers by either one of the two methods described in the previous section. The method, that uses plane normals is used in this work, because the planes provide an averaging of many points. Thus this method is more robust to noise.

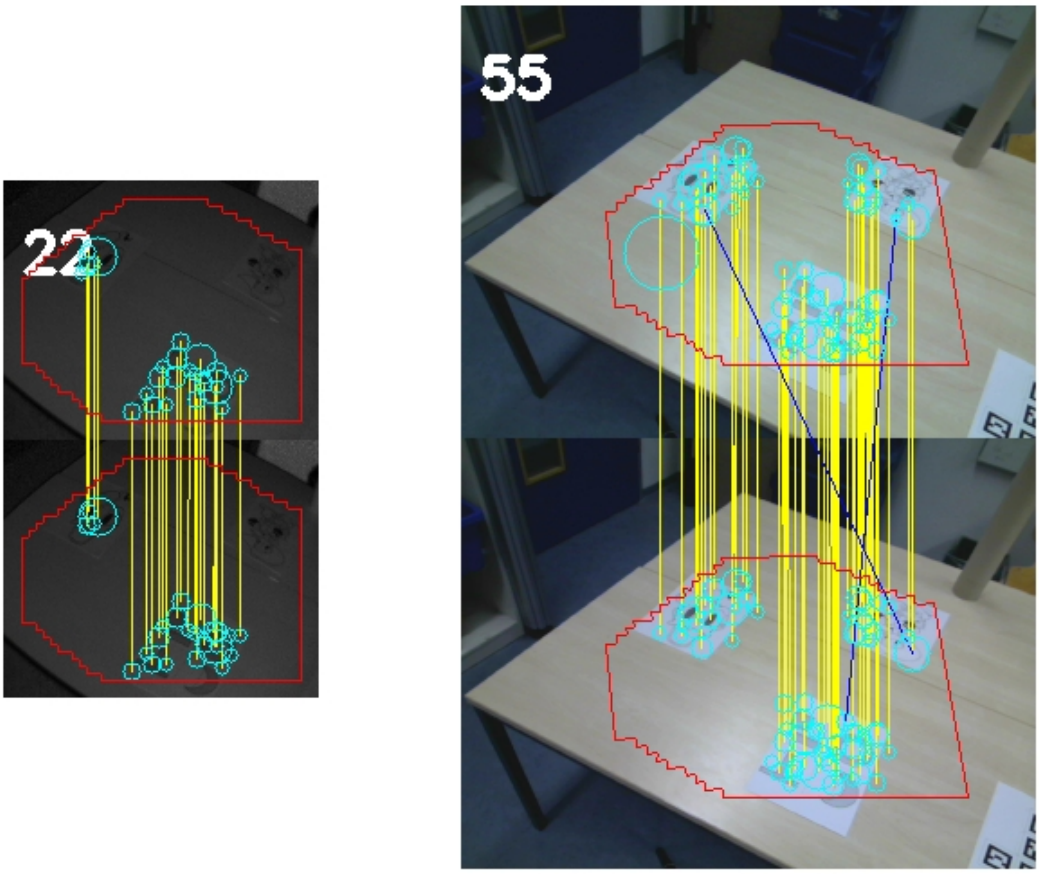


Figure 7.1: RANSAC: discarding of wrong correspondences

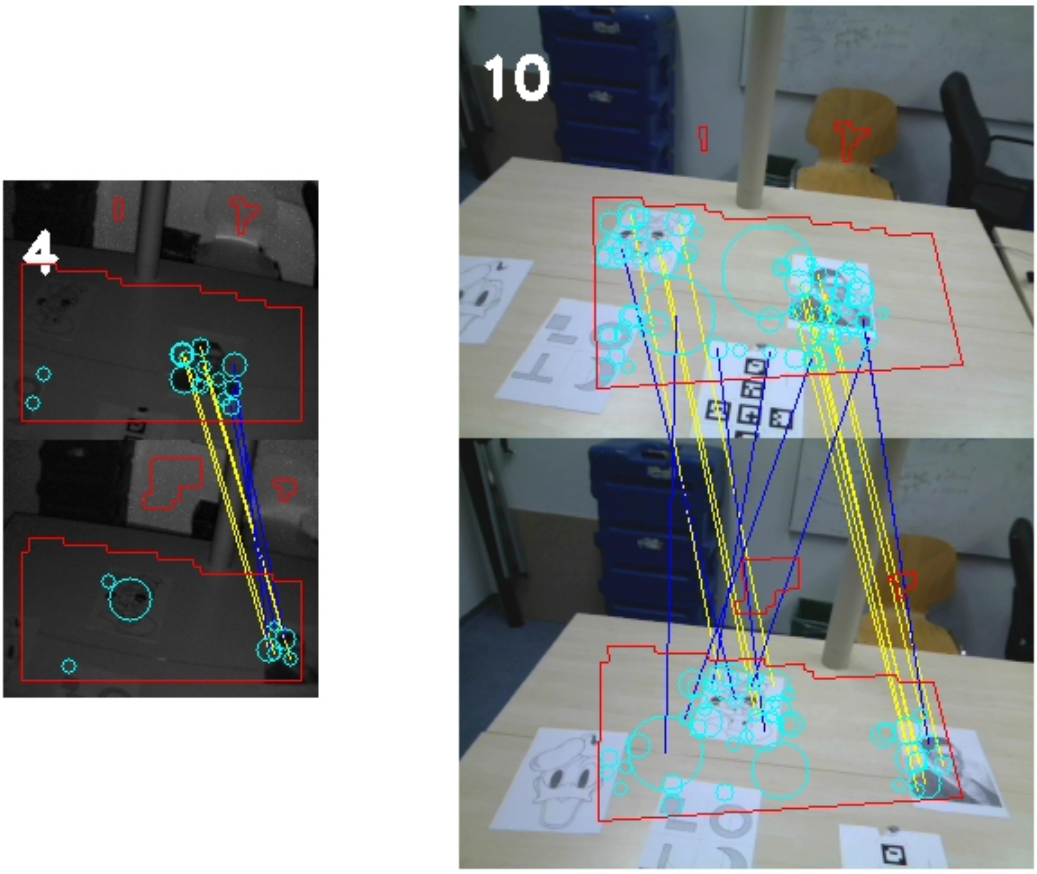


Figure 7.2: RANSAC: discarding many outlying correspondences

8 Evaluation

8.1 Evaluation with the ART measurement system

The method, how to evaluate the accuracy of the estimated pose change, was specified at the beginning of the work: It was intended to use a system to track a rigid body of the company ART for the ground truth. This system gives the absolute pose of a rigid body consisting of infra red light reflecting balls in the ART world coordinate system.

The pose change is determined as the transform of the TOF coordinate system of frame i-1 to the TOF coordinate system of frame i: ${}^{TOF_{i-1}}T_{TOFi}$. Thus the pose changes of the rigid body do not interest, but the pose changes of the TOF camera coordinate systems. For this reason the pose of the TOF coordinate system in body coordinates ${}^{BODY}T_{TOF}$ (which has been determined during the calibration process as described in section 4.4.5) is transformed by ${}^{ART}T_{BODY}$ to determine the pose of the TOF coordinate system in ART world coordinates:

$$ {}^{ART}T_{TOF} = {}^{ART}T_{BODY} \cdot {}^{BODY}T_{TOF}. \tag{8.1}$$

The pose change in TOF coordinates between two frames can then be calculated as the transform between the poses ${}^{ART}T_{TOF}$ of the two frames:

$$ {}^{TOF_{i-1}}T_{TOFi} = ({}^{ART}T_{TOF_{i-1}})^{-1} \cdot {}^{ART}T_{TOF_i} \tag{8.2}$$

Some test series have been produced, where the pose change has been estimated by the algorithm developed in this work and the ART poses have been read and saved for every frame. The pose changes in TOF coordinates have then be calculated using the ART poses as described before. For every frame, the pose change ${}^{TOF_{i-1}}T_{TOFi}$ has then be inverted and its 3D points transformed by this inverse pose change. For each frame, the so generated points have then be added to a point cloud, that contains the so produced points of the previous frames. This point cloud has been visualized with VTK. Figure 8.2 shows the scene and figure 8.3 shows the accumulated point cloud for three frames.

If the described method for the generation of the ground truth poses would work accurately, the 3D point cloud of the scene would grow with every frame in such a way, that the parts of the scene, that are visible in both frames overlap completely. As can be seen in figure 8.3, the table plate is not overlapping completely, because there is a difference in the angles of the scenes. The so produced 3D scenes are growing with every frame and reproduce the movement of the TOF camera, but a small error is visible for every frame.

For comparison, the same growing 3D scene was produced with the pose changes calculated by the algorithm that was developed in this work. The 3D scene for the first three frames is shown in figure 8.1. This figure shows, that the ART measures of the TOF camera poses could not be used as ground truth, as the developed algorithm gives better results.

As the angle measurement of the rigid body with the ART system is very precise, the measure of the pose of the rigid body cannot be the reason for the bad accuracy of the

Figure 8.1: Scene for a rough test of the pose change measure with the rigid body

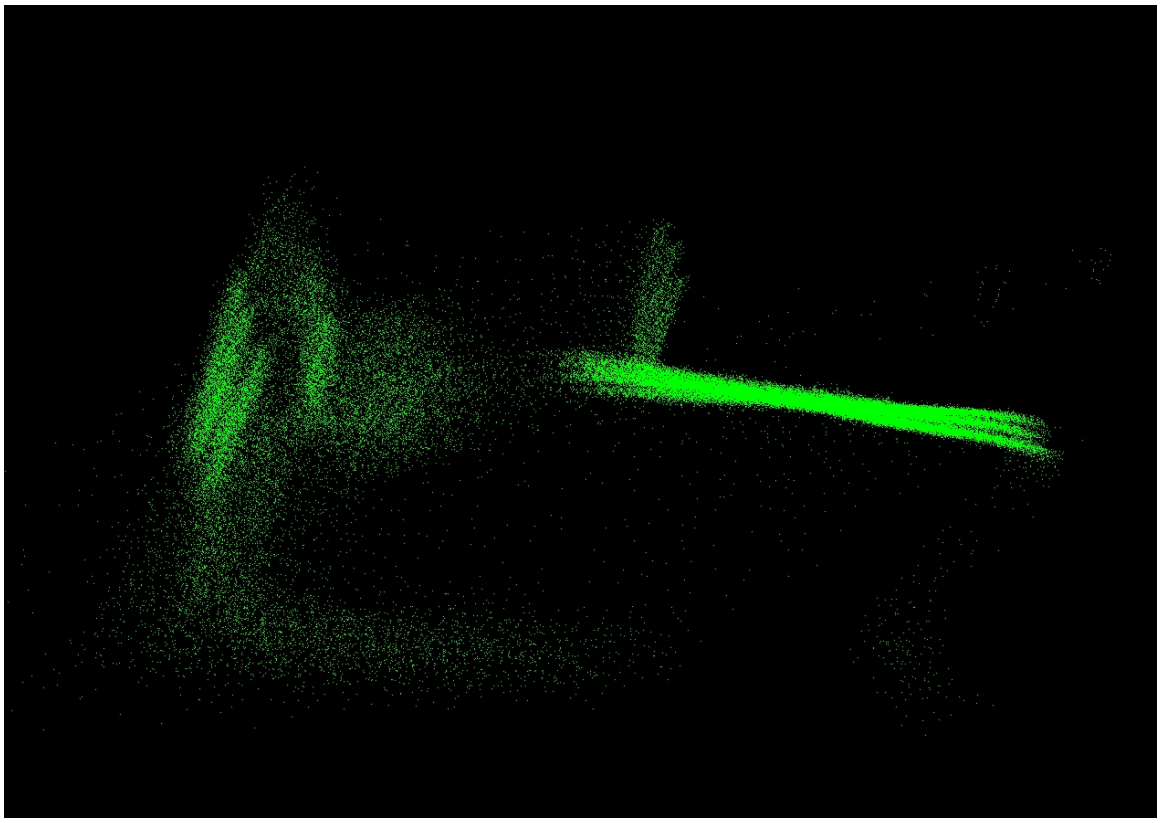


Figure 8.2: Pose change in the TOF coordinates estimated with the rigid body

described ground truth method. It is assumed to be caused by an imprecise determination of the registration between the coordinate system of the rigid body and the coordinate system of the TOF camera. This registration was described in section 4.4.5 and depends

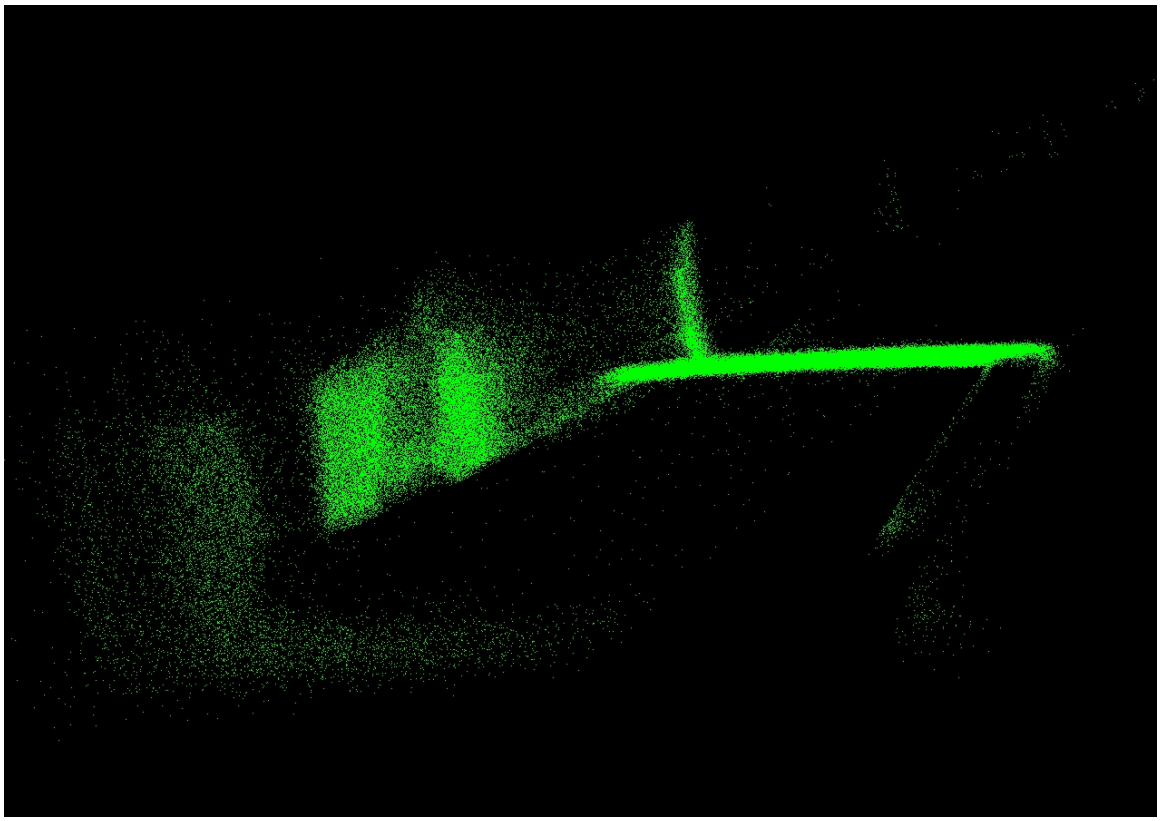


Figure 8.3: Pose change in the TOF coordinates estimated with the proposed algorithm

on three steps: First, the registration of a chessboard pattern in ART world coordinates, second the registration of the rigid body in ART world coordinates and third the pose estimation of the TOF camera to the chessboard pattern.

The accuracy of the first two steps can be supposed high, because the ART measurement system has a high accuracy. By contrast the accuracy of the estimation of the cameras pose to the chessboard pattern depends mainly on the following parameters: The resolution of the amplitude image, the size of the chessboard pattern in the image, the accuracy of the intrinsic camera parameters and the accuracy of the chessboard corner detection. As the amplitude images of the TOF-camera have very low resolution, a chessboard pattern with only 5x4 corners was used. The corner detection was performed by OpenCV and did not provide good results, as can be seen in figure 4.4. This is explicable by the low signal to noise ratio of the amplitude images. An imprecise measure of the chessboards pose in camera coordinates yields an inaccurate registration of the coordinate system of the rigid body to the coordinate system of the TOF camera.

The accuracy of this registration could not be improved, so the evaluation had to be divided into two parts. The rotation angle can be evaluated with the ART system, because the rotation angle of the rigid body has to be the same as the rotation angle of the estimated pose change. Due to the fact, that the translation depends on the previously calculated rotation, the measurement of the translation accuracy can be done only for movements without rotation. For this reason, the camera attachment was mounted to a electronically

(a) Scene 1 for rotation evaluation

(b) Scene 2 for rotation evaluation

Figure 8.4: Evaluation of the rotation angle

controllable linear actuator. In this way, pure translation movements with given distances could be performed.

8.2 Test method and test scenarios

8.2.1 Rotation

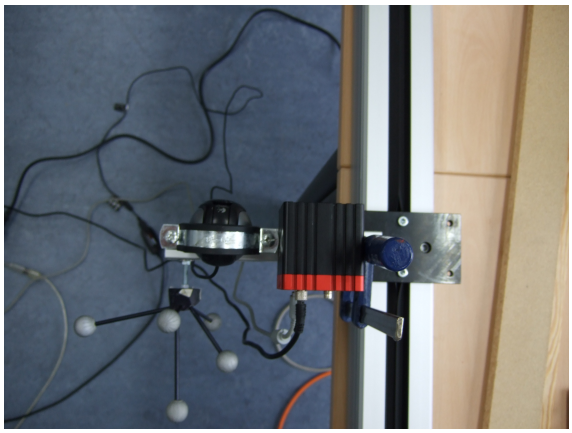
The pose change of the TOF camera was on the one hand calculated with the ART system as described in the previous section and on the other hand it was determined by the algorithm developed in this work. Both rotation matrices have been transformed to rotation vectors. For rotation evaluation, the lengths of the rotation vectors are compared.

Two test scenarios have been used. One with a single table plate (figure 8.4(a)) and one with a shelf (figure 8.4(b)). For the first test, the camera was mounted to a tripod while the second test was performed free-hand. Together 25 frames have been captured. The ART system measured rotations of up to 6.9 degree between these frames. The average frame-to-frame angle error of the rotation with the camera attachment mounted on a tripod was 0.64 degree and for free-hand movements it was 0.71 degree. The maximum angle error was 1.48 degree.

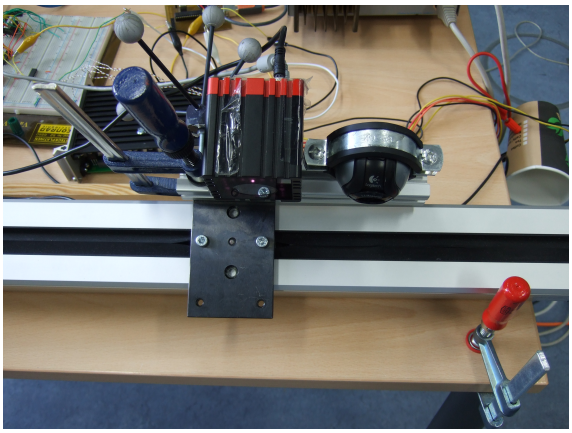
8.2.2 Translation

As the estimated rotation is containing errors and this errors are directly affecting the translation, it was necessary to prevent rotations in the evaluation of the translation. For this reason, the camera attachment was mounted to an electronically controllable linear actuator. The translation distances for the test are 10 cm and 20 cm. As the used linear actuator is very precise, it was considered as ground truth. The test setup is shown in figure 8.6(a).

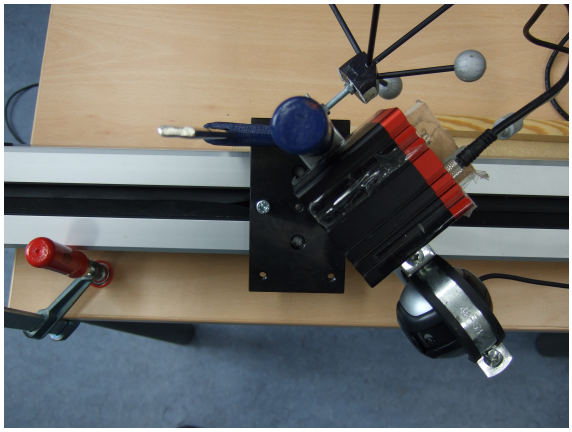
Fourteen tests have been carried out with different scenes (that can be seen in figure 8.6). The camera attachment was mounted in different ways, shown in figure 8.5, to perform



(a) Mainly motion in Z-direction

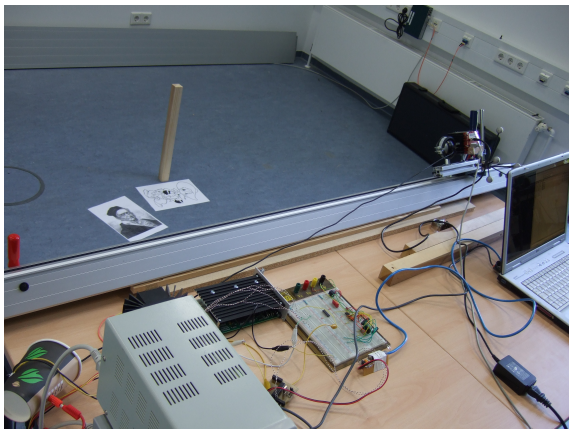


(b) Mainly motion in X-direction



(c) Mainly motion in X-and Z-direction

Figure 8.5: Mounting of the camera attachment for translation evaluation



(a) Scene 1 for translation evaluation

(b) Scene 2 for translation evaluation

(c) Scene 3 for translation evaluation

(d) Scene 4 for translation evaluation

(e) Scene 5 for translation evaluation

Figure 8.6: Evaluation of the translation distance

movements in mainly x-direction (fig. 8.5(b)), z-direction (fig. 8.5(a)) and x-z-direction (fig. 8.5(c)).

123 pose change estimations have been tested for translations of 10 cm. The overall average frame-to-frame error was 1.49 cm, which is 14.9% of the translation distance, and the maximum error 6.76 cm. The best results have been achieved with movements in Z-direction. The average error was 0.14 cm for translations in this direction.

For translations of 20 cm, 51 pose changes have been tested. An overall average frame-to-frame error of 2.24 cm was measured, which equates to 11.2% of the translation distance. The maximum error was 14.0 cm. Like for translations of 10 cm, the best result could be achieved with movements in Z-direction. The mean error of this movements amounts to 2 mm, which is 1% of the translation distance.

Discussion of the results

The tests have shown, that the accuracy of translations in X-direction is relatively low, compared to translations in Z-direction. Both, rotation and translation depend directly on the quality of the 2D feature point positions. For example, if a feature position is detected with an error of one pixel in u-direction, the 3D position of this feature, that is calculated from the depth map, would have an error of more than 1cm, if the 3D feature is 2.5 m away from the camera. This is calculated by

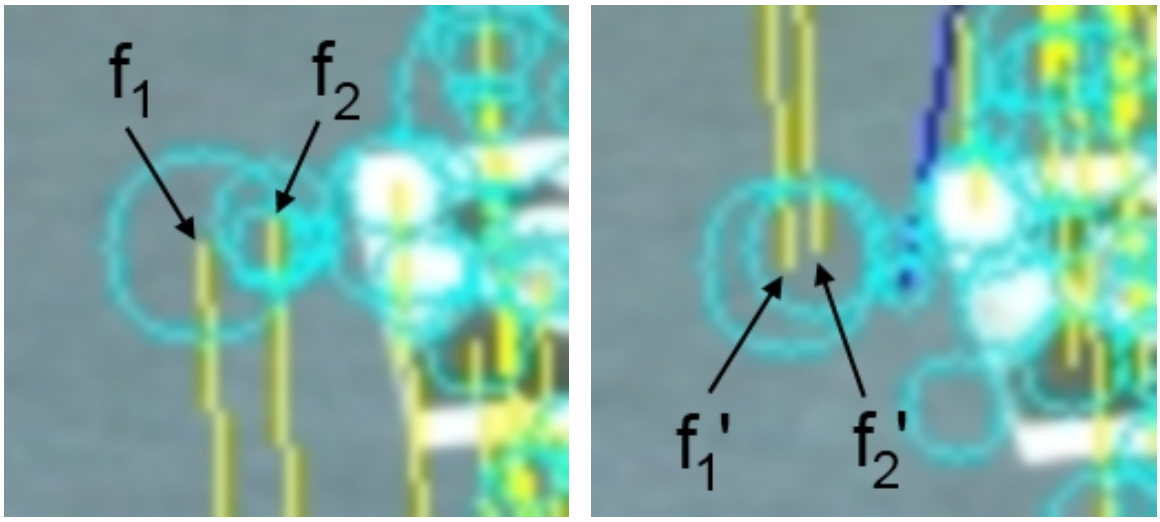
$$\Delta X = \frac{\Delta u \cdot Z}{k_u f}, \tag{8.3}$$

where $\Delta u = 1$ and $k_u f$ is determined with the intrinsic camera calibration (see section 4.3) and is approximately 250 pixel for the TOF camera and the RGB webcam. Figure 8.7 illustrates an example of SURF feature positions that are containing drift errors in u- and v-direction. The relative positions of the features f_1 and f_2 of the first frame and f_1' and f_2' of the second frame, that are illustrated in figure 8.7(a), are not the same in this two frames. This leads to the determination of errored 3D feature positions and finally to bad results in the estimation of rotation and translation. As the SURF feature positions can have drifts in u- and v-direction (u and v are the pixel coordinates of the features), it can be assumed, that the pose estimation achieves also relatively low accuracy for translations in Y-direction.

Better results could be obtained for translations along the Z-direction. This is due to the fact, that the detected planes are an averaging of its generating points and the 3D feature points are orthogonally projected on this planes (see section 7.3). The orthogonal projection of the 3D features onto the planes eliminates the noise of the depth determination of the TOF camera, because the detected planes in the tests have been almost parallel to the image sensor. Also drift errors in the 3D feature positions do not matter, as these errors affect just the X- and Y- components of the 3D features after the projection on the detected plane. Thus the Z-values of the 3D feature points are very precise which results in a high accuracy for translations along the optical axis of the camera.

8.2.3 Processing time

The evaluation of processing time was performed on the two scenes that are shown in figure 8.4. They have been chosen, because large areas of the first scene are planes and



(a) SURF feature points (b) corresponding feature points

Figure 8.7: Error of the positions of SURF features

many features could be matched while the second contains just small planes with less features.

The complete processing time for frame-to-frame registration of the first scene was 0.44s with 43 correct feature correspondences. For the second scene just 0.29s are needed, with 4 detected correct feature correspondences.

The test was performed on a Intel Core2 Duo CPU with 3.0 GHz and 2GB RAM.

9 Sample Applications

Two sample applications have been implemented. They both use the pose estimate of the proposed algorithm.

9.1 3D map generation

3D map generation of the environment is illustrated in figure 9.1. The sensor movement is inverted to align the point cloud of each new frame to the already build 3D map. Figures 9.1(a) and 9.1(b) show the RGB image of the first and the last frame.

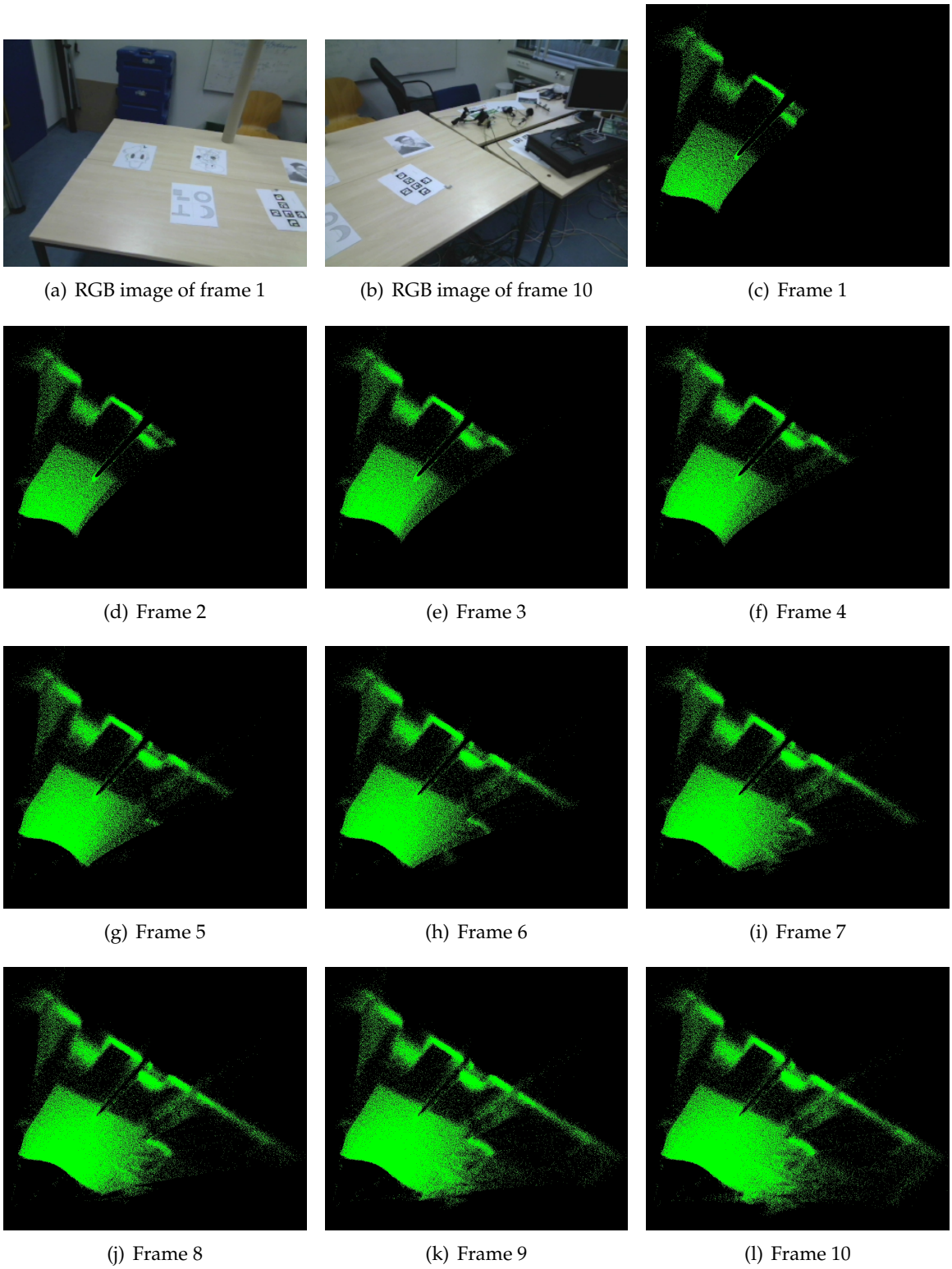
Especially indoor environments are suitable for mapping with this algorithm, because planes can be seen from nearly every point of view (for example the ground, walls, table plates or the ceiling). The proposed algorithm estimates all six degrees of freedom of the pose change and thus complete indoor environments, can be mapped.

9.2 Augmentation of a cube

Placing virtual objects into a scene that is seen by a camera is one of the central issues of augmented reality. If an object should stay on a fixed position in this scene or should move relative to it while the camera is moving, the motion of the camera has to be estimated. Using the proposed algorithm for this task brings an additional advantage: Objects are often placed on planes, be it table plates, be it the ground or be it other planar objects. This can be done easily because the plane equations are known by the algorithm.

The sample application first detects and visualizes the planes. The user can then place a virtual 3D cube directly on the planar surface of the scene object by clicking on the desired position in the image.

The application lets the cube stay in a fix relative pose to the scene, independently of the motion of the camera, as can be seen in figure 9.2.



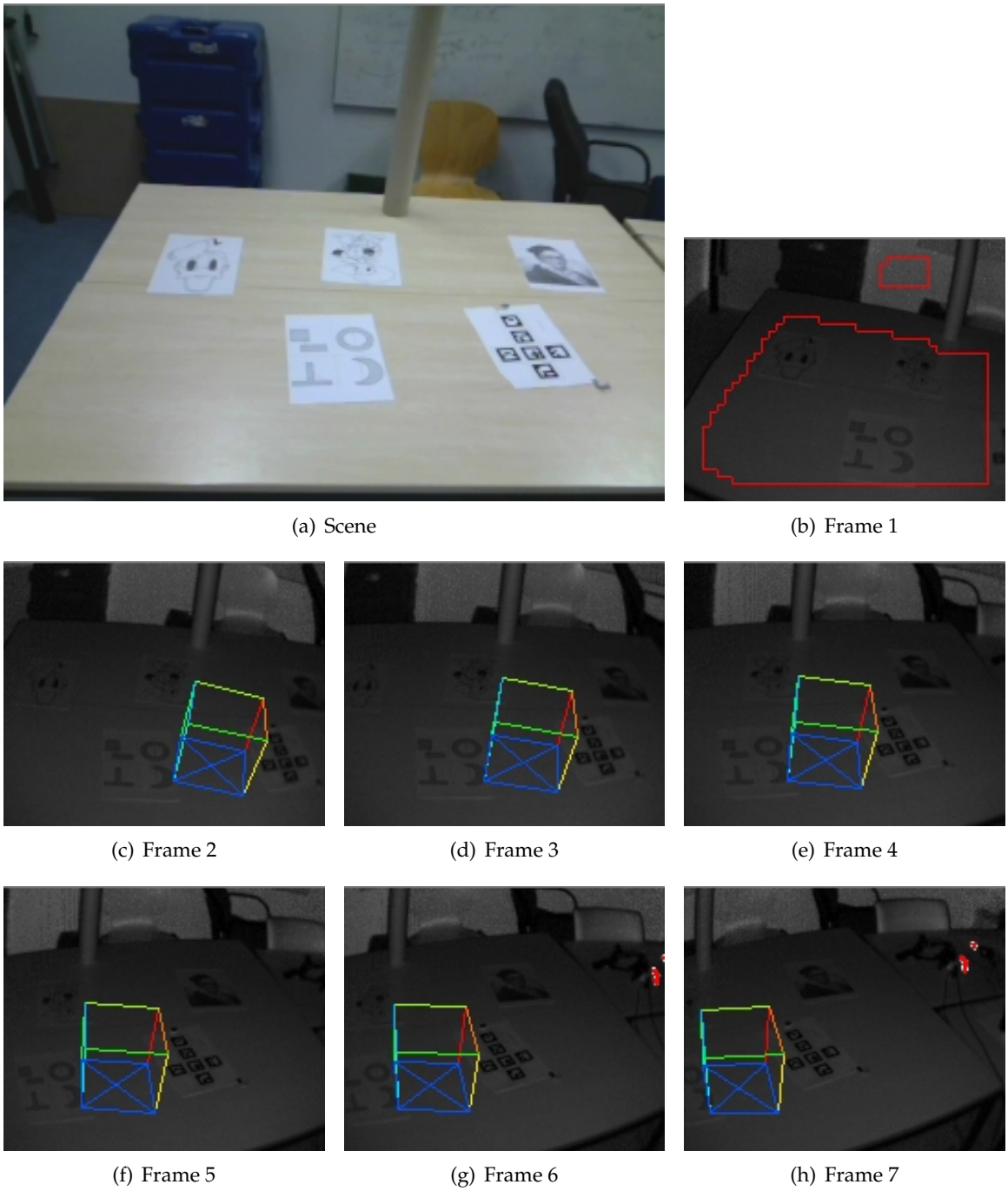
(a) RGB image of frame 1 (b) RGB image of frame 10 (c) Frame 1

(d) Frame 2 (e) Frame 3 (f) Frame 4

(g) Frame 5 (h) Frame 6 (i) Frame 7

(j) Frame 8 (k) Frame 9 (l) Frame 10

Figure 9.1: Map building as application



(a) Scene (b) Frame 1

(c) Frame 2 (d) Frame 3 (e) Frame 4

(f) Frame 5 (g) Frame 6 (h) Frame 7

Figure 9.2: Augmented reality as application: augmentation of a cube

10 Conclusion and outlook

This work demonstrates, that geometric 3D features can be combined successfully with 2D visual features. A multimodal system is presented, that combines a time of flight camera with an RGB webcam. Planes are recognized in the 3D point cloud and the RGB camera is used for tracking features on the texture of the planes. The pose is estimated by combining the normal vectors of the planes with the 3D point coordinates of the visual features. A comparison with the state of the art algorithms described in section 2.1 shows, that this algorithm outperforms comparable methods. It is faster than algorithms that process data offline while it is more accurate than algorithms that estimate poses online. With 2 to 3 frames per second it is suitable for online pose estimation.

To demonstrate this, two applications have been build which use the pose estimate of the proposed algorithm: 3D map generation in runtime and the augmentation of objects.

TOF sensors are too expensive for private use and thus are applied just in the field of industry or research. Nevertheless this technology is still very new and the devices will probably get less expensive in the future. Very recently, novel cheap 3D range cameras are conquering the market. One such sensor is the Kinect of the companies Microsoft and PrimeSense that is available since November 2010. Its working principle is light coding [3] and consist of a fixed attachment of a camera and a device that illuminates the scene with a known pattern.

The depth map of light coding sensors contains gaps, that occur from light rays that are not visible to the camera because they are occluded by objects. Due to this shadows it is not straightforward to apply algorithms, that have been developed for TOF cameras, to light coding sensors. Anyway, the proposed algorithm should be applicable also to light coding cameras because shadow regions do not contain planes and thus have no influence to the pose estimation.

List of Figures

List of Tables

Bibliography

[1] R. Adams and L. Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641 –647, jun 1994.

[2] Dror Aiger, Niloy J. Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. *ACM Trans. Graph.*, 27:85:1–85:10, August 2008.

[3] Chadi Albitar, P Graebling, and Christophe Doignon. Robust structured light coding for 3d reconstruction. *IEEE 11th International Conference on Computer Vision (2007)*, (August):1–6, 2007.

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

[5] Rodrigues R. C. B. and Pellegrino S. R. M. An experimental evaluation of algorithms for aerial image matching. *International Conference on theoretical, experimental and applied signal and image processing techniques*, 17:416–419, 2010.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

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

[8] Christian Braeuer-Burchardt. A simple new method for precise lens distortion correction of low cost camera systems. In Carl Rasmussen, Heinrich Buelthoff, Bernhard Schoelkopf, and Martin Giese, editors, *Pattern Recognition*, volume 3175 of *Lecture Notes in Computer Science*, pages 570–577. Springer Berlin / Heidelberg, 2004.

[9] B. Buttgen, F. Lustenberger, and P. Seitz. Demodulation pixel based on static drift fields. *Electron Devices, IEEE Transactions on*, 53(11):2741 –2747, nov. 2006.

[10] James Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Technical report, Stanford University, 2006.

[11] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Mach. Vision Appl.*, 9:272–290, March 1997.

[12] J.C.A. Fernandes, M.J.O. Ferreira, J.A.B.C. Neves, and C.A.C. Couto. Fast correction of lens distortion for image applications. In *Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on*, volume 2, pages 708 –712 vol.2, jul 1997.

[13] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*.

[14] E. R. Golder and J. G. Settle. The box-muller method for generating pseudo-random normal deviates. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 25(1):pp. 12–20, 1976.

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

[16] Berthold K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4:59–78, 1990. 10.1007/BF00137443.

[17] B Huhle, P Jenke, and Wolfgang Strasser. On-the-fly scene acquisition with a handy multi-sensor system. *International Journal of Intelligent Systems Technologies and Applications*, 5(3), 2008.

[18] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.

[19] Timothee Jost and Heinz Huegli. Fast icp algorithms for shape registration. In Luc Van Gool, editor, *Pattern Recognition*, volume 2449 of *Lecture Notes in Computer Science*, pages 91–99. Springer Berlin / Heidelberg, 2002.

[20] Luo Juan and O Gwun. A comparison of sift , pca-sift and surf. *International Journal of Image Processing IJIP*, 3(4):143–152, 2009.

[21] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004.

[22] John Mallon and Paul F. Whelan. Precise radial un-distortion of images. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 - Volume 01*, ICPR '04, pages 18–21, Washington, DC, USA, 2004. IEEE Computer Society.

[23] S. May, D. Droeschel, S. Fuchs, D. Holz, and A. Nuchter. Robust 3d-mapping with time-of-flight cameras. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1673 –1678, oct. 2009.

[24] S. May, D. Droeschel, S. Fuchs, D. Holz, and A. Nuchter. Robust 3d-mapping with time-of-flight cameras. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1673 –1678, oct. 2009.

[25] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[26] K. Ohno, T. Nomura, and S. Tadokoro. Real-time robot trajectory estimation and 3d map construction using 3d camera. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5279 –5285, oct. 2006.

[27] L. Papula. *Mathematik fuer Ingenieure und Naturwissenschaftler.* Band1. Vieweg, 2001.

[28] H. Rinne. *Taschenbuch der Statistik*. Wissenschaftlicher Verlag Harri Deutsch GmbH, 2008.

[29] Agnes Swadzba, Bing Liu, Jochen Penne, Oliver Jesorsky, and Ralf Kompe. A comprehensive system for 3d modeling from range images acquirered from a 3d tof sensor. In *International Conference on Computer Vision Systems*. University Library of Bielefeld, 2007.

[30] Christoffer Valgren and Achim J Lilienthal. Sift, surf and seasons: Appearance-based long-term localization in outdoor environments. *Robotics and Autonomous Systems*, 58(2):149–156, 2010.

[31] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330 – 1334, nov 2000.