TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG

The University of Resources. Since 1765.

Faculty of Mathematics and Computer Science
Institute for Computer Science
Professorship for Virtual Reality and Multimedia

Master Thesis

Post-Processing of Depth Images and Laser Scan Data for Feature-based Pose Estimation

Jonas Toth

Master Angewandte Informatik
Matriculation register: 57319

June 8, 2020

Tutor/First Proofreader:
Prof. Dr. Bernhard Jung

Second Proofreader:
M. Sc. Robert Lösch

Eidesstattliche Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen benutzt habe. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, habe ich in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Diese Versicherung bezieht sich auch auf die bildlichen Darstellungen.

8. Juni 2020 Jonas Toth

Declaration

I hereby declare that I completed this work without any improper help from a third party and without using any aids other than those cited. All ideas derived directly or indirectly from other sources are identified as such. This declaration also refers to the representation of figures and visual material.

June 8, 2020 Jonas Toth

Abstract

Features — special identifiable bits of data — and their recognition between different images play an important role in computer vision to solve tasks like visual odometry, place and object recognition or global localization. Methods for these areas are established and implemented for common mobile devices and robots. Even though depth data processing is omnipresent for robotic systems, feature detectors for salient, recognizable segments of such pointclouds are not an established method for reliable and robust localization without prior knowledge.

This work proposes depth image processing steps to utilize feature detectors and descriptors for color images on depth data. In order to apply the algorithms SIFT, SURF, ORB and AKAZE on depth images, they are converted into derived feature images that encode the local geometry of the measured environment. Multiple possible transformations of the depth data, from the already proposed Bearing-Angle image to measures of curvature and finally the novel Flexion image are developed, analyzed and evaluated with respect to, among other things, keypoint stability and discrimination potential of the descriptors. The aspects keypoint count, size, response, the matching distance between true and false positives and additional measures like precision, recall and informedness are determined for four experimental datasets. They consist of a synthetic scene, an underground mining environment, an office scene and LiDAR scans. In addition, the impact of filtering the depth data with edge-preserving filters is analyzed on the outcome.

All experiments indicate a consistently better performance of the Flexion image compared to the Bearing-Angle image and proof that SIFT and AKAZE are suitable as feature detectors and descriptors whereas SURF and ORB are not. This insight is underlined with the computation of a visual odometry benchmark. The research forms a solid foundation to further develop the use of classical computer vision algorithms on Flexion images to accomplish feature-related tasks with depth data.

Contents

Acronyms

AKAZE	Accelerated KAZE
BRIEF	Binary Robust Independent Elementary Features
CPD	Coherent Point Drift
DoF	Degrees-of-Freedom
DoG	Difference of Gaussian
FAST	Features from Accelerated Segment Test
fps	frames per second
GMM	Gaussian mixture model
GP-GPU	General Purposes GPU
ICP	Iterative Closest Points
KAZE	Japanese for wind
laser	light amplification by stimulated emission of radiation
LDB	Local Difference Binary
LED	Light Emitting Diode
LiDAR	Light Detection and Ranging
M-LDB	Modified-Local Difference Binary
NDT	Normal Distributions Transform
ORB	Oriented FAST and Rotated BRIEF
RANSAC	Random Sample Consensus
ROC	Receiver-Operating-Characteristics
ROS	Robotic Operating System
SIFT	Scale Invariant Feature Transform
SIMD	Single-Instruction-Multiple-Data
SLAM	Simultaneous Mapping and Localization
SURF	Speeded-Up Robust Features
ToF	Time-of-Flight

Glossary

Bearing-Angle The Bearing-Angle represents the angle between the light ray from the sensor to the depth value and the connection to a neighbouring depth value. The Bearing-Angle can be calculated with arbitrary neighbouring depth values. Both LiDAR scans and depth images can be converted using a proper intrinsic calibration of the sensors.

Bearing-Angle image The Bearing-Angle Image is a derived image type where each pixel encodes the Bearing-Angle for the light ray of this pixel. The angle is discretized from the range $(0, \pi)$ to the color depth of the target image. The resulting images are grayscale.

Curvature Curvature is a concept from differential geometry that applies to manifolds. The principal curvatures k_1 and k_2 measure how surfaces bend at that point. Different specific types of curvature do exist.

Feature Features are special points or elements of an image that can be recognized and distinguished from their surroundings. Corners and edges are simple features.

Flexion image The Flexion Image is a method to convert depth data. A scalar value is computed for each pixel of the depth image, that measures the angle between different estimations of the surface normal. This measure is useful for obtaining high-level image features from low-level geometry information.

Gaussian curvature Gaussian curvature is the product of k_1 and k_2 and is used in many fields. The Gaussian curvature characterizes points on a surface by its sign and magnitude. For negative values of the Gaussian curvature, the point is said to be hyperbolic, for positive values the point is an elliptic point. Points with Gaussian curvature being zero are parabolic points.

Mean curvature Mean curvature is the sum of k_1 and k_2 and one of the main curvature types.

Multi-Directional Bearing-Angle image This image is a feature image, similar to Bearing-Angle images. For each pixel, the sum of each Bearing-Angle in forward and backward direction is calculated. This angle is calculated in both diagonals, horizontal and vertical direction. Finally, the maximum value is used as pixel value after scaling.

Structure-from-Motion A processing pipeline that reconstructs world geometry from processing camera footage, extracting distance information in multiple processing steps. A final global optimization of backprojection errors can be done to improve the overall quality of the resulting model.

Symbol Definition

R	matrix
α	angles
$\mathbf{P}_{\mathcal{C}}$	camera coordinate
$\mathbf{P}_{\mathcal{I}}$	image coordinate
$\mathbf{P}_{\mathcal{P}}$	pixel coordinate
$\mathbf{P}_{\mathcal{S}}$	coordinate on unit sphere
$\mathbf{P}_{\mathcal{W}}$	world coordinate
$\mathbf{P}$	points
$\mathcal{F}$	pixels of feature images
$\vec{v}, \vec{V}$	vectors
a	scalars

List of Tables

List of Figures

1 Introduction

Visual odometry [1] and feature-based localization with optical sensors [2] are established technologies in robotics. Additionally, the use of depth sensors increased with their availability and fast integration into robotic systems. This naturally leads to the task of exploiting the depth information for problems like localization and mapping, which at their core require to align multiple pointclouds to each other. One dominant algorithm to solve this problem is ICP (Iterative Closest Points) [3]. It has various characteristics and limitations that require an initial estimate for the relative pose between pointclouds [4]. Loop closure in SLAM (Simultaneous Mapping and Localization) [5], global localization and place recognition [6] can not rely on such an initial estimate. Those problems are commonly solved with a feature-based approach, that detects distinctive points in different color images and finds correspondences of real world elements captured from multiple views. Detecting salient points in an image and describing the local neighbourhood in a recognizable manner is at the core of many solutions of computer vision related problems and a well researched topic [7].

The aim of this thesis is to transfer this technology from optical images to depth images and range data from LiDAR scans by proposing the novel Flexion image. Visual features require local brightness gradients that are not present in raw depth images. Therefore, a conversion of the depth image to a derived image encodes local geometrical relationships of neighbouring depth pixels and results in an image that feature detectors can work on — a feature image. Standard feature detectors and descriptors, namely SIFT, SURF, ORB and AKAZE, can then detect interesting geometrical shapes as normal keypoints. Their surroundings and real world context are captured with the keypoint descriptor. Scale and orientation detection are part of the keypoint detector.

The proposed feature images have different visual characteristics than color images. A feature-based depth data registration requires stable keypoints between multiple views and a discriminating descriptor between different geometries (Figure 1). Researching the properties of both the keypoints and the descriptors is therefore mandatory work before solving more complex tasks like global localization. This

Fig. 1: *Illustration of Feature Detection and Matching on Flexion images.* This figure demonstrates the processing steps from depth image (left) to features detected on a converted Flexion image (middle) that are matched between multiple views (right). Green points indicate correctly identified correspondences between consecutive images, undetected correspondences are purple and orange lines indicate incorrect matches.

work is part of developing robotic systems for underground mining environments. It is common to have detailed LiDAR scans of mine segments, which is a part of mine surveying. Bad lighting conditions challenge the classical optical systems and algorithms. Other techniques, like GPS, are outright impossible to use. A positive outcome of the proposed method can be the first step towards using LiDAR scans as map.

The main contribution of this thesis is a novel way to convert depth data into a derived feature image and the in-depth analysis of the performance of keypoint detectors and descriptors on feature images. All developed tools can be reused as both library code and executable binary to create such images and run feature algorithms on them. Both the qualitative and quantitative findings build an empirical foundation on developing this approach further for more complex feature-based tasks.

Section 2 introduces the related work on pointcloud registration and feature performance comparisons. Necessary foundational knowledge on depth sensors and the math of modeling their data is presented in Section 3. Additionally, it gives a high level introduction on keypoint detectors and descriptors and how their performance can be evaluated. Section 4 describes the novel depth data processing pipeline, starting with edge-preserving filtering followed by conversion to feature images and finally explaining the evaluated feature detection and description algorithms. The experiments evaluate this process. Section 5 describes the metrics, datasets and algorithm configurations. Section 6 presents and discusses the results. Finally, Section 7 concludes the work and proposes further research areas.

2 Related Work

Scaramuzza's work on calibrating the extrinsic of a LiDAR scanner to a camera [8] introduces the Bearing-Angle image. The conversion of range data to a derived visual representation format allows manual selection of corresponding corners, edges and other salient regions in both the color image and the Bearing-Angle image [8]. Minimizing the reprojection error for the selected points allows to compute a relative rotation and translation between both modalities. Lin et al. [9] apply the SURF [10] feature detector on Bearing-Angle images to reconstruct the relative pose between dense pointclouds. They additionally compare the performance of this registration to state-of-the-art ICP algorithms. They conclude, that the ICP has higher precision but execution speed is drastically improved when using the Bearing-Angle based pose as initial state. Zhuang et al. [11] apply SIFT [12] on Bearing-Angle images for scene recognition on LiDAR scans. To the best of the author's knowledge, no other work includes the use of Bearing-Angle images.

2.1 Pointcloud Registration

Range sensors produce a set of points and one natural task is to determine the relative transformation between two such sets — pointcloud registration. This task has many applications in photogrammetry, robotics and engineering.

ICP, originally formulated by Besl [3], is a robust and easy to implement algorithm to achieve this registration. At its core is the assignment of point correspondences between both point sets, calculating a relative pose that results in the lowest distance between the chosen points and evaluating the disparity of both pointclouds. The mismatch of the pose is iteratively reduced by finding better correspondences using the distance between the points of the pointclouds as error metric. This basic approach received multiple improvements with generalized ICP [13, 14] providing a fully probabilistic formulation. ICP provides a 6 Degrees-of-Freedom (DoF) registration consisting of a rotation and translation. Convergence or an upper bound of the error is not guaranteed and in general relies on a good initial estimate of the relative

pose of both pointclouds, rendering it useless for global registration problems without prior knowledge. Failure to converge on a relative pose can happen both due to a bad initial pose or not corresponding pointclouds.

A different approach to pointcloud registration is provided by the Normal Distributions Transform (NDT) algorithm proposed by Biber and Straßer [15]. It works without explicit correspondences between both pointclouds. First, the pointcloud is subdivided into cells. Each cell approximates a normal distribution of the probability of measuring a point at a position within the cell. Registering a second pointcloud is a matter of maximizing the likelihood that the pointcloud is measured in the reference pointcloud under a given pose. The normal distributions are differentiable and classical numerical algorithms, like Newton's algorithm, can be used for the optimization.

Myronenko and Song [16] proposed the Coherent Point Drift (CPD) algorithm. Similar to NDT, the algorithm utilizes probabilistic methods to achieve the registration. A pointcloud is modelled with a Gaussian mixture model (GMM) defined through multiple centroids and their variances. Again, alignment of two pointclouds means the maximal probability of the data points under the GMM. In iterative refinement steps of the relative pose, the centroids of the GMM are enforced to move in topological consistent fashion, hence coherent point drift. The review articles [17, 18] introduce state-of-the-art pointcloud registration algorithms both in general and for robotic applications. All these methods for pointcloud registration are not suitable to search for a matching segment of a pointcloud or to even select a suitable pointcloud from a database. For such a task features come into play because they provide a tool to discard unrelated pointclouds fast.

Elbaz et al. [19] extract subsets of a pointcloud, analyze its main directional components using a principal component analysis and synthesize a depth map for these patches. A deep neural network based auto-encoder computes a low-dimensional descriptor for each patch. Registration of pointclouds is achieved by matching these descriptors followed by a conventional fine-tuning step.

Steder et al. [20] proposed an interest point detector for pointclouds. The image's curvature is analyzed by first smoothing with a Gaussian and then computing the

second derivatives of the range data. High curvature is used for saliency, but special points on edges and regions of occlusion are filtered out first. Interest points must succeed an empirical found saliency threshold to be selected. The feature descriptors for pointclouds lack the desired functionality and robustness. Image based features are superior in terms of experience in the field, availability and success in a diverse set of use cases [21–23].

2.2 Feature Algorithm Comparison

Keypoint detectors and feature descriptors provide a way to detect salient regions of a gray-scale image that can be consistently detected between different views of a scene. The breakthrough development is Lowe's SIFT [12] algorithm. The early benchmark [24] of SIFT to other classical algorithms, like the Harris corner detector [25], demonstrated the strong performance of SIFT. Since the early 2000s more work on improving upon SIFT in terms of performance, compute requirements and diverse applications lead to more competitors like SURF [10], ORB [26] and AKAZE [27]. A recent comparison between those modern, well established algorithms by Andersson and Reyna Marquez [7] still puts SIFT as a top-performer, with AKAZE yielding similar but slightly less accurate results. ORB [26] on the other hand is less accurate but due to its lower computational cost still a viable option, especially for mobile applications [7]. The use of classical feature descriptors and template based matching for different modalities, like thermal imaging, is evaluated by Gesto-Diaz et al. [28]. They conclude that the detectors result in many correct matches within on modality. Correct inter-modal matching on the other hand is harder and depends on the modalities used. They note that combining thermal and depth images yield the worst performance regardless of the used registration technique. The result is encouraging that algorithms like SIFT [12] are not bound to color images but can be used in scenarios with vastly different appearance. It also motivates the search for better ways to process depth data for feature-based matching.

3 Fundamentals

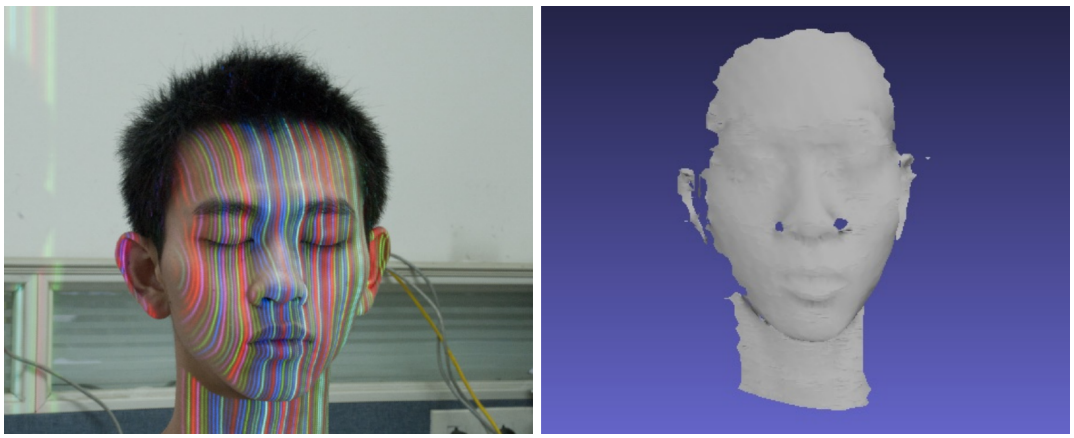
This section introduces the relevant principles of depth sensing, depth data representation, image formation and provides the basics for feature matching evaluation.

3.1 Depth Sensors

Depth sensors play an increasingly important role in modern robotic applications. They give robots a fast mean to detect obstacles and locate themselves in their environments. The following paragraphs describe the underlying sensing principles for the depth sensors commonly used in mobile robotic applications.

3.1.1 Structured-Light Depth Sensors

Structured-light depth sensors project a known light pattern into space and optically sense the reflection [29]. The pattern appears distorted from a viewpoint different than the projector due to the shape of the reflecting object. Figure 2 provides an example for structured-light sensing using visible light. Stripes and grids of lines or points are common for sensors in robotic applications [29], such as the Kinectv1 [30] and Intel RealSense [31] sensors.



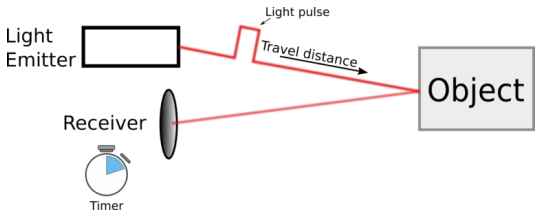
(a) Visible structured light pattern **(b)** Reconstructed result

Fig. 2: *Demonstration of structured-light depth sensors.* One way to reconstruct depth information are structured light depth sensors. A predefined light pattern is projected into space. As the photons are reflected from obstacles at different distances relative to the observer, the light pattern appears transformed. This deformation is used to reconstruct the geometry of the objects. The light emitted is usually not visible to humans as infrared light sources and cameras are used often. The images are taken from [32].

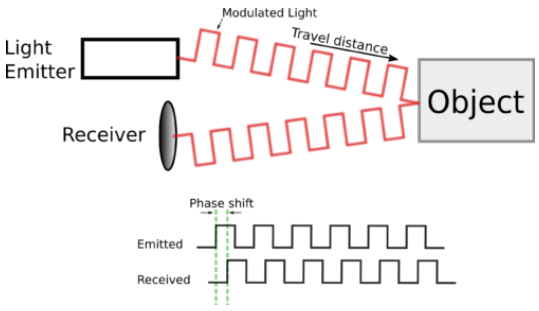
Viewing the projected patterns from multiple viewpoints allows triangulation to recover the depth information. Mobile robotic depth sensors achieve frame rates of 30 fps or more. The projected pattern should not interfere with other optic devices leading to the usage of infrared light.

3.1.2 Time-of-Flight (ToF) cameras

The second prevalent sensing technology measures the traveling time of light [33, p. 27-41]. Multiplying the measured round-trip time by the known speed of light in the surrounding medium results in twice the distance of the object. Such a sensor system requires a light sources. Unlike the specialized patterns emitted for structured light, this can be an ordinary infrared LED, again preventing interference in other spectra of light. A pulsed and synchronized light emitter, as exemplified in Figure 3a, allows direct measurement of the round-trip time of light. Such a system requires exact synchronisation and a high temporal resolution to achieve accurate measurements. The different and more recent development in depth sensing technology is measuring the phase shift (Figure 3b) of the reflected light to recover the traveling time of the light wave. This is the underlying principle of the Kinectv2 depth sensor [30]. The drawback of this technology is the limited range of the depth sensor, due to the ambiguity of the wave signal. Using modulation techniques to create a wave signal with more than one frequency increases the range of the sensor [33, p. 27-41].



(a) One way to measure the time-of-flight for photons is to emit a light pulse and measure the round-trip time until the reception of the reflection.



(b) The second, more recent approach is measuring the phase-shift of photons relative to the light source. This allows the usage of a continuous light source.

Fig. 3: *Illustration of the measuring principles for ToF cameras.* Time-of-Flight is first of all a measurement principle for distances used in different contexts. In this specific case ToF sensors measure the round-trip time of light from emission, reflection and sensing. Multiplying the measured time by the speed of light results in the distance the light traveled. The illustrations are adopted from [34].

3.1.3 Light Detection and Ranging (LiDAR)

Light Detection and Ranging (LiDAR) uses the same measurement principle as ToF cameras [35, p. 239]. It determines the distance of an object using round-trip time measurements. The difference between LiDAR and ToF cameras is the usage of a laser as light source and the overall setup of the sensor system. The most common configurations use a horizontally rotating mirror that reflects the beam into the room. For vertical coverage, the measurement head is rotated in total. Each complete horizontal turn is called a scan line.

Some LiDARs, especially mobile systems with real time requirements, measure tens of scan lines. Such systems achieve a higher frame rate. A dense LiDAR scan is required for the optical feature-based registration approach. Such scans can be created with terrestrial LiDAR stations, common in geodesic applications. Additionally, the intensity of the reflected light beam might be obtained as second channel.

Flash LiDAR is a scanner-less type of LiDAR that does not require a rotating mirror or other mechanical system [36]. A flash LiDAR uses the measurement principles described for ToF cameras and measures a dense depth image in one shot.

3.2 Coordinate Systems

This section introduces the used naming conventions for coordinate systems that model the sensory systems used in computer vision. Homogeneous coordinates allow efficient computation of affine transformations with one matrix-vector multiplication and knowledge of them is a prerequisite to fully understand the pinhole camera model in Section 3.3.1. Corke [37, p. 15-39,533] provides excellent coverage of these topics.

Naming Conventions

A point in space is referenced by a vector in that space. The global reference frame is arbitrarily chosen in *world coordinates*. Within that world coordinate system exist one or more sensors spanning their own coordinate system. Referencing the point relative to such a sensor yields *camera coordinates*. The sensor itself performs some

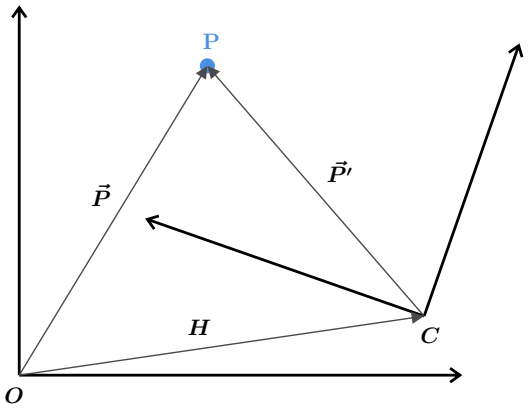


Fig. 4: *Coordinate transformation.* Relationship between different representation of the same point in different coordinate systems. The relationship displayed here and the corresponding linear algebra holds true for Cartesian coordinate systems of any dimension. Figure based on [37, p.20].

form of dimension reducing projection of the three dimensional point to a two dimensional point, an *image coordinate* [37, p. 251-261]. *Image coordinates* are an intermediate representation before discretising the point to individual pixels, called *pixel coordinates*. Inverting this projection results in *spherical coordinates*, camera coordinates with unit length that just encode the direction of a light ray that hits a pixel. Different sensor models describe these projecting transformations mathematically. The relative transformation between *camera coordinates* and *world coordinates*, the sensors *pose*, can change over time. Multiple sensors can exist within the *world coordinate system*.

Coordinate Transformations and Pose

Let $\mathbf{O}_\mathcal{W}$ be the origin of the *world coordinate system*. A three dimensional vector $\vec{P_\mathcal{W}} = \begin{pmatrix} U & V & W \end{pmatrix}^T$ represents a point $\mathbf{P}_\mathcal{W}$ in this room relative to the origin $\mathbf{O}_\mathcal{W}$. Seeing this point $\mathbf{P}_\mathcal{W}$ from a camera $\mathbf{C}_\mathcal{W}$ within the same room is equivalent to transforming the coordinate system from *world coordinates* to *camera coordinates* by a change of basis, as demonstrated in Figure 4. The relative rotation and translation of the camera $\mathbf{C}_\mathcal{W}$ to the room is also called its *pose* [37, p.15-39]. It is parameterized by a rotation matrix R and a translation vector $\vec{t}$. Homogeneous coordinates provide a computational more efficient way to apply pose transformations to vectors with a single 4 dimensional matrix-vector multiplication (Equation 1) instead of

a three dimensional matrix-vector multiplication and an additional 3×1 vector addition. The point $\mathbf{P}_{\mathcal{W}}$ seen from the camera $\mathbf{C}_{\mathcal{W}}$ is computed with the homogeneous transformation matrix $H = \begin{bmatrix} & R & & \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{1}$$

$$\vec{P_{\mathcal{C}}} = H\vec{P_{\mathcal{W}}}$$

3.3 Image Formation and Camera Models

Each depth sensor measurement returns a matrix with depth or range values. The projection model of the sensor performs the calculation of the camera coordinates for each point. Additionally, sensors usually experience some form of imperfection in the imaging process, like lens distortion or misalignment of components. These imperfections are corrected with additional models, like the distortion model for pinhole cameras.

Both calibration and error correction are out of the scope of this thesis and sensor input is expected to adhere to the basic models. This might require preprocessing of the raw sensor data. On some robotic platforms, like ROS (Robotic Operating System), this is done automatically when providing a matching calibration of the sensor.

3.3.1 Pinhole Camera Model

The pinhole camera model is used for both classical and depth cameras. Figure 5 shows the perspective projection of three-dimensional points onto the plane [39, p. 25-35]. This projection is not depth preserving. The model is parameterized by multiple parameters. The focal lengths $f_x > 0$ and $f_y > 0$ correspond to the distance

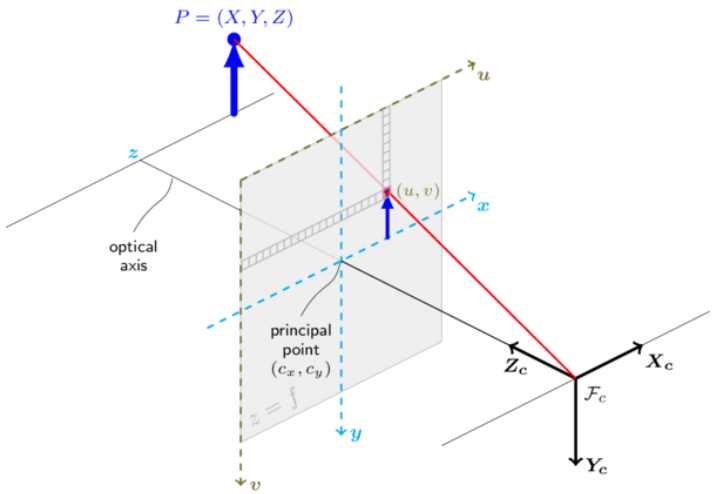


Fig. 5: *Pinhole camera model.* The pinhole camera model performs the perspective projection of three dimensional points into the plane. Real cameras additionally have a lens system that can result in radial and tangential distortion. These effects must be compensated with a proper calibration of distortion models. The camera-coordinate-to-pixel-coordinate projection loses information about the depth. Therefore, the back projection results only in the direction of the light ray. The depth sensor gives the additional distance information to recover the point in camera coordinates. Graphic adopted from [38].

of the idealized pinhole to the image plane. The skew s, that resembles non-quadratic pixel shape and is most of the time $s = 0$ and the image center $c_x > 0$ and $c_y > 0$.

Forward Projection

Let $\vec{P_{\mathcal{W}}} = \begin{pmatrix} X & Y & Z \end{pmatrix}^T$, $Z \neq 0$ be a world point in camera coordinates. The perspective projection

$$\vec{P_{\mathcal{I}}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{2}$$

results in the point $\vec{P_{\mathcal{W}}}$ in image coordinates $\vec{P_{\mathcal{I}}}$. The image coordinates could be distorted and any correction would happen at this stage. Multiplication of the 3×3 upper triangular camera matrix K and the homogeneous image coordinates results in the final pixel coordinates:

$$K\vec{P_{\mathcal{I}}} = \begin{pmatrix} f_x & f_x s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \vec{P_{\mathcal{P}}}. \tag{3}$$

Backward Projection

Applying the inverse transformations allows to recover the direction of the light ray matching pixel $\vec{P_{\mathcal{P}}}$. The potentially distorted image coordinates $\vec{P_{\mathcal{I}}}$ are obtained with the following formula:

$$\begin{aligned} \vec{P_{\mathcal{I}}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} &= K^{-1}\vec{P_{\mathcal{P}}} \\ \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} \frac{1}{f_x} & -\frac{s}{f_y} \\ 0 & \frac{1}{f_y} \end{pmatrix} \begin{pmatrix} u - c_x \\ v - c_y \end{pmatrix}. \end{aligned} \tag{4}$$

Again, if a distortion model is used, the inverse transformation needs to be applied to $\vec{P_{\mathcal{I}}}$. The final step is the projection to the unit sphere, reconstructing the direction of the idealized light ray, that hit the pixel. This is equivalent to norming the vector $\vec{P_{\mathcal{I}}}$.

$$P_{\mathcal{S}} = \begin{pmatrix} Xs \\ Ys \\ Zs \end{pmatrix} = \frac{1}{\sqrt{x^2 + y^2 + 1}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{5}$$

3.3.2 Equirectangular Camera Model

The equirectangular camera model maps the whole unit sphere to the plane [40, p. 90]. Equirectangular images have constant, but potentially different angular resolution in x- and y-direction. This mapping stores panoramas that are usually stitched together from multiple cameras. The LiDAR sensor output can be stored as an equirectangular image as well. Supporting this camera model gives flexibility to support all kinds of sensors. Multiple pinhole sensors can be stitched to one virtual sensor and mapped to the equirectangular model. The projections are mostly a conversion from Cartesian to spherical coordinates and described here in brief. Additionally, to suite the LiDAR sensor better, a slight addition for a vertical field of view is added. This field of view simply crops the full equirectangular image to the predefined slice and

discards other coordinates as invalid. Figure 6 shows the relationship between points in spherical coordinates and the equirectangular projection.

The model is parameterized by the width $w > 0$ and height $h > 0$ of the image and the vertical field of view $[\theta_{min}, \theta_{max}],\ \theta_{min} < \theta_{max}$ which defaults to $\theta_{min} = 0$ and $\theta_{max} = \pi$. This definition yields the vertical and horizontal angular resolution of a pixel, which is a constant:

$$d\varphi = \frac{2\pi}{w}, \quad d\theta \ = \frac{\theta_{max} - \theta_{min}}{h}. \tag{6}$$

The used spherical coordinates $z = (r, \varphi, \theta)$ convention defines the radius $r \geq 0$, the azimuthal angle $\varphi \in [-\pi, \pi)$ and the polar angle $\theta \in [0, \pi]$.

Forward Projection

Let $\vec{P_C} = \begin{pmatrix} X & Y & Z \end{pmatrix}^T$ be a point in camera coordinates. This point is converted to spherical coordinates:

$$\begin{aligned} r &= \sqrt{X^2 + Y^2 + Z^2} \\ \theta &= \arccos \frac{Z}{r} \\ \varphi &= \arctan2 \frac{Y}{X}. \end{aligned} \tag{7}$$

This conversion is ill defined for $r = 0$ which corresponds to a missing range measurement and can be skipped safely. Finally, the spherical coordinates are converted

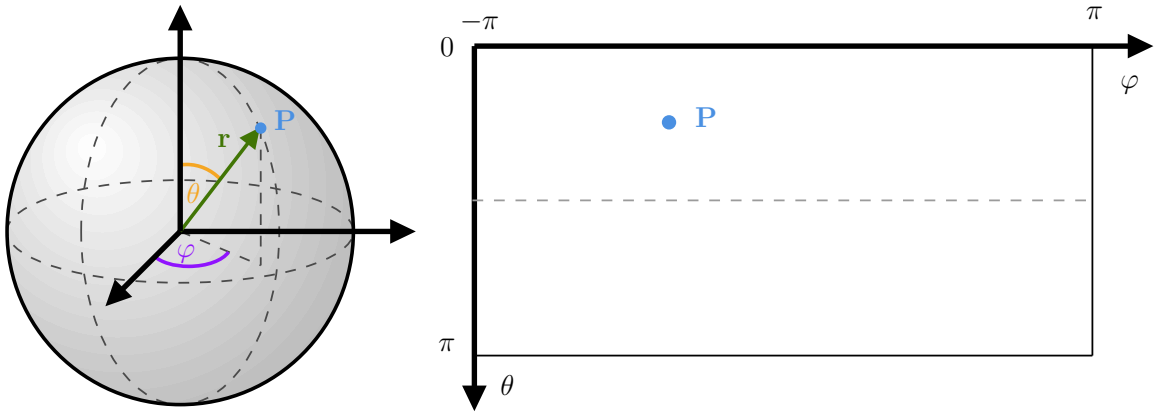


Fig. 6: *Spherical camera model and equirectangular images.* Each Cartesian point that is not the origin can be uniquely converted to spherical coordinates. The origin $\mathbf{O}_C$ is interpreted as an invalid measurement in the context of this thesis and therefore filtered out during processing.

to pixel coordinates with:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{\varphi+\pi}{d\varphi} \\ \frac{\theta}{d\theta} \end{pmatrix}. \tag{8}$$

Backward Projection

The backward projection recovers the light ray direction from pixel coordinates. It is analogous to the forward projection mainly a conversion from spherical to Cartesian coordinates. In general, $r = 1$ as the resulting vector is of unit length:

$$\begin{gathered} \varphi = ud\varphi - \pi \\ \theta = \theta_{min} + vd\theta \\ \begin{pmatrix} X_s \\ Y_s \\ Z_s \end{pmatrix} = \begin{pmatrix} \sin\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\theta \end{pmatrix}. \end{gathered} \tag{9}$$

3.3.3 Range and Depth Conversion

There are two possible conventions on how to store depth data, both demonstrated in Figure 7. The orthographic depth d is the distance from the image plane and commonly returned by pinhole depth sensors like the Kinectv2 [30]. On the other hand, the range r is the Euclidean norm of the vector from the camera origin to $\vec{P_C}$. For consistency and generality, depth data is converted to range data before further processing takes place. This simplifies later conversion and mixing of different camera models in a coherent fashion. The conversion depends on the camera model and only the pinhole model is subject to analysis here, because LiDAR scanner return range values.

From the forward projection function of the pinhole model (Equation 2) follows the connection of the range and orthographic depth of the camera coordinates $\vec{P_C}$ and image coordinates $\vec{P_I}$ of the point $\mathbf{P}$ (Equation 10).

$$\begin{aligned} \vec{P_I} &= \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \quad Z = d \\ \implies Z\vec{P_I} &= \vec{P_C} \implies Z\|\vec{P_I}\|_2 = \|\vec{P_C}\|_2 \\ \iff r &= Z\|\vec{P_I}\|_2 \end{aligned} \tag{10}$$

The inverse of the Z_s component of the backward projection from Equation 5 provides the scaling factor between orthographic depth and range. Consequently, the range r for the depth d at image coordinates $\begin{pmatrix} x & y \end{pmatrix}^T$ with corresponding coordinates on the unit sphere $\begin{pmatrix} X_s & Y_s & Z_s \end{pmatrix}^T$ is computed with:

$$r = \frac{d}{Z_s} = d\sqrt{x^2 + y^2 + 1}. \tag{11}$$

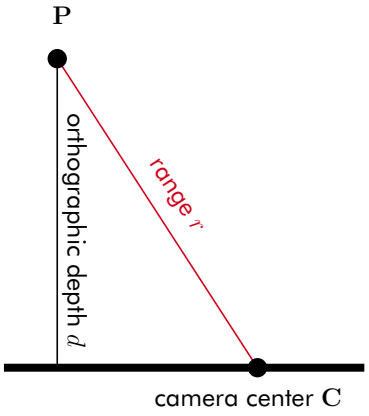


Fig. 7: *Orthographic depth and range visualized.* The distance information of a point relative to the center of a sensor at $\vec{C_C}$ can either be stored as the orthographic depth, mathematically the Z component of the camera coordinates or the range, equivalent to the Euclidean norm of the point $\vec{P_C}$ in camera coordinates. The orthographic depth is commonly provided by depth cameras and the range by LiDAR systems.

3.4 Keypoint Detection and Feature Description

Extracting information from images is a common task in computer vision. One approach is the detection of salient regions of special interest — so called *features*. Such regions can have arbitrary detectable properties, like a sharp brightness gradient or being corner-like. Detecting such a region results in a *keypoint* that might have properties like a *size*, *response* and sometimes even an *orientation* (Figure 8). Keypoints on itself are not descriptive enough to be identified in different images. A changing view point or even a different camera result in changes to the overall shape and appearance of the scene. Reliable keypoint identification requires the analysis of bigger image patches. The *keypoint descriptor* analyzes a predefined region around the centered keypoint and extracts a vector for that region. The extracted descriptor has a high dimension with tens to hundreds of elements that for example store local brightness gradients.

Connecting the keypoints between different images uses the extracted descriptors for each keypoint. Similar descriptors are associated. Similarity of the descriptors is calculated with a matching norm dependent on the structure of the descriptor. Dense real descriptors commonly use the Euclidean norm whereas binary descriptors use the Hamming norm. Additional constraints, like geometric consistency, improve the

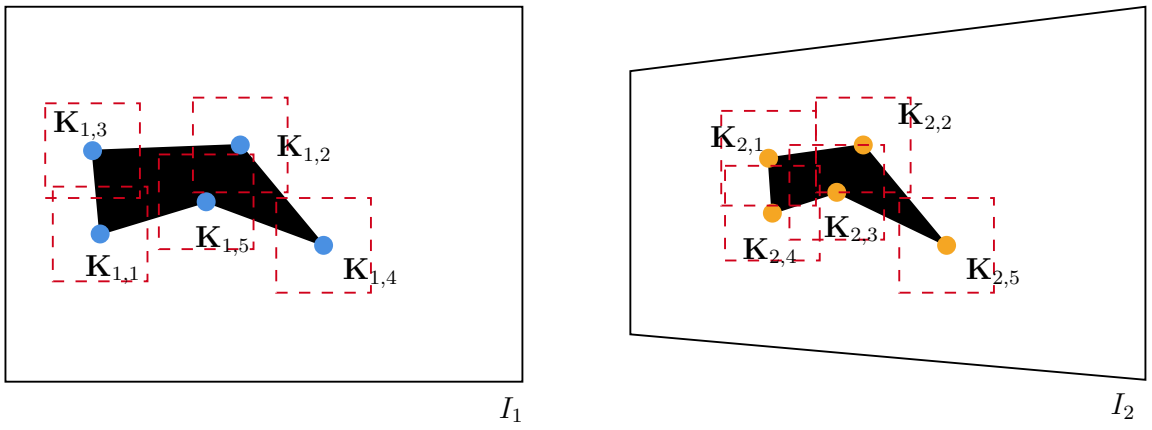


Fig. 8: *Schematic feature detection and matching.* This figure displays two polygons from different view points. The feature detection algorithm detects the corners of the polygon as keypoints in both images. The order of detection is not stable and establishing the correspondence between the detected keypoints requires further processing. A feature descriptor examines to local neighbourhood of each keypoint, visualized with the red dotted square. This descriptor is then used to find the most similar region around a keypoint of the other image. This process is called *feature matching*.

accuracy of the matching result. For the provided example a potential matching result might be

$$\begin{aligned} &\mathbf{K_{1,2}} \leftrightarrow \mathbf{K_{2,2}} \bullet \quad \mathbf{K_{1,4}} \leftrightarrow \mathbf{K_{2,5}} \bullet \quad \mathbf{K_{1,5}} \leftrightarrow \mathbf{K_{2,3}} \bullet \\ &\mathbf{K_{1,3}} \leftrightarrow \mathbf{K_{2,4}} \qquad \mathbf{K_{1,1}} \leftrightarrow \mathbf{K_{2,1}} \end{aligned} \tag{12}$$

whereby green bullets indicate a correct match. An important descriptor design criteria is the trade-off between computational cost and descriptive power, as especially descriptor matching is computationally expensive.

Features play a key role in multiple computer vision tasks. Place or object recognition and visual vocabularies rely on the identification of similar descriptors in queried image to already indexed descriptors of known images. Typically, storage and computational constrains require preprocessing the descriptors with clustering techniques to reduce the number of descriptors to store and allow hierarchical matching.

The second major application is visual odometry and Structure-from-Motion. Corresponding keypoints provide landmarks for triangulation and pose reconstruction. The relative pose between the two camera is computed with Nistér's 5-point algorithm [41]. To obtain a stable and consistent result for the pose the RANSAC (Random Sample Consensus) [42] algorithm performs multiple pose computations with random subsets of keypoint correspondences. The assumed pose is used to project the keypoints from one image into the other. A high pixel distance for the projected keypoints to the expected keypoints indicates a wrong pose, showing that the used subset of keypoints for the pose calculation had false matches. After multiple iterations, a consensus pose might be obtained and can be optimized as final pose. If no such consensus pose exists, the image pair is rejected.

Establishing the correspondence between keypoints of multiple images can be supported by using additional heuristics for descriptor matching. The first heuristic is *cross checking*. Descriptors correspond only if the distance of the descriptors is minimal in both directions. The second commonly used heuristic is *Lowe's ratio check* [12] putting the best and second best descriptor match distance into relation and requiring a certain distance ratio to be superseded. The last simple heuristic is the definition of an *upper bound* for the match distance. Improvements to RANSAC [43, 44] use

more sophisticated keypoint sampling criteria by exploiting spatial consistency or the descriptor distance.

More complicated keypoint preprocessing can be done to reduce the number of descriptors to match. Keypoints can be discarded to achieve a higher spread over the image and avoid clusters of keypoints that provoke descriptor ambiguity. The detector response is another important criteria for sorting keypoints by quality. If a priori information of the approximate motion of the camera is known, matches can be ranked based on the expected displacement of the keypoint.

3.5 Statistics of Binary Classifiers

Evaluating the performance of the feature matching algorithms on the converted depth images is done by analyzing the matching process as a binary classifier. The classification task is to determine if a keypoint $\mathbf{K_1}$ of image I_1 corresponds with keypoint $\mathbf{K_2}$ in image I_2 with a binary outcome. A pair of keypoints corresponds, if both pixel coordinates projected into space intersect at the same point, allowing for a little bit of uncertainty, e.g. 2 px backprojection difference. It is necessary to consider the distribution of a quantity and analyze it from multiple view points. Therefore, the final evaluation will consider other criteria, like distribution and characteristics of the keypoints.

Pairs of keypoints are defined as *positive* (P) if they correspond and *negative* (N) otherwise. The predicted outcome of the matching system is either *yes* (y) or *no* (n). The result of this prediction is in one of four categories. A correspondence can be correctly recognized, a *true positive* (TP), or mistakenly recognized, a *false positive* (FP). Similarly, the lack of correspondence can be either correctly or incorrectly detected (*true negatives* (TN) and *false negatives* (FN), respectively). The *confusion matrix*, a specialized case of a *contingency table* [45, p.21], keeps track of classification outcomes and is the basis for many performance criteria and derived metrics. Table 1 demonstrates the arrangement of elements in a confusion matrix. The provided values can be either an absolute count of elements or a relative share of total elements and must be exhaustive.

	P	N	
y	TP	FP	Total *yes*
n	FN	TN	Total *no*
	Total *Positives* ($\#P$)	Total *Negatives* ($\#N$)	

Tab. 1: *Definition of the confusion matrix.* A confusion matrix summarizes the decision quality of a binary classifier. The elements to be classified can be in either category; *positive* (P) or *negative* (N). Correct classifications of elements are either *true positives* (TP) or *true negatives* (TN). Misclassification results in *false positives* (FP) or *false negatives* (FN).

All *positive* elements are *relevant elements* in the sense, that the keypoint pair points to the same world coordinate. Predicted correspondences (y) are called *selected elements*. Different common ratios are derived from the confusion matrix [46].

True Positive Rate The true positive rate defines the ratio of correctly identified positive elements to total positive elements and is also called **hit rate**, **recall** or **sensitivity**.

$$TPR = \frac{TP}{\#P} \tag{13}$$

False Positive Rate This ratio shows how many negative elements are mistakenly classified as positive elements. It is sometimes named **false alarm rate** or **fallout**.

$$FPR = \frac{FP}{\#N} \tag{14}$$

True Negative Rate Also called **specificity**, the true negative rate is the ratio of correctly rejected elements.

$$\begin{aligned} TNR &= \frac{TN}{\#N} \\ &= 1 - FPR \end{aligned} \tag{15}$$

Accuracy The accuracy is the percentage of correctly classified elements. This measure is called **Rand index** in data clustering.

$$A = \frac{TP + TN}{\#P + \#N} = \frac{TP + TN}{TP + FP + TN + FN} \tag{16}$$

Precision The precision is the ratio of true positive elements in all selected elements.

$$precision = \frac{TP}{TP + FP} \tag{17}$$

Each of these ratios can be of any value between $[0, 1]$ and might be written as a percentage. Both precision and recall are commonly used to compare and evaluate feature matching algorithms and keypoint descriptors. The proposed metric share the pitfall, that a low number of total elements or highly skewed data can result in very strong results for each metric, but correspond to a weak real world performance.

Receiver Operating Statistics Analysis

Choosing a suitable configuration or algorithm combination requires trade-offs between different aspects of the final system. Receiver-Operating-Characteristics (ROC) analysis [46] provides a tool to visualize the trade-offs between different systems. The central component of this analysis are the *ROC graphs*. They plot the false positive rate and true positive rate, the *ROC space*, of multiple classifiers in one graph, with Figure 9 containing all qualitatively different cases.

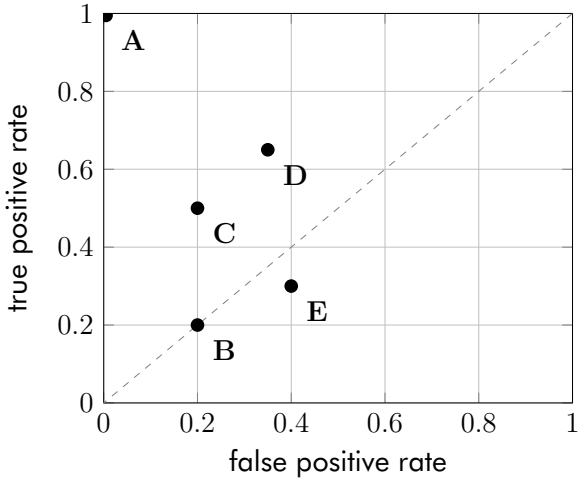


Fig. 9: *The elements of a ROC graph.* A ROC graph plots the true positive rate versus the false positive rate of a binary classifier. Each point is one classification system. The diagonal $f(x) = x$ is equivalent to a 50% chance of correct classification and a system on this line has no predictive power, as in this case $\mathbf{B}$. Points above this line ($\mathbf{A}$, $\mathbf{C}$, $\mathbf{D}$) indicate better-than-random performance and the bigger the distance to this diagonal the higher the informedness, also called Youden's index, of this system. The optimal classifier ($\mathbf{A}$) is in the top-left corner and has no false decisions. Points below the diagonal ($\mathbf{E}$) might still have predictive power but indicate a wrong labeling of the outcomes.

Points above the diagonal have predictive power proportional to the distance to the diagonal. This distance is also called the **Youden's index** $\mathbf{J}$ [47] and is a measure of informedness of the classifier:

$$J = sensitivity + specificity - 1 \quad = \frac{TP}{TP + FN} + \frac{TN}{TN + FP} - 1. \tag{18}$$

Unless there are no samples tested, Youden's index is $\mathbf{J} \in [0, 1]$. The higher $\mathbf{J}$ the better the two groups are separated by the classifier. ROC analysis gives a valuable instrument to compare the relative performance of classifiers. A final judgement requires to take additional measures like absolute number of classified elements into account. Fawcett's [46] work provides a more detailed introduction to ROC analysis in machine learning.

4 Depth Image Processing Pipeline

This section introduces the proposed steps to convert depth images to feature images. Depth images are affected by different error sources and intrinsic noise of the sensory system. The novel feature image are sensitive to these errors and edge-preserving filtering, covered first, mitigates their effects. Following this, multiple conversion strategies are shown that transform depth data to feature images. Additional remarks on their implementation are provided. The four selected feature detectors, SIFT, SURF, ORB and AKAZE, and their design is explained subsequently, covering all processing steps from the raw depth images to extracted features.

4.1 Edge-Preserving Filtering

Filtering is an operation to reduce the impact of sensor noise from limited resolution and other random effects on measured quantities. These deviations propagate through the feature image conversion. The feature images, introduced in Section 4.2, build upon information information about changes in the geometry of sensed objects. Therefore, sharp changes, e.g. edges and corners, shall be preserved by the filter, requiring edge-preserving filters. Each tested filter uses OpenCV's [48] proven and readily available implementation.

4.1.1 Median Blur

Median Blur, introduced to image processing by Frieden [49], is a well established technique to reduce salt-and-pepper noise and is an effective edge-preserving filter. The filter modifies each pixel of the image equally. A window of $n \times m$ pixels with $n, m \in \mathbb{N}_{2k+1}$ is centered at the pixel. The median of all pixels in that window is calculated and stored as the new value. Floating point operations are not required and the filter is implementable with $\mathcal{O}(n)$ [50] time complexity. n and m are the controllable parameters. OpenCV's [48] implementation additionally utilizes SIMD (Single-Instruction-Multiple-Data) instructions for increased processing speed.

4.1.2 Bilateral Filter

Bilateral filtering, introduced by Tomasi and Manduchi [51], considers two factors in the filtering process. *Spatial closeness*, similar to median filtering, and *similarity* of the values are combined in the bilateral filter. Its application is convolutional, similar to the median blur and other filters. Each neighbouring pixel's effect on the central pixel is weighted by a geometric *closeness* and *similarity* function. Similarity can be based on a classical distance norm of photometric values, like brightness, or be more sophisticated, like perceived similarity. A common function for both aspects is a Gaussian kernel with the Euclidean norm as distance measure. Figure 10 provides an example application of the bilateral filter on a single channel, two-dimensional signal with the Gaussian function for *closeness* and *similarity*. The filter is dependant on the functions for *similarity* and *closeness* and the available parameters are defined through these functions. The Gaussian case is controlled by the parameters σ_d, the standard deviation of *closeness* and σ_r, the standard deviation of *similarity*. The mathematical formulation — based on the original publication [51] — for the *closeness* c in the Gaussian case of the bilateral filter is:

$$\begin{aligned} c(\xi, \vec{x}) &= \exp\left(-\frac{1}{2}\left(\frac{d(\xi, \vec{x})}{\sigma_d}\right)^2\right) \\ d(\xi, \vec{x}) &= d(\xi - \vec{x}) = \|\xi - \vec{x}\|_2. \end{aligned} \tag{19}$$

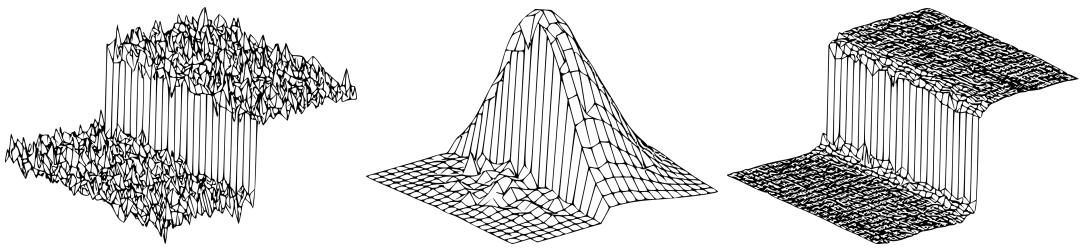


Fig. 10: *Bilateral filtering visualized.* The figures show how the bilateral filter works on a single channel step function. Figure 10a shows the original signal with random noise added. The weighting of its neighbouring pixels is computed for a pixel in the center. The weights are visualized in Figure 10b. The lower values of the step are neglected due to their lack of similarity regardless of closeness. Figure 10c shows the result of the full convolution of the filter, yielding a smoothed signal without blurred edge. Images adopted from [51].

This weighs any point's ξ relevance from the input signal to the filtered point $\vec{x}$ based on it's distance. The definition of the *similarity* s follows the same principle:

$$\begin{aligned} s(\xi, \vec{x}) &= \exp\left(-\frac{1}{2}\left(\frac{\theta(\mathbf{f}(\xi), \mathbf{f}(\vec{x}))}{\sigma_r}\right)^2\right) \\ \theta(\phi, \mathbf{f}) &= \theta(\phi - \mathbf{f}) = \|\phi - \mathbf{f}\|_2. \end{aligned} \tag{20}$$

The function $\mathbf{f}(\vec{x})$ is the value of the signal at the position $\vec{x}$ and $\theta(\phi, \mathbf{f})$ weighs these values based on their Euclidean distance. To compute the filtered value on any point $\vec{x}$, these functions need to be applied to the whole signal. The continuous formulation is given here, but the filter is a discrete convolution for images.

$$h(\vec{x}) = \frac{1}{k(\vec{x})} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) c(\xi, \vec{x}) s(\mathbf{f}(\xi), \mathbf{f}(\vec{x})) d\xi$$

with the normalization: (21)

$$k(\vec{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \vec{x}) s(\mathbf{f}(\xi), \mathbf{f}(\vec{x})) d\xi$$

Higher values of σ_d and σ_r result in stronger filtering of the result. The bilateral filter is a single pass filter, but computationally expensive and slower than median blur.

4.2 Conversion from Depth Image to Feature Image

The conversion of depth images to feature images needs to result in a single scalar value because all keypoint detectors work on a single signal channel. Every feature image is calculated on range data in floating point format, potentially requiring the preprocessing step presented in Section 3.3.3. The representation change from integer to floating point data does no further processing than simple type conversion.

This section introduces and develops multiple potential feature images that were considered during this thesis. Conversions for images of the *Synthetic* scene (Section 5.3.1), containing different primitive geometric structures, serve as examples for the visual appeal of each feature image type.

4.2.1 Bearing-Angle Image

Scaramuzza [8] proposes the Bearing-Angle images that assigns each pixel the angle β, defined in Figure 11. This angle is spanned by connecting the two idealized light rays for pixels next to each other. The neighbourhood relationship between pixels can be chosen arbitrarily resulting in four Bearing-Angle images, horizontal, vertical, diagonal and anti-diagonal. The second variable is the direction the angle is calculated, e.g. for horizontal images it can be calculated from left-to-right or right-to-left. This does not exhibit new information, because the angle of the other direction is immediately known from the fact that the sum of the angles is $180°$. Nonetheless, the direction must be defined to obtain stable results.

The formula for the Bearing-Angle β is derived with the cosine theorem. Both Scaramuzza [8] and Lin et al. [9] have typos in the formulae they provided. A full derivation for the correct equation is provided in Appendix A. For the horizontal left-to-right calculation the formula is as follows:

$$\beta = \arccos \frac{d_{i,j} - d_{i-1,j} \cos \Delta\varphi}{\sqrt{d_{i,j}^2 + d_{i-1,j}^2 - 2 d_{i,j} d_{i-1,j} \cos \Delta\varphi}}. \tag{22}$$

The angular resolution $\Delta\varphi$ between two pixels of the depth image depends on the camera model in use. The pinhole model's angular resolution changes between pixel pairs, equirectangular image have a constant resolution. In general, the angle $\Delta\varphi$ can be calculated with the spherical coordinates $P_{\mathcal{S},i,j}^{\vec{}}, P_{\mathcal{S},i-1,j}^{\vec{}}$ (Equation 23).

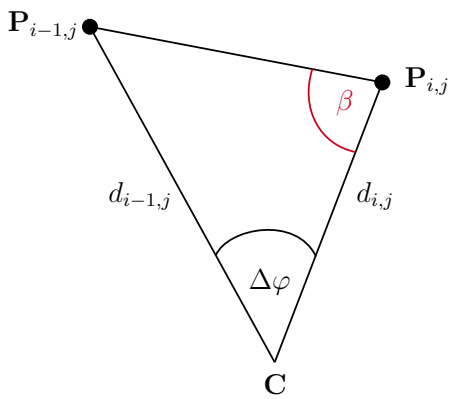


Fig. 11: *Schematic representation of Bearing-Angles.* This figure shows the relationship of the light rays that form the Bearing-Angle β.

$$\begin{aligned} \|\vec{P_{\mathcal{S},i,j}}\|_2 = \|\vec{P_{\mathcal{S},i-1,j}}\|_2 = 1 \implies \vec{P_{\mathcal{S},i,j}} \cdot \vec{P_{\mathcal{S},i-1,j}} = \cos\Delta\varphi \\ \Delta\varphi = \arccos \vec{P_{\mathcal{S},i,j}} \cdot \vec{P_{\mathcal{S},i-1,j}} \end{aligned} \tag{23}$$

The Bearing-Angle is in the range $\beta \in (0, \pi)\ rad$. Linear scaling of the angle to the color depth of the target image results in a grayscale image suitable for feature extraction. A general scaling function for an `unsigned 8 bit` image and arbitrary angle range is provided by Equation 24. This formulation can be used for different color depths and other potential angle calculations that result in different boundary conditions.

$$\begin{aligned} \beta_{min} &= 0 & c_{min} &= 0 \\ \beta_{max} &= \pi & c_{max} &= 255 \\ \beta_{scaled} &= \left\lfloor c_{min} + \beta \frac{c_{max} - c_{min}}{\beta_{max} - \beta_{min}} \right\rfloor \end{aligned} \tag{24}$$

Characteristics

A deeper analysis of the Bearing-Angle image helps to understand advantages and disadvantages for its usage. Figure 12 demonstrates the visual changes of a synthetic scene under certain camera transformations.

The most notable property is the lack of rotation invariance. This follows directly from the definition of the Bearing-Angle. A triangle is build by a predefined pixel relationship. Calculating the Bearing-Angle from multiple directions could lead to

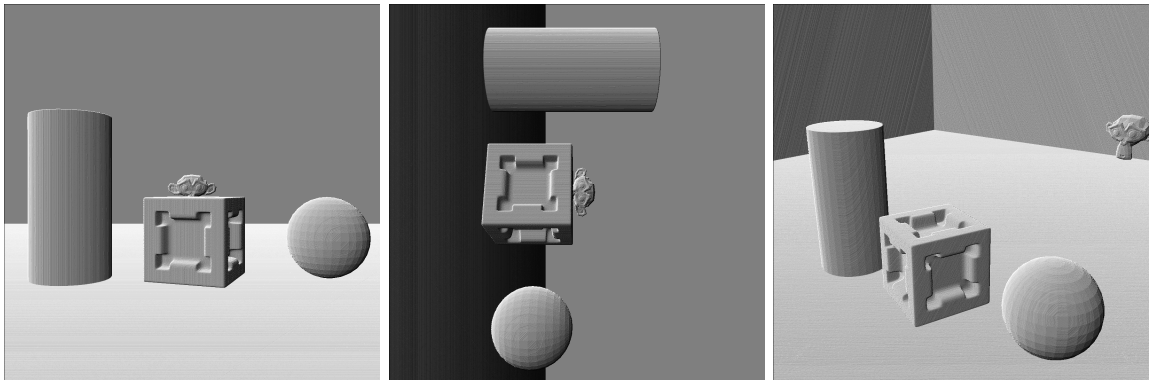


Fig. 12: *Bearing-Angle image characteristics.* The Bearing-Angle image is not invariant to rotation and viewpoint changes. This property limits its applicability for automatic registration of bigger discontinues changes of the camera pose. Each depth image was converted with the diagonal (top-left-to-bottom-right direction) implementation of the Bearing-Angle formula.

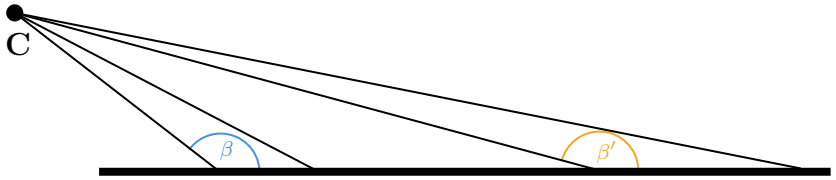


Fig. 13: *Two Bearing-Angles for the same ground plane.* Different shadings of plane surfaces depend on the perspective projection and the distance from the camera center C to the surface.

some rotation stability, because rotations of $45°$ corresponds to a different direction of pixel neighbourhood. The second apparent aspect is the change in shading, for example on the flat ground but on other surfaces, too. This effect is due to the perspective transformation and the distance of a point to the camera, as Figure 13 demonstrates. Round objects, like spheres, experience no visual change in shading. The triangles of the light rays are invariant to camera transformation for such objects.

4.2.2 Multi-Directional Bearing Angle Image

The *Multi-Directional Bearing-Angle image* attempts to overcome the limitation of missing rotation invariance for the classical Bearing-Angle image. Instead of computing one angle in one direction, two adjacent triangles are combined. Figure 14 shows how the triangles are related. Both angles β and γ are calculated with Equation 22. For γ the values of $d_{i,j}$ and $d_{i-1,j}$ need to be swapped. Both angles are

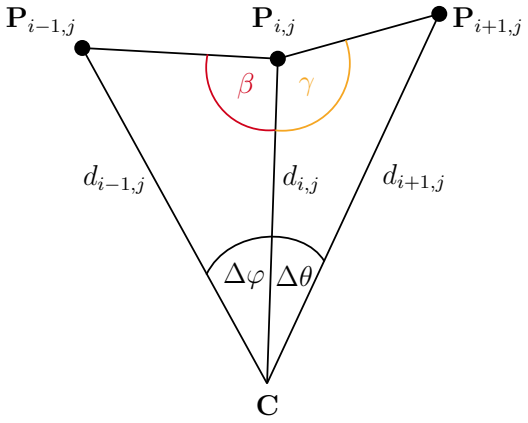


Fig. 14: *Schematic representation of the Multi-Directional Bearing-Angle image.* The Multi-Directional Bearing-Angle image composes two Bearing-Angles in vertical, horizontal, diagonal and anti-diagonal direction. The maximum angle is then selected as pixel value.

added together, resulting in the angle of the surface at this position.

$$\mathcal{B}_{direction} = \beta + \gamma \tag{25}$$

This calculation is again defined for only one direction and not rotation invariant, but already symmetrical. The value $\mathcal{B}_{direction}$ can be defined for diagonal, anti-diagonal, horizontal and vertical sampling of the image. The final step for rotation invariance is the reduction of these four values to their maximum:

$$\mathcal{B} = \max\left\{\mathcal{B}_{diagonal}, \mathcal{B}_{anti-diagonal}, \mathcal{B}_{horizontal}, \mathcal{B}_{vertical}\right\} \tag{26}$$

The converted images in Figure 15 show the same scene and positions as for the Bearing-Angle image. The goal of rotational invariance is achieved, unfortunately the image lacks texture and contrast. Qualitative tests with common feature detectors yielded weak keypoint responses. Therefore, this approach received no further attention.

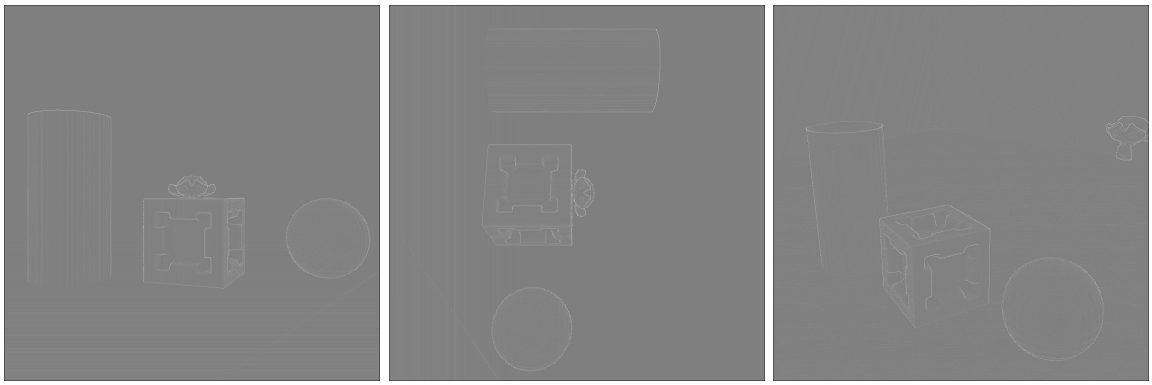
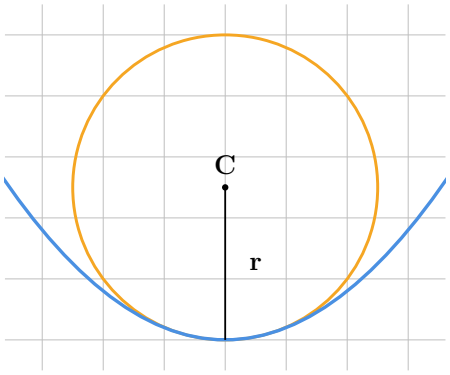


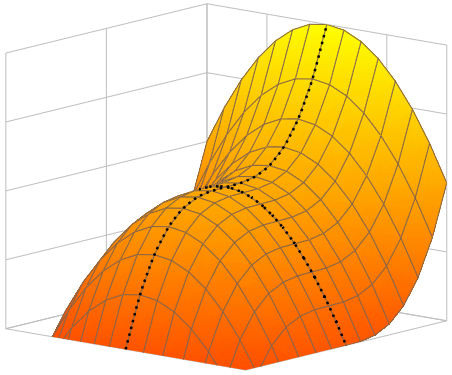
Fig. 15: *Rotation invariance of Multi-Directional Bearing-Angle image.* The images show the same camera positions as the Bearing-Angle and demonstrates the rotation invariance. Each edge in the geometry forms a visible white line. No other textures exist.

4.2.3 Curvature Image

Differential geometry provides a scalar quantity assignable to each point of a one-dimensional differentiable function, the *curvature* κ [52, p. 10]. This concept can be generalized to higher dimensions and two common curvatures exist for surfaces in three-dimensional space, *Gaussian curvature* and *mean curvature*. A straight line has



(a) The curvature of a line at a specific point is defined by its osculating circle.



(b) The principal curvatures are the maximum and minimum curvatures in all directions.

Fig. 16: *Curvature of curves and surfaces.* The curvature of a curve is defined through its osculating circle. This simple definition is not possible for surfaces in three dimensions. Analyzing all directions, each point on the surface does have a minimum and maximum curvature — the *principal curvatures*.

no curvature, hence $\kappa = 0$. The curvature of a circle is defined is the inverse of its radius

$$\kappa_{circle} = \frac{1}{r}. \tag{27}$$

Any point's curvature on a one-dimensional, twice differentiable function is defined as the curvature of its osculating circle, visualized in Figure 16a. The curvature of any point of a surface in three-dimensional space is ambiguous, because it has infinite many curves crossing through this point. But a maximum and minimum curvature exist, the *principal curvatures* k_1 and k_2. Figure 16b visualizes the directions of the minimum and maximum curvature of a surface in three-dimensional space as dotted black lines. The *Gaussian curvature* $\mathcal{K}$ is defined as the product

$$\mathcal{K} = k_1 k_2 \tag{28}$$

and the *mean curvature* $\mathcal{H}$ as the mean

$$\mathcal{H} = \frac{1}{2}(k_1 k_2) \tag{29}$$

of the principal curvatures.

Both quantities can be calculated differently, as the curvature is related to the derivatives of a function. Let $f(u, v)$ be a two-dimensional function representing the

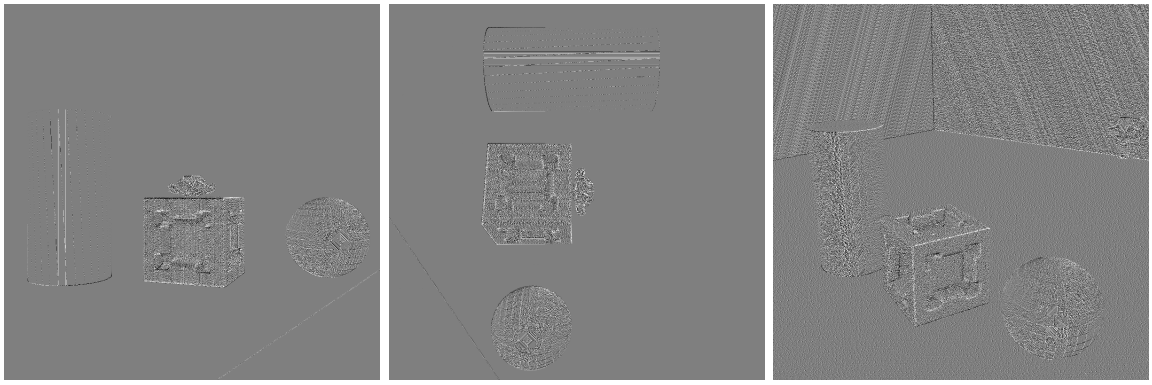


Fig. 17: *Mean curvature in the Synthetic scene.* The mean curvature feature image for the same synthetic scene are similar to Multi-Directional Bearing-Angle images, making them a second non-promising option.

depth image and f_u, f_v, f_{uu} and f_{vv} its partial derivatives. Then the Gaussian curvature and mean curvature are computed with:

$$\begin{aligned} \mathcal{K} &= \frac{f_{uu}f_{vv} - f_{uv}^2}{\left(1 + f_u^2 + f_v^2\right)^2}, \\ \mathcal{H} &= \frac{(1+f_v^2)f_{uu} - 2f_u f_v f_{uv} + (1+f_u^2)f_{vv}}{2\sqrt{1 + f_u^2 + f_v^2}^3}. \end{aligned} \tag{30}$$

The central differential quotients approximate the first and second derivatives with $\mathcal{O}(\Delta x^2)$ accuracy:

$$\begin{aligned} f_x &= \frac{y_{i+1} - y_{i-1}}{2\Delta x} \\ f_{xx} &= \frac{y_{i+1} + y_{i-1} - 2y_i}{\Delta x^2}. \end{aligned} \tag{31}$$

The result of the mean curvature conversion in Figure 17 is similar to the Multi-Directional Bearing-Angle image. The mean curvature is rotation invariant but the result is already very unstable to the noise induced by integer precision depth images. Figure 18 shows the images for the Gaussian curvature. It is even more prone to noise in the input.

Both the mean curvature and Gaussian curvature are unbound and result in any real number. Simple scaling between the computed minimum and maximum value is unstable between images and results in completely gray images as these quantities have noticeable outliers. As solution for this problem, the computed values are clamped to a predefined range, for example $\mathcal{H}, \mathcal{K} \in (-20, 20)$ and then scaled to the

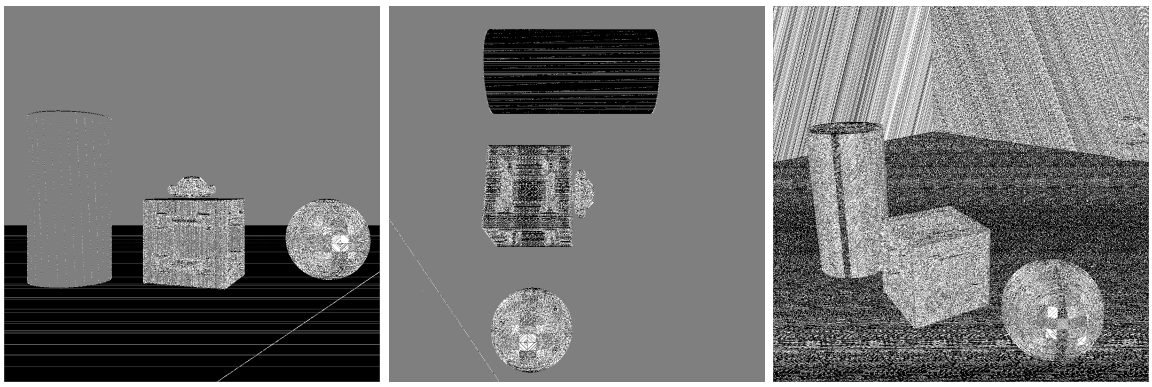


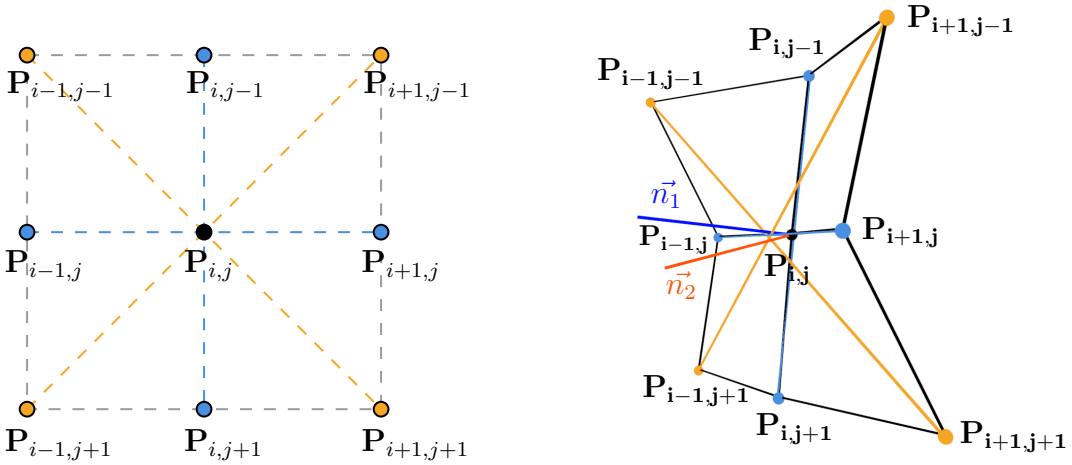
Fig. 18: *Gaussian curvature in the Synthetic scene.* The Gaussian curvature feature images for the synthetic scene. The results are not promising either.

desired image depth of the feature image. Computing the curvature for the whole image has the theoretical weakness, that discontinuities at object boundaries are not respected. The derivatives do not exist at these points for perfect depth sensing. A mathematically accurate computation requires object identification or meshing of the scene. Such a requirement seems unreasonable for a foundational processing step in potential real-time use cases.

4.2.4 Flexion Image

The local shape of an object that is sampled by a depth sensor is characterized by its surface normal. A surface normal is a vector perpendicular to the surface of the object. The core idea of the Flexion image is the approximation of the surface normal with two different sets of neighbouring pixels and measuring their difference.

Each measured pixel is back-projected to camera coordinates, scaling the spherical coordinates with its measured range as first step. The normal for a measured surface point $\mathbf{P_{i,j}}$ is calculated with the cross product of vectors connecting the neighbouring points of $\mathbf{P_{i,j}}$ (Equation 32). This point relationship is visualized in Figure 19a. Using the diagonal and vertical neighbouring points (blue) results in a different normal than the diagonal neighbours (orange) do. As Figure 19b demonstrates, both normals span an angle.



(a) The normals for a point $\mathbf{P_{i,j}}$ can be estimated by either its diagonal or horizontal and vertical neighbours.

(b) The estimated normals span an angle depending on the local shape of the measured surface.

Fig. 19: *Schematic representation of the Flexion image calculation.* This figure demonstrates how non-planar surfaces have different normals for diagonal and non-diagonal estimation. This difference is utilized as measure for surface flexion.

$$\begin{aligned} \vec{n_1} &= \frac{\vec{P_{i,j-1}} - \vec{P_{i,j+1}}}{\|\vec{P_{i,j-1}} - \vec{P_{i,j+1}}\|_2} \times \frac{\vec{P_{i-1,j}} - \vec{P_{i+1,j}}}{\|\vec{P_{i-1,j}} - \vec{P_{i+1,j}}\|_2} \\ \vec{n_2} &= \frac{\vec{P_{i-1,j-1}} - \vec{P_{i+1,j+1}}}{\|\vec{P_{i-1,j-1}} - \vec{P_{i+1,j+1}}\|_2} \times \frac{\vec{P_{i-1,j+1}} - \vec{P_{i+1,j-1}}}{\|\vec{P_{i-1,j+1}} - \vec{P_{i+1,j-1}}\|_2} \end{aligned} \tag{32}$$

It shall be pointed out, that both normals $\vec{n_1}$ and $\vec{n_2}$ are not unit length but each factor of the cross product is. Finally, the Flexion $\mathcal{F}$ of point $\mathbf{P_{i,j}}$ is defined as:

$$\mathcal{F} = \|\vec{n_1} \cdot \vec{n_2}\|_2. \tag{33}$$

Because $\|\vec{n_1}\|_2, \|\vec{n_2}\|_2 \in [0, 1]$ the value of is bound to $\mathcal{F} \in [0, 1]$. Creating the final image requires a linear scaling to the desired output image depth using Equation 24.

Characteristics

Figure 20 visualizes the Flexion image for the same camera positions as for the other feature images. The Flexion image is rotation invariant. Rotation of either an object or the camera does not change the difference between the two normal approximations. A flat surface has an almost constant shading, because the normal approximation results in the same vector directions. Flat surfaces not perpendicular to the camera

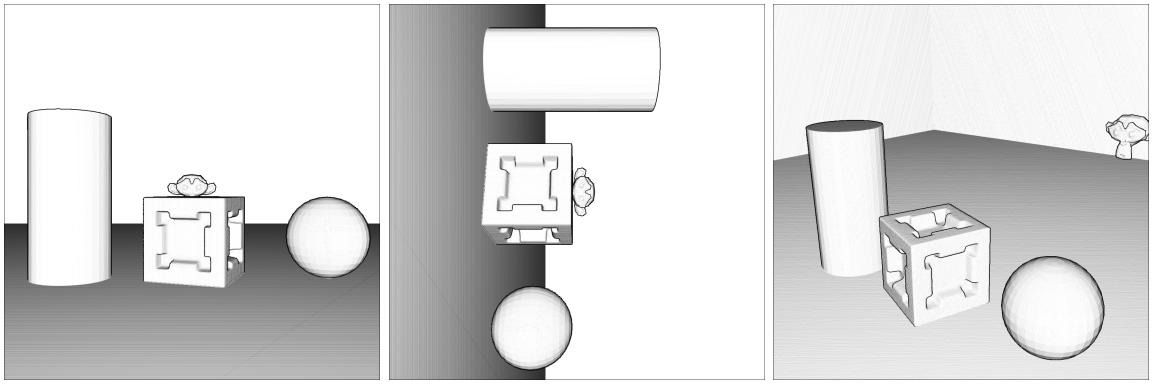


Fig. 20: *Characteristic look of a Flexion image.* These figures demonstrate the characteristic look of Flexion images. Their appearance is very plastic and the shading effects give a good sense for depth. The conversion is rotation invariant.

plane have non-constant shading with the brightness reducing the further the surface patch is apart from the camera center. This effect is caused by the perspective transformation. The norm of cross product is maximal if both multiplied vectors are perpendicular:

$$\|\vec{v_1} \times \vec{v_2}\|_2 = \|\vec{v_1}\|_2 \|\vec{v_2}\|_2 \sin \angle(\vec{v_1}, \vec{v_2}). \tag{34}$$

Because the vectors multiplied to form the normals n_1 and n_2 have different angles based on their distance to the camera (Figure 21), the lengths of the normals differ throughout the image. The lack of normalization of $\vec{n_1}$ and $\vec{n_2}$ propagates this effect through to the scalar product, as the length of the both vectors are multiplied:

$$\vec{v_1} \cdot \vec{v_2} = \|\vec{v_1}\|_2 \|\vec{v_2}\|_2 \cos \angle(\vec{v_1}, \vec{v_2}). \tag{35}$$

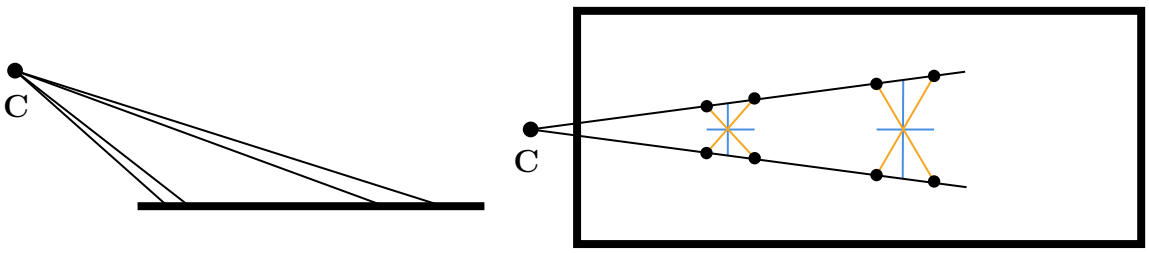


Fig. 21: *Explanation of shading effect in Flexion images.* The angle between the diagonals decreases with increasing distance from the camera center. This results in shorter normals.

4.2.5 Implementation Details

All software developed during the thesis, including the analysis and supplementary code, are developed with rigor software engineering methods. The library components have 100 % unit- and integration-test line coverage. Each final executable is heavily tested, too. The overall line coverage for all project code is above 98%. The implementation language is C++-17 [53] and all library dependencies use at least C++-11 [54]. All code obeys to strong typing, design by contract [55] and modern idioms of the C++ programming language [56]. To uncover runtime problems potentially present in C++ through raw memory access, the LLVM Address-, Memory-, Thread- and Undefined-Behaviour-Sanitizers [57] are run over all tests. Additional static analysis is done by clang's thread-safety analysis [58], clang static analyzer [59] and clang-tidy [60]. All detected issues were immediately fixed during development. The use of continuous integration [61] during development indicated defects within hours and ensured fast development of code with a low defect rate.

The implementation goal of the developed software is to serve as a correct reference implementation for the proposed feature images. Therefore, no special action has been taken to improve latency or throughput of the computations. Only simple measures for speedup, namely exploiting the embarrassingly parallel nature of the processing and compiler optimizations are employed. Figure 22 shows the results of micro benchmarks of the two promising feature images. Each of the feature im-

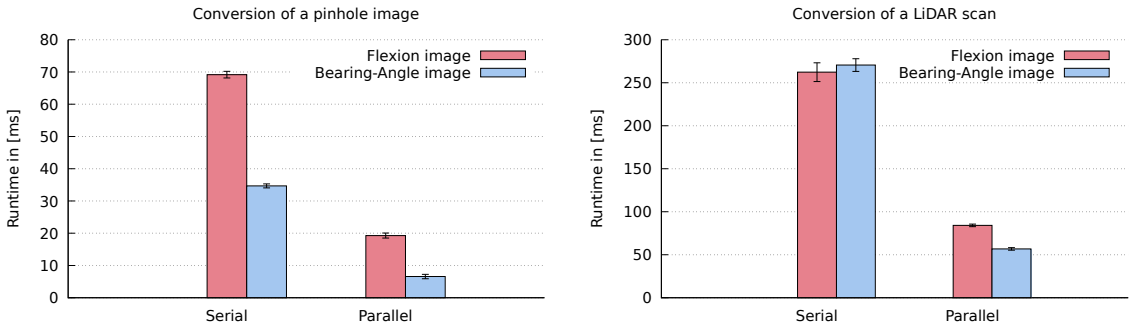


Fig. 22: *Benchmarks for Flexion and Bearing-Angle image conversions.* The benchmarks are run on an Intel i7-8550U with 1.8 GHz base clock and 4 hyper-threaded cores. Each conversion is repeated 100 times for solid statistical data. The resolution of the pinhole image is 960×540 px and the LiDAR scan is 3600×800 px big. The higher memory consumption of the LiDAR scan results in a memory bound execution for the single threaded conversion.

age conversions is implemented as C++ library code working on OpenCV's [48] `cv::Mat` matrix type. The conversion is generic in the sense, that any camera model implementing the forward and backward projection for pixel coordinates is suitable for the feature image conversion. This generality is achieved through the use of templates. Type requirements are enforced with `static_assert()` and concept-like [62] requirement definitions. One important type ensures consistent use of coordinate systems with a special type that prohibits the code:

```
coordinate<float, frame::pixel> px = coordinate<float, frame::image>(0.5F, 0.25F);
```

to compile. All image access operations must adhere to the reference frame annotation and special transformations, like projections, are the only way to switch between coordinate systems.

Multiple library dependencies support the functionality of the project and shall be mentioned without a particular order. The already mentioned OpenCV [48] project provides functionality for image handling and processing. Required types and functionality for glue-code, parallelization and general programming utilize *cli11* [63], *rang* [64], *cpp-taskflow* [65], *fmt-lib* [66], *GSL* [67], *Eigen3* [68] and *Boost* [69]. Functionality and performance testing are done with *doctest* [70] and *libnonius* [71]. Each used revision is documented in the code repository and differs between versions of the thesis code.

4.3 Feature Detection and Description

After preprocessing and image conversion, the final processing step of the range data is the feature detection and description. This thesis analyzes SIFT [12], SURF [10], ORB [26] and AKAZE [27]. Although each keypoint detector and descriptor is designed with gray-scale images in mind, they are optimized for classical camera images that provide more texture than the converted feature images. Key to all algorithms is the search for salient regions in the image that can be consistently detected from different viewpoints and changing environmental conditions. The detector performance is critical to apply classical image registration approaches to the converted feature images. The following sections introduce the algorithms briefly and mentions

key aspects of their functionality. Details and foundational concepts are out of scope and the original publications are highly recommended for further comprehension.

4.3.1 SIFT (Scale Invariant Feature Transform)

Introduced by Lowe [12, 72], SIFT is an established feature detection algorithm. The design goal of the algorithm is to detect keypoints that are invariant with respect to scale and rotation. Robustness against noise, change in illumination and affine transformations is additionally considered. SIFT's design influenced the design decisions of the other presented algorithms heavily.

The input image is processed in a hierarchical pyramid of down-scaled versions of the image, called octaves. On each octave a Gaussian filter is applied consecutively with increased standard deviation on each run. The difference between two consecutive blurred images is computed, the so called Difference of Gaussian (DoG) forming an approximation of the Laplacian of the image. Lindeberg's [73] work on scale space and automatic scale selection gives a theoretical introduction why this process gives a good way to select the scale of features. The underlying principle exploits that fine structures dissolve at bigger scales. This process is related to the diffusion equation that is strongly tied to the Laplacian of the image. Keypoint candidates are local extrema in these DoGs. Each candidate is filtered by a contrast threshold, its accurate position is determined using a local spline interpolation and its scale is assigned based on the octave and blurring factor. Edge-like responses are filtered with an estimate of the ratio between the principal curvatures. Highly skewed curvatures indicate an edge.

Every keypoint candidate gets one or more orientations assigned. These are computed by storing the brightness gradients of the local neighbourhood in an orientation histogram. Each peak in this histogram reflects a dominant direction of gradients and the corresponding angle is used as the orientation of the keypoint. According to Lowe [12], about 15% of keypoints have multiple peaks and get multiple orientations assigned. The orientation is used to rotate the local neighbourhood of the keypoint to compute the descriptor consistently.

The image gradients direction and magnitude are computed for each pixel in a local environment, first weighted by a Gaussian to magnify the influence of close pixels and then binned into orientation histograms. Each histogram reflects a fraction of the sampling grid. A 16×16 grid around the keypoint, that is stored to 4×4 histograms with 8 bins for the orientations, results in a vector of 128 elements in the descriptor. These dimensions can be adjusted, but the 128 element descriptor is most common. Finally, the histograms are normalized to increase robustness against illumination changes. The Euclidean distance of the descriptor vectors is the most common similarity measure. Arandjelović and Zisserman [74] introduced RootSIFT using the Hellinger distance [75] as better measure for similarity of histograms.

4.3.2 SURF (Speeded-Up Robust Features)

Bay et al. introduced SURF [10] to achieve similar detector and descriptor performance as SIFT but at a lower computational cost. SURF utilizes integral images [76]. Each pixel's value is the sum of all pixels in the rectangle formed by the origin and the pixel itself. This representation reduces computational complexity.

The underlying principal of scale space and extrema detection derives from the same principles as for SIFT but undergoes simplification. The Gaussian convolution is approximated with a box filter that approximate a second order Gaussian derivative and is related to the Hessian matrix. Instead of building a pyramid of images at different scales, the filter itself is scaled up successively. Maxima undergo a non-maximum suppression at different scales. The maximum of the Hessian determinant for the detected pixel is interpolated in image scale and space. The determinants value is the keypoint's response.

Orientation assignment of SURF can be skipped for scenarios that do not involve camera rotation. The rotation is derived from the Haar wavelets [77] response in x- and y-direction in the local neighbourhood. The descriptor itself is constructed from a 4×4 grid structure of sample points around the detected keypoint. Again, the Haar wavelets response in x- and y-direction are used to derive the elements of the descriptor.

4.3.3 ORB (Oriented FAST and Rotated BRIEF)

ORB [26] combines the FAST [78] (Features from Accelerated Segment Test) feature detector and BRIEF [79] (Binary Robust Independent Elementary Features) descriptor with additional improvements to both algorithms. The design goal is to achieve similar matching performance to SIFT and SURF but to be computationally cheaper to make usage in embedded devices feasible.

The FAST detector evaluates each pixel by testing the brightness of 16 pixels on a circle around it. If there are enough consecutive pixels brighter or darker than the center with a certain threshold, the pixel is accepted as corner. This approach results in a decision tree. The optimal traversal of this tree is learned by observing real world execution of the algorithm on predefined traversal. ORB detects keypoints with FAST, filters edge results with the Harris corner detector [25] and assigns scale using image pyramids. The orientation of the keypoint is computed with the *intensity centroid* [80].

Each keypoint's descriptor uses a rotated version of BRIEF. BRIEF is a binary descriptor built from a set of binary intensity tests of consecutive pixels around the keypoint. The more recent publications, like BRIEF, ORB and M-LDB, employ binary descriptors for their fast matching speed. The distance of binary descriptors can be computed with the Hamming distance that reduces to *XOR* instructions on machine level. Ziegler et al. [81] provide a theoretical analysis of BRIEF and its siblings and show, that those are hashing schemes of Kendall's τ metric [82].

4.3.4 AKAZE

KAZE's [83] (Japanese for wind) and AKAZE's [27] (Accelerated KAZE) approaches are similar to SIFT's. Again, the diffusion of the brightness is the starting point for scale space analysis. KAZE applies non-linear diffusion filtering with a special computational scheme, that makes it feasible to solve the diffusion in real time. The difference to the Gaussian blurring is the preservation of edges. The Gaussian blurs the whole image, reducing the signal crispness at higher scales. Contrary, the non-linear diffusion preserves edges over higher scales. After non-maximum suppression, the determinant of the Hessian is computed and the interpolation of the maximum on

different scales yields the position of the keypoint. The orientation of the keypoint is, similar to SIFT, the dominant direction of the derivatives in the local neighbourhood of the keypoint.

The descriptor is M-LDB (Modified-Local Difference Binary), which is based on LDB [84] (Local Difference Binary). It works very similar to BRIEF. But instead of comparing intensities of sampled pixel, it compares the average intensities of sampled areas around the keypoint. The orientation of the keypoint is taken into account when sampling the areas.

4.3.5 Short Comparison

Table 2 gives a short overview of the presented descriptors and how their runtime and usage characteristics compare to each other. This overview does not reflect the performance of the algorithms in terms of accuracy or descriptiveness, but compares design decisions and execution characteristics. All presented algorithms are designed with color images in mind and all previous performance evaluations are not significant for the novel feature image types.

	SIFT	SURF	ORB	AKAZE
Detector Speed	-	+	++	+
Descriptor Speed	- -	-	++	+
Matching Speed	- -	-	++	++
Memory Footprint	- -	-	++	++

Tab. 2: *Comparison of feature detection algorithms.* The algorithms have different runtime requirements and characteristics. The comparison is just based on the design decisions of each algorithm, but does not reflect real world performance of an algorithm integrated in a complex use case.

5 Experiments

The experiments to evaluate the performance of Bearing-Angle and Flexion images span a variety of sensors, datasets and processing setups. First, the metrics to determine if a feature algorithm produces utilisable results are explained. Then, the applied settings of the algorithms and evaluation datasets are presented.

5.1 Metrics

To determine the performance of the feature-based approach, the matching of keypoints is framed in terms of a binary classifier. For a keypoint correspondence between two images, the classification task is to determine if those keypoints correspond to the same point in the real world or not. The correspondence is computed based on the relative pose of consecutive images and only two consecutive images are analyzed. The following subsections describe each component of this evaluation pipeline in more detail.

5.1.1 Ground Truth Poses

The dataset *Mine* provides ground truth poses from a prior Structure-from-Motion reconstruction. Due to the global optimization of the Structure-from-Motion algorithm, these poses do not match with the depth values of the depth images. The other datasets do not contain a precomputed pose. Therefore, each pose is computed and refined with an ICP algorithm, namely OpenCV's `FASTICPOdometry` [48] that is based on the KinectFusion [85] work. This relative pose serves as foundation of the evaluation and is assumed to be correct within small tolerance. The accuracy assumption of the `FASTICPOdometry` poses is tested by manual inspection of the backprojection of obviously related keypoints and the results showed no systematic erroneous poses.

5.1.2 Backprojection and Distance Threshold

In two consecutive images, all keypoints are extracted with the chosen algorithm and settings. Each keypoint of the first image I_1 is projected to the unit sphere, e.g. with

Equation 5, multiplied with the range value of the corresponding pixel. The resulting camera coordinate is transformed to the second camera coordinate system (Equation 1). Projecting the keypoint into the second camera with the matching forward projection (e.g. Equation 2) finalizes the procedure, demonstrated in Figure 23. This results in two sets of keypoints in the second image I_2, the keypoints detected in this frame and the keypoints detected in the previous frame, seen from this frame's pose. Assuming a correct relative pose, the corresponding keypoints have a small pixel distance between each other. *Small* is a relative value and the chosen threshold in the experiments is 2 px. This approach can only result in a proper analysis for a sparse distribution of keypoints that actually correspond to image structure and requires small changes in pose. When the pose difference is big, it can happen that keypoints get projected to the same position as some other valid keypoint without referencing the same world point. Too many points create ambiguity with respect to the backprojection tolerance.

First, OpenCV's `BFMatcher` [48] matcher establishes correspondences based on the descriptor distance and applies crosschecking — the descriptors must have minimal distance in both directions — as only heuristic. The matches are partitioned into four sets: *true-positive*, *false-positive*, *true-negative* and *false-negative* as introduced in Section 3.5. The union of these sets are all detected keypoints. First, the matches are analyzed for true and false positives and true correspondences are removed from

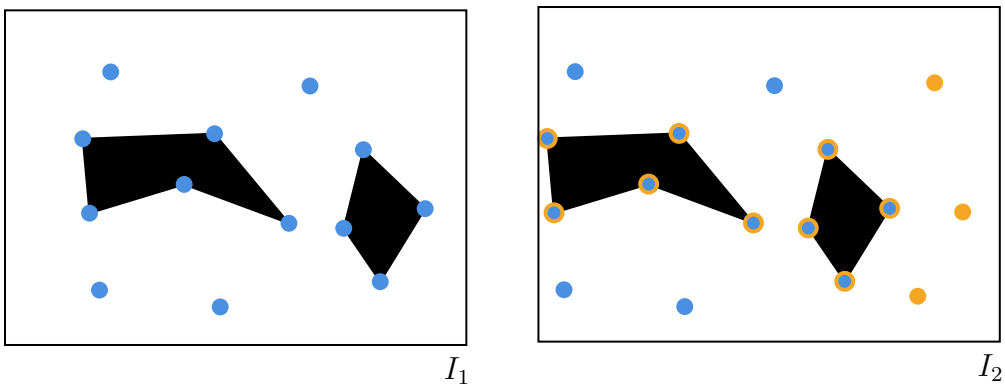


Fig. 23: *Backprojection of keypoints visualized.* The chosen approach of projecting keypoints between frames is demonstrated in this figure. Keypoints of image I_1 are blue dots in both images and the orange ones are only detected in image I_2. Actual correspondences result in very close points in I_2, indicated as blue dot with orange border, whereas unrelated points show no proximity. This assumption holds for small changes in pose and relatively sparse distribution of detected keypoints.

further analysis. Let $proj$ be the keypoints from I_1, backprojected to I_2. kps are the keypoints from I_2, $matches$ are pairs of keypoints the matching algorithm considers as correspondences and $threshold$ is the maximum allowed backprojection error for keypoints to still be considered a true correspondence. Algorithm 1 classifies all $matches$ as true or false positives.

Algorithm 1 This algorithm distinguishes between a true and a false positive match.

Require: $\forall m \in matches \Rightarrow m.first \in proj \wedge m.second \in kps$
Require: $threshold > 0.0$
Ensure: $|true| + |false| = |matches|$
  function CLASSIFYPOSITIVES(proj, kps, matches, threshold)
    for all $m \in matches$ **do**
      if DISTANCE$(proj[m.first], kps[m.second]) < threshold$ **then**
        $true \leftarrow true \cup m$
        $proj \leftarrow proj \setminus m.first$ ▷ Prevent double usage
      else
        $false \leftarrow false \cup m$
      end if
      $kps \leftarrow kps \setminus m.second$
    end for
  end function

After this process some keypoints detected in I_2 might be left to analyze for a correspondence that were not assigned during matching, the false negatives (Algorithm 2).

Algorithm 2 The unmatched keypoints are classified as true or false negative.

Require: $threshold > 0.0$
Require: $kps_{pre-call}$ does not contain matched keypoints
Ensure: $|false\_negatives| + |true\_negatives| = |kps_{pre-call}|$
  function CLASSIFYNEGATIVES(proj, kps, threshold)
    for all $k \in kps$ **do**
      $closest \leftarrow$ FINDCLOSESTFROM$(proj, k)$
      if DISTANCE$(closest, k) < threshold$ **then**
        $false\_negatives \leftarrow false\_negatives \cup (k, closest)$
        $proj \leftarrow proj \setminus closest$
      end if
    end for
    $true\_negatives \leftarrow kps \setminus false\_negatives$
  end function

To classify the negatives, each keypoint's distance from I_2 to the backprojected keypoints from I_1 is tested to be within the defined threshold. If this is the case, both keypoints are defined as corresponding and result in a false negative. The remaining points are true negatives. The partitioning of the keypoints allows to analyze more aspects of matches, e.g. the descriptor distances for true and false positives.

5.1.3 Histograms and Summary Statistics

To visualize and understand the properties of keypoints and matches, the partitioned keypoint's are tracked in histograms. Summary statistics reduce the statistical distributions to a few representative values. Combining different measures, such as the measure of location, distribution and shape gives key insights.

Both histograms and its summary statistics are created for the keypoint size, keypoint response, descriptor distance between all keypoints and both descriptor distance for *true-positives* and *false-positives*. The quantitative measures for each property are the *minimum* and *maximum* value, *median* and *arithmetic mean*, *variance* and *standard deviation* and *skeweness* of the distribution.

5.1.4 Classification Evaluation

The analysis of the keypoint and descriptor characteristics already gives some insight into the algorithm performance but is not suitable for a comparison between different algorithms. For this task the quality of the decisions the keypoint matching algorithm must be assessed. The evaluation builds on the analysis of binary classifiers as introduced in Section 3.5.

The matching of descriptors is done in a simple but consistent way. A match is defined as the closest descriptor in the other image, that also matches in the other direction, a heuristic commonly called *crosschecking*. No other criteria like a maximum matching distance is taken into account. For each obtained confusion matrix, the ratios *precision*, *recall* or *sensitivity*, *fallout* or *false alarm rate*, *accuracy* or the *Rand index* and *Youden's index* are computed at first. For a comparison be-

tween algorithms and configurations the results are plotted in Receiver-Operating-Characteristics (ROC) space.

5.2 Parameter Search

As first step the unfiltered depth images are used for the conversions. The keypoint count, distribution, response and size are analyzed. Different algorithm and filter configurations are analyzed and compared to this baseline. Notable differences receive further attention and analysis.

The effect of different filters for the depth image is tested by running the median blur and bilateral filter on the raw range data, effectively creating a new depth image source. Each of those data sources is analyzed with the same settings for each algorithm. The total number of configurations is the Cartesian product of the data sources (Table 3) with each algorithm configuration (Tables 4 and 5). Only the *Synthetic* dataset is not filtered as its range values have no measurement error.

	Median Blur	**Bilateral Filter**
Parameters	$w = 5 \times 5px$	$\sigma_{color} = 25$ $\sigma_{space} = 7$

Tab. 3: The parameters for the filters applied to the sensor data.

Name	**SIFT Parameters**	**AKAZE Parameters**

Default	`keypoint size > 5px`
	`contrast threshold = 0.04`
	σ `= 1.6`
		`descriptor: M-LDB`
		`diffusity: PM_G2`

Best only	`best 400 keypoints`	`best 400 keypoints`
Variation 1	`contrast threshold = 0.02`	`descriptor: KAZE`
Variation 2	`contrast threshold = 0.08`	`diffusity: PM_G1`
Variation 3	σ `= 1.0`	`diffusity: WEICKERT`
Variation 4	σ `= 2.2`	`diffusity: CHARBONNIER`

Tab. 4: *Parameters evaluated for SIFT and AKAZE. Default* is used as basis for all configurations and only differences are presented. The keypoints of SIFT are filtered by keypoint size, because many small keypoints were detected on artifacts of sensor noise.

Name	SURF Parameters	Name	ORB Parameters
Default	`octaves = 4` `layers per octave = 1`	Default	`Score Metric: FAST`

Best Only	`octaves = 1`
	`layers per octave = 1`
	`best 400 keypoints`
		Default Harris	`Score Metric: Harris`

Tab. 5: *Parameters evaluated for ORB and SURF.* Both ORB and SURF are only analyzed with two configurations. The algorithms showed systematic issues when applied to the novel feature images, that are discussed later.

5.3 Datasets

5.3.1 Synthetic

The first dataset is a manually created scene in Blender [86] demonstrating the principle effects of rotation and translation on the conversion results. Both rotation and translation are done in isolation as well combined. The scene consists of a sphere, a cylinder, a cube-like object with additional edges of different smoothness and a complex monkey head in a room (Figure 24). The camera movement is rendered as an animation and the depth buffer of each frame is extracted and used as depth image. The camera matrix is calculated from the rendering settings and its parameters are provided in Table 6. The total animation consists of 211 images and no noise is applied to the depth image (example images in Appendix B.1).

Parameter	Blender Camera
Principle	render/pinhole
Resolution	1080 $\times$ 1080 px
f_x, f_y	2220.0, 2220.0
c_x, c_y	540.0, 540.0

Tab. 6: Blender camera intrinsic

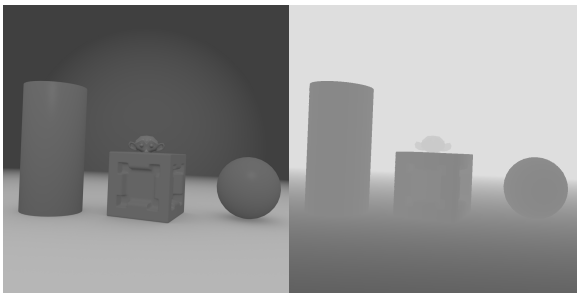


Fig. 24: One frame as normal synthetic and depth image.

5.3.2 Mine

One real-world dataset is obtained with a mobile robot using a Kinectv2 in an underground mining environment, the *Lehrpfad* of the Reiche Zeche, the education and research mine of the TU Bergakademie Freiberg. The route has been previously reconstructed with Structure-from-Motion using color images. Due to the global optimization in the Structure-from-Motion pipeline not preserving the proper scale for the depth images, the translations do not match the measured distances of the depth sensor. The preexisting poses are used as initial pose for ICP refinement. *Mine* is the biggest tested dataset with 734 frames and has the biggest variation in the visible structures (example images in Appendix B.5). It is a challenging dataset with high amount of noise, many missing depth values and differing data quality over the whole set of images.

Parameter	**Kinectv2**
Principle	pinhole
Resolution	960 × 540 px
f_x, f_y	519.23, 522.23
c_x, c_y	479.46, 272.74

Tab. 7: *Mine* Kinectv2 intrinsic.

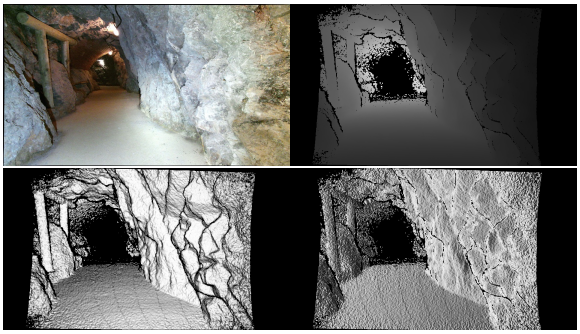


Fig. 25: Images of the *Mine* dataset.

5.3.3 Office

The second Kinectv2 dataset is taken in an office with many smaller elements, wires and other man-made objects. It contains 57 images composed of translation and rotation of the depth sensor (example images in Appendix B.2). No trajectory reconstruction other than the ICP is used for the relative poses. It contains similar measurement errors as the *Mine* dataset, but has more distinctive shapes and better overall conditions for the depth sensor.

Parameter	Kinectv2
Principle	pinhole
Resolution	960 × 540 px
f_x, f_y	519.23, 522.23
c_x, c_y	479.46, 272.74

Tab. 8: *Office* Kinectv2 intrinsic.

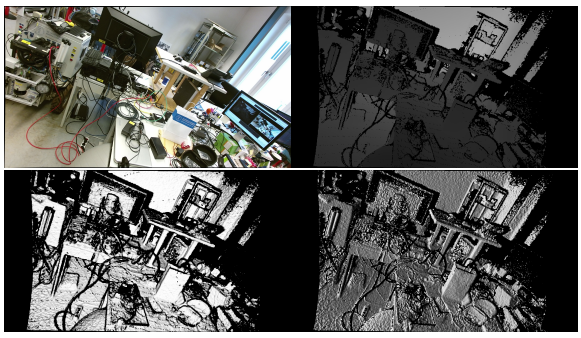


Fig. 26: Images of the *Office* dataset.

5.3.4 Laserscan

Full LiDAR scans are taken as part of classical mine surveying in the Reiche Zeche. The scans were done in the section *Wilhelm-Stehender-Süd* of the mine. These scans are examined for the conversions, too. Because they were done with classical artificial marker based registration in mind, the overlap between those scans is minimal. This unfortunately means, that a direct matching between scans would be meaningless. The distribution and characteristics of the extracted features is analyzed, though. From the raw dataset only 6 scans are selected, because each scan had a slightly different resolution and aspect ratio (example scans in Appendix B.3). The selected scans were manually processed to have a common resolution and aspect ratio.

Parameter	Riegl Z300
Principle	equirectangular
Resolution	3600 × 800 px
θ_{min}, θ_{max}	0.87, 2.27 (49.85°, 130.06°)

Tab. 9: Riegl Z300 LiDAR intrinsic.

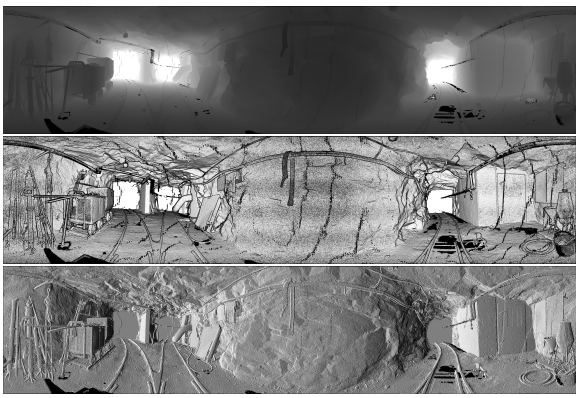


Fig. 27: The raw LiDAR scans and their conversions.

	Synthetic	Mine	Office	Laserscan
Camera Model	pinhole	pinhole	pinhole	equirectangular
Number Images	211	734	57	6
Distribution	✔	✔	✔	✔
Keypoint Characteristics	✔	✔	✔	✔
Matching Performance	✔	✔	✔	✘

Tab. 10: An overview of all datasets and aspects of them were analyzed.

5.3.5 Synthetic City Scene Odometry

The datasets presented before evaluate the feature algorithms and show their potential and performance. To demonstrate, that the results are applicable in other scenarios, the most promising algorithms are used to compute a visual odometry. A synthetic city dataset (Section B.4), developed by Zhang et al. [87], that provides a ground truth trajectory, serves as test. The in-house visual odometry implementation, based on Singh's [88] work, is very basic with the following four steps. First, the features are detected and their descriptors extracted. The descriptors are matched. The essential matrix is computed using RANSAC and the consensus pose is used as relative pose. Each feature algorithm is executed with its default configuration. These conditions imply that the result is not optimal and further optimizations would improve it further.

6 Results and Discussion

First, the evaluation of the keypoint detector and descriptors is presented for SIFT, SURF, ORB and AKAZE with the focus on central results. The out-of-the-box performance of each algorithm is compared with multiple custom processing configurations. After a short conclusion of the feature performance, SIFT and AKAZE are selected to compute a visual odometry on an unseen benchmark dataset. Subsequently, the results are discussed.

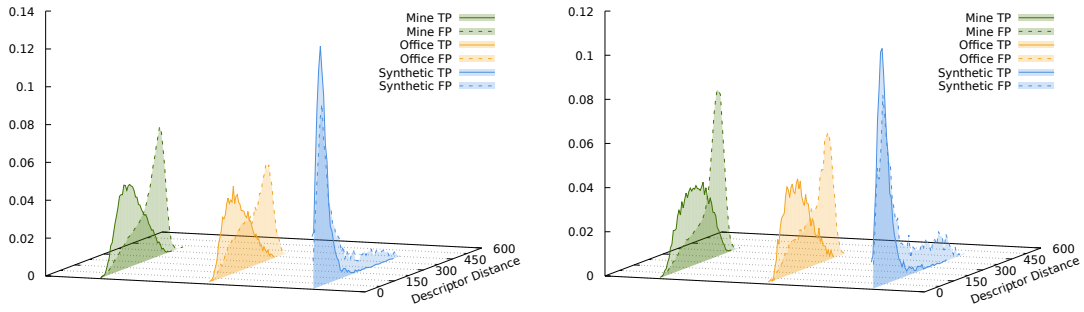
6.1 Algorithm Performance

6.1.1 SIFT

The number of detected keypoints is significantly higher for the Flexion images compared to the Bearing-Angle images in each dataset (Table 11). Approximately twice as many correspondences between keypoints are detected for Flexion images. The distribution of the keypoints in each dataset indicates no obvious problems in any regard (Figure D54). In *Mine*, the keypoints are not as common in the ground area but denser on the walls. This is expected, as those have the most surface structure. Similarly in *Synthetic*, the rotation of the camera shows a clear trace in the keypoint distribution, demonstrating that they stick to geometrically salient regions. Neither the

	Synthetic$_{\mathcal{F}}$	**Synthetic**$_{\beta}$	**Mine**$_{\mathcal{F}}$	**Mine**$_{\beta}$	**Office**$_{\mathcal{F}}$	**Office**$_{\beta}$
Keypoint Count	15,856	7,851	321,529	243,131	30,614	21,052
Correspondences	6,162	3,636	23,035	11,428	5,288	2,437
True Positives	5,997	3,580	18,491	8,362	4,524	2,119
False Positives	4,010	1,711	108,468	85,075	9,348	7,088
False Negatives	165	56	4,544	3,066	764	318
Precision	0.599	0.677	0.146	0.090	0.326	0.230
Recall	0.973	0.985	0.803	0.732	0.856	0.870
Youden's index	0.557	0.576	0.439	0.364	0.480	0.483
Accuracy	0.736	0.774	0.648	0.637	0.665	0.643

Tab. 11: *Keypoint and matching results for* `SIFT/raw/default`. The Flexion image results in more detected keypoints for all datasets and hence more correspondences. The recall of SIFT is outstanding and above 80% for even the demanding *Mine* dataset.



(a) Flexion image **(b)** Bearing-Angle image

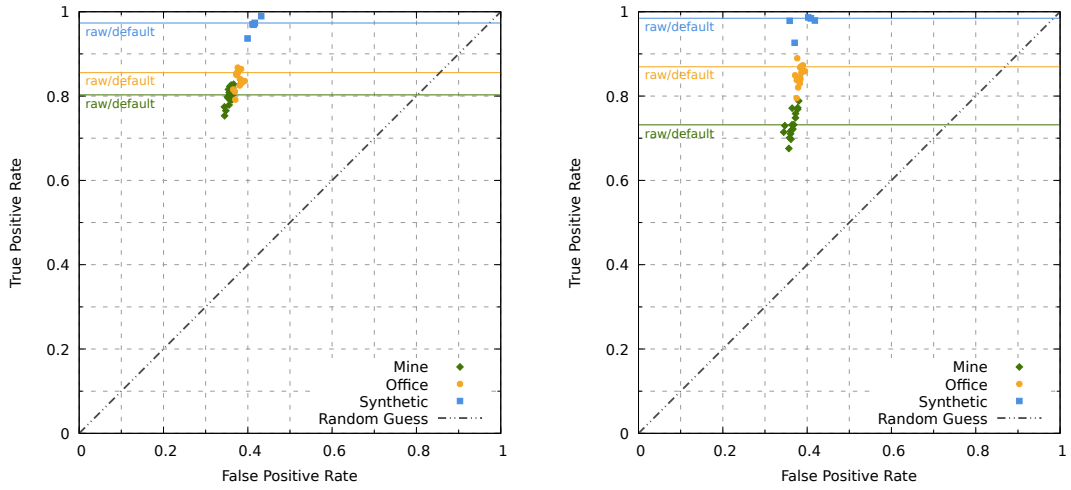
Fig. 28: *SIFT descriptors distances.* The distributions of descriptor distances show a clear relationship between the distance and the likelihood of a true or false positive. This insight justifies a maximum matching distance to separate false positives better from true positives.

keypoint sizes, responses or location distribution show surprising characteristics and their distributions are plotted in Appendix D.1.

The most relevant criterion for matching performance is the descriptors capability to differentiate between regions. For each dataset, except *Synthetic*, the distribution of the descriptor distance of matches shows that true positives and false positives result in a different distribution (Figure 28). Each curve is normalized with the total number of true and false positives per dataset for comparison. Defining a maximum matching distance threshold would result in less false positives for both conversion types. The overlapping curves for *Synthetic* are an artifact of the inherent similarity of different regions and can not be avoided. Appendix C.1 contains plots visualizing the matching performance in each dataset in the actual images. In terms of the ROC analysis (Figure 29) the Flexion image shows a slightly better performance, but no clear advantage can be established from the plot. Additional parameter variation in the SIFT algorithm and filtering setup varies the true positive rate by only 5% and the false positive rate by even less.

The differences between the datasets are inherent to their quality and setup. Most notably, SIFT has a very high true positive rate (recall) showing its descriptive power and potential for object recognition tasks. The very similar false positive rate in all datasets points to inherent similarities in the geometries of the scenes that might be explained by the lack of textures as differentiating factor.

Both feature image conversions demonstrate that the salient geometric regions can be recognized with SIFT. The Flexion image produces more keypoints and correspondences compared to the Bearing-Angle image. Additionally, the true positive rate in *Mine* is slightly higher in this challenging dataset. From these findings the author concludes that the Flexion image performs better. The theoretical aspects, like rotation invariance and the inclusion of more information per pixel, are additional arguments to prefer the Flexion image.



(a) Flexion image **(b)** Bearing-Angle image

Fig. 29: *ROC graphs for SIFT*. The ROC plot for SIFT is tightly clustered for both conversions. No distinguished improvements are observable by parameter variations. The unfiltered depth image with default SIFT settings is marked with a horizontal line.

6.1.2 SURF

SURF showed vastly different results then expected. The first observation is that the sheer numbers of detected keypoints with default settings are rendering the keypoint classification results meaningless. After some experimentation with the settings it became clear, that on each octave and every blurring convolution a high number of keypoints are detected redundantly. This finding was true for both the Flexion images and Bearing-Angle images, resulting in a suboptimal keypoint detector performance in scale space. In theory, different keypoint detectors can be combined with different keypoint descriptors and as SURF is commonly referred to as the faster SIFT,

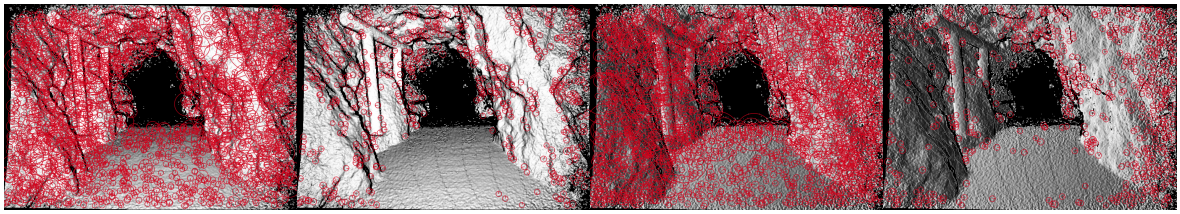
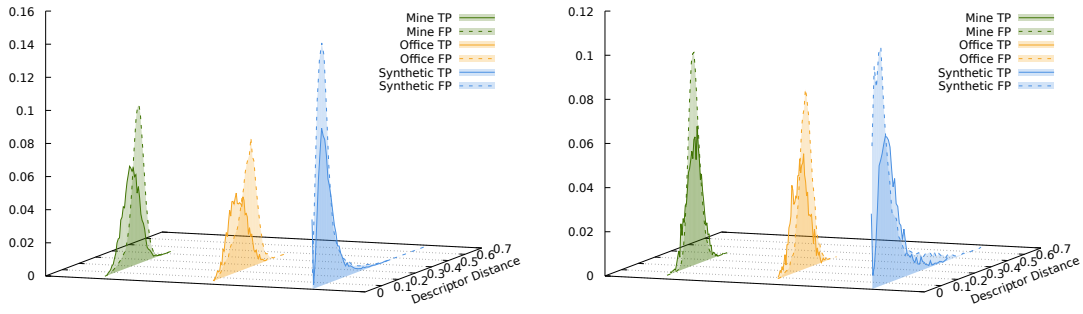


Fig. 30: *SURF keypoints plotted in feature images.* This figure shows the number of keypoints that are detected with SURF in its default configuration compared to just using one octave. Limiting the number of keypoints is mandatory to achieve some potentially meaningful result.

its description potential should still be evaluated. The default settings for SURF are still used, but only one octave with four blurring convolutions detects keypoints (Figure 30), resulting in the keypoint statistics presented in Table 12. The number of correspondences for the adjusted SURF setup is comparable to SIFT detecting keypoints on multiple scales. But the high number of false negatives already indicates lesser descriptor performance. This impression is solidified with the descriptor distance analysis in Figure 31. A slight separation of true and false positives exists for Flexion images, which is not the case for Bearing-Angle images. Even this slight separation is neglected by drastically higher numbers of false positives. Separating both distributions through a maximum matching distance is hard to justify. For a 64 element descriptor that SURF deploys, the match distance around 0.3 seems very low, too. The ROC analysis (Figure 32) points in the same direction showing performance around the *Random Guess* quality, for *Mine* even below. The suspected reason for this

	Synthetic$_{\mathcal{F}}$	**Synthetic**$_{\beta}$	**Mine**$_{\mathcal{F}}$	**Mine**$_{\beta}$	**Office**$_{\mathcal{F}}$	**Office**$_{\beta}$
Keypoint Count	105,788	23,203	440,400	440,400	34,200	34,200
Correspondences	28,836	5,221	23,132	13,846	4,172	2,859
True Positives	14,047	3,186	7,688	2,003	2,131	1,223
False Positives	27,884	6,148	144,506	137,922	10,734	10,632
False Negatives	14,789	2,035	15,444	11,843	2,041	1,636
Precision	0.335	0.341	0.051	0.014	0.166	0.103
Recall	0.487	0.610	0.332	0.145	0.511	0.428
Youden's index	0.123	0.268	−0.015	−0.179	0.146	0.082
Accuracy	0.595	0.647	0.636	0.660	0.620	0.635

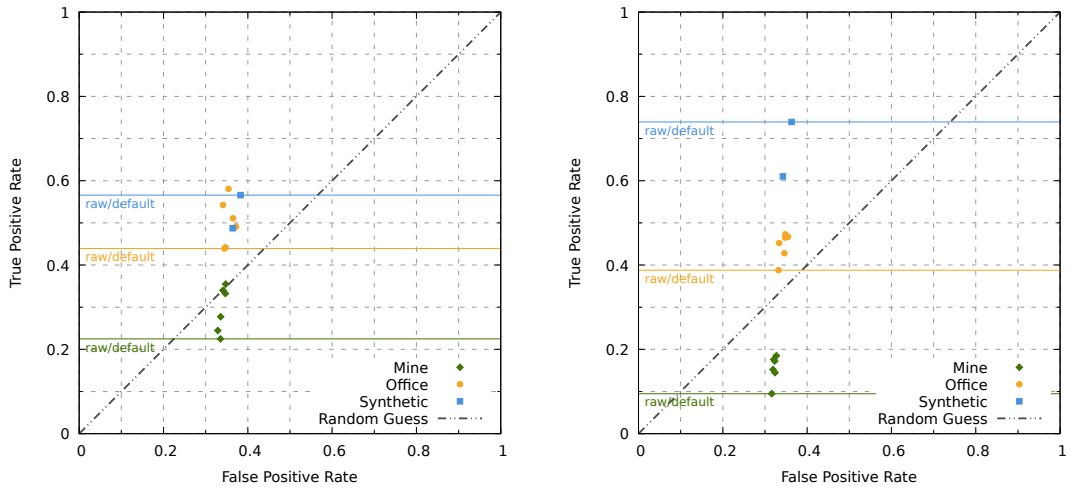
Tab. 12: *Keypoint and matching result for* `SURF/raw/best-only`. The Youden index around 0.1 and even below 0 already indicates bad performance. The number of keypoints is still substantially higher than for SIFT, even with the limit of the keypoint count and usage of only one octave.



(a) Flexion image **(b)** Bearing-Angle image

Fig. 31: *SURF descriptor distances.* The descriptor of SURF does not discriminate regions well. True positives barely separate from false positives.

poor performance of SURF is the use of the Haar-Wavelet response as key metric of the descriptor. The feature image conversions do not result in sharp edges of brightness, as is the case for classical color images and the textures on real world objects. As the feature images result in more subtle gray changes, SURF looses its descriptive power. This evaluation needs to be taken into account when comparing to prior results of Lin et al. [9]. The registration performance use the Bearing-Angle images was inferior to a classical ICP but improved its convergence speed. SURF can produce true keypoint correspondences for some datasets and with the use RANSAC an approximate pose estimation is thinkable. Nonetheless, SURF falls flat compared to



(a) Flexion image **(b)** Bearing-Angle image

Fig. 32: *The ROC graphs for SURF.* No metric for SURF shows any indication of a good classification for correspondences. The ROC graph underlines, that no configurations or data preprocessing substantially improves the situation.

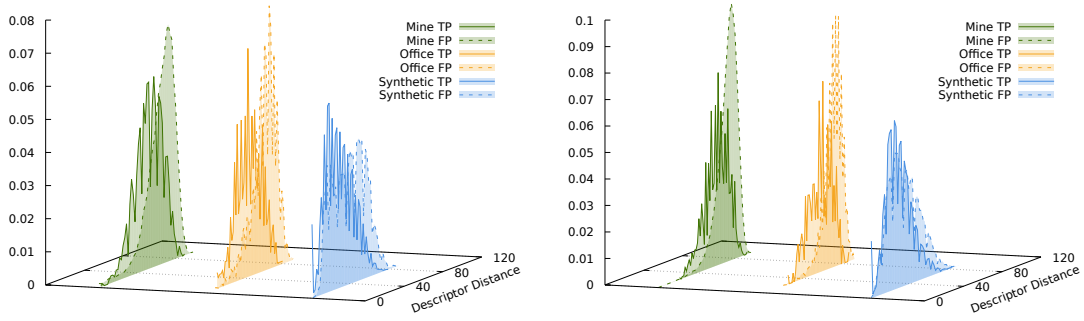
SIFT and should not be used on the proposed feature images. Appendix D.2 presents additional statistics for the SURF keypoints.

6.1.3 ORB

ORB's detector finds less keypoints than SIFT for both Flexion images and Bearing-Angle images. But their size, response and location distribution (Appendix D.3) indicate severe problems. Keypoints are not detected over multiple scales and the response shows a very inconsistent pattern over the different datasets. For *Mine*, the keypoints cluster in the middle and the borders of the field-of-view of the depth sensors. Those regions contains sharp edges from feature image to black because of missing range data. The FAST keypoint detector characteristics of comparing brightness values at different sampling points is likely to blame for the bad detector performance. Similar to SURF's shortcoming, it does not deal well with the smoother gray changes but requires sharp edges. As ORB employs blurring and scale space, too, the effect of missing contrast between sampling points worsens in higher levels of the pyramid, yielding no keypoints there.

	Synthetic$_{\mathcal{F}}$	**Synthetic**$_{\beta}$	**Mine**$_{\mathcal{F}}$	**Mine**$_{\beta}$	**Office**$_{\mathcal{F}}$	**Office**$_{\beta}$
Keypoint Count	52,186	46,459	170,387	162,144	14,051	12,829
Correspondences	17,798	16,593	9,157	6,441	1,921	618
True Positives	10,085	10,203	2,730	1,335	1,246	293
False Positives	11,341	11,559	51,575	58,681	3,889	4,498
False Negatives	7,713	6,390	6,427	5,106	675	325
Precision	0.471	0.469	0.050	0.022	0.243	0.061
Recall	0.567	0.615	0.298	0.207	0.649	0.474
Youden's index	0.234	0.225	−0.022	−0.170	0.322	0.099
Accuracy	0.633	0.612	0.659	0.606	0.670	0.617

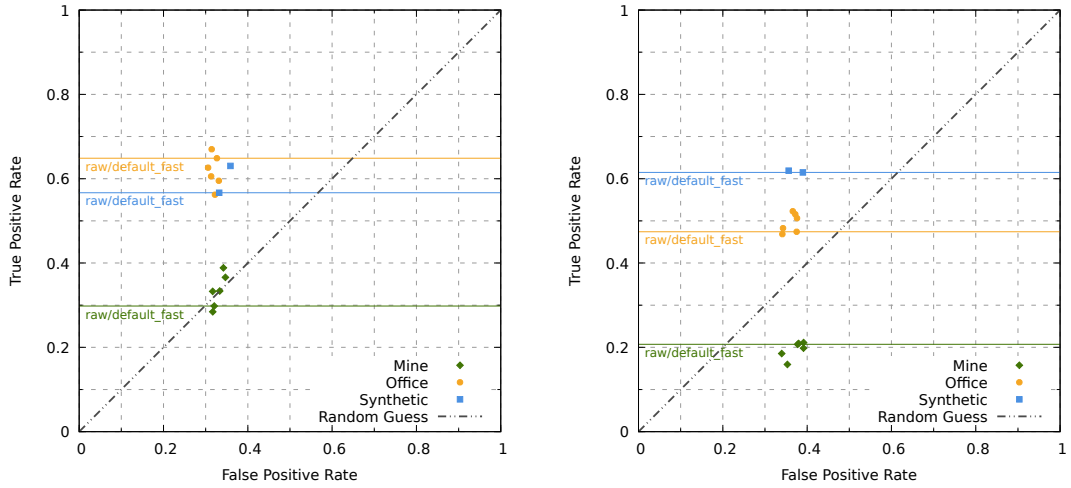
Tab. 13: *Keypoint and matching results for* `ORB/raw/default`. The high number of false negatives and the low Youden index are an indicator for problems of the algorithm. The amount of additional keypoints for Flexion images is lower than for SIFT.



(a) Flexion image **(b)** Bearing-Angle image

Fig. 33: *ORB descriptor distances.* The results show a slight separation of true from false positives. The binary descriptor compares brightness values on specific sampling points. This mechanism is different from SIFT or SURF and seemingly not as robust as SIFT.

The descriptor distance in Figure 33 show some separation between true and false positives. This is better than SURF, but worse than SIFT. The mixed results are reflected in ROC space (Figure 34) by the random performance for *Mine* and OK performance for *Synthetic* and *Office*. Everything considered, ORB is not suitable for the proposed feature images.



(a) Flexion image **(b)** Bearing-Angle image

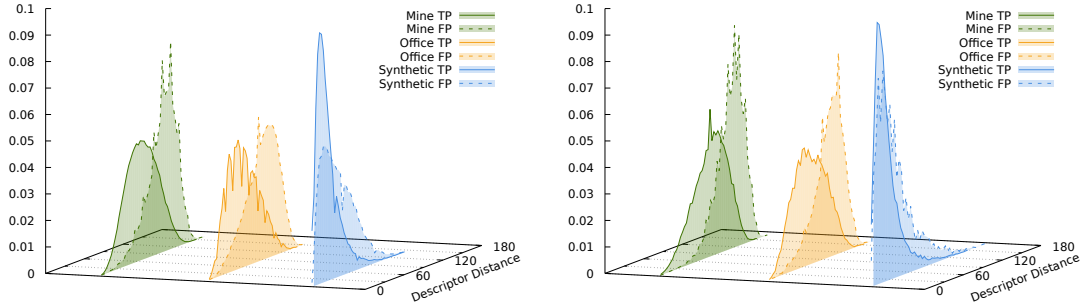
Fig. 34: *ROC graphs for ORB.* The results for ORB are mixed in the ROC graph. It can not be rejected based on their findings, but additional characteristics of the keypoints and their distribution are more important factors for this decision.

6.1.4 AKAZE

	$\text{Synthetic}_{\mathcal{F}}$	Synthetic_{β}	$\text{Mine}_{\mathcal{F}}$	Mine_{β}	$\text{Office}_{\mathcal{F}}$	Office_{β}
Keypoint Count	217,505	44,221	2,168,190	2,198,592	171,000	171,000
Correspondences	96,371	26,455	297,949	222,495	43,037	36,309
True Positives	76,603	22,179	110,339	35,387	24,025	15,455
False Positives	59,032	8,865	648,603	651,091	48,868	48,697
False Negatives	19,768	4,276	187,610	187,108	19,012	20,854
Precision	0.565	0.714	0.145	0.052	0.330	0.241
Recall	0.795	0.838	0.370	0.159	0.558	0.426
Youden's index	0.304	0.336	0.023	−0.171	0.167	0.056
Accuracy	0.636	0.702	0.614	0.618	0.596	0.586

Tab. 14: *Keypoint and matching results for `AKAZE/raw/default`.* AKAZE achieves better results than SURF and ORB. The number of correspondences is very high compared to SIFT, but true positives are still higher than false negatives. This indicates reasonable descriptor performance and stability of the keypoint detector.

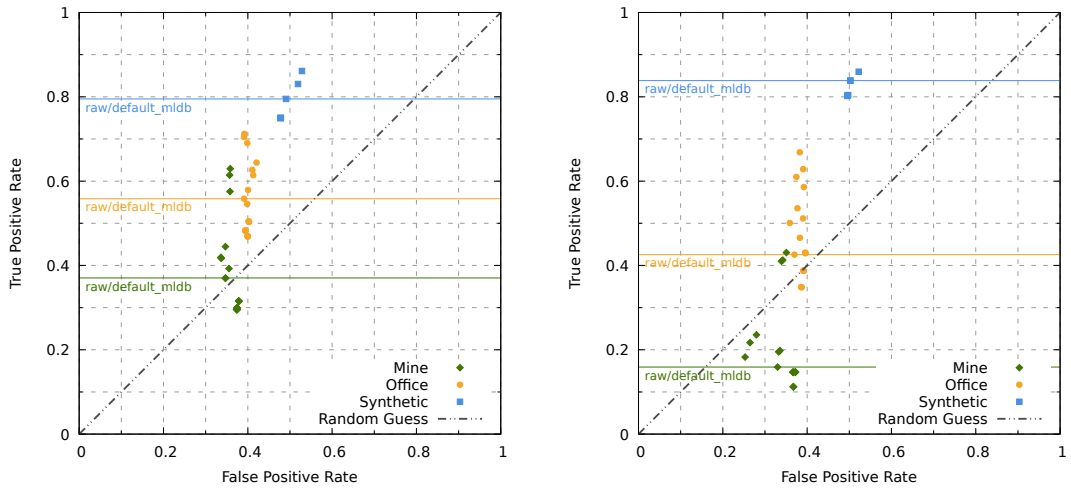
The last evaluated feature detection algorithm is AKAZE. Its keypoint detector design is again closer to SIFT. AKAZE results in approximately one order of magnitude more detected keypoints and correspondences compared to SIFT. Its keypoint characteristics show expected results (Appendix D.4). The detection does not result in as similar diverse keypoint sizes and the response is narrower with a stronger skew towards small responses compared to SIFT. The M-LDB descriptor shows separation between true and false positives across the datasets and feature images (Figure 35). This indicates good performance of the binary descriptor. Additionally it reaffirms the finding of ORB's BRIEF descriptor demonstrating separation potential. The en-



(a) Flexion images **(b)** Bearing-Angle images

Fig. 35: *AKAZE descriptors distances.* The true and false positives are partially separable by the descriptor distance. The Flexion image shows better separation than the Bearing-Angle image. The descriptor is binary and similar to ORB's descriptor, but the improvements show measurable effects.

hancement of M-LDB solidifies the tendency though. The ROC graph in Figure 36 shows that AKAZE can perform quiet reasonable across all datasets. There exists a configuration for each dataset with significantly higher true than false positive rate for the Flexion images. The top performers for each dataset are the configurations using only the best 400 keypoints for matching, regardless of filtering. Applying additional heuristics in matching, like a maximum matching distance, will certainly improve the ratio of true to false positives further. The backprojections in Appendix C.2 show the higher amount of keypoints compared to SIFT. But bad scale coverage leads to unsatisfying keypoint coverage for the *Laserscan* dataset. There, SIFT has clear qualitative advantages over AKAZE, because its keypoints cover the whole scan.



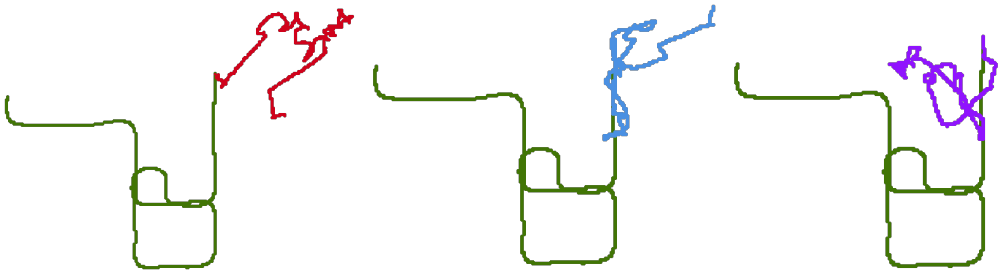
(a) Flexion image **(b)** Bearing-Angle image

Fig. 36: *ROC graphs for AKAZE.* AKAZE results in a bigger spread of the algorithm performance depending on the configuration. The Bearing-Angle image falls behind the Flexion image. The higher scoring configurations apply filtering of the keypoints by their response.

6.1.5 Odometry

The final application of the feature detection and matching algorithms is the computation of a visual odometry. Only the promising candidates SIFT and AKAZE are compared. Both feature images and the color images are used separately to determine the visual odometry for a given trajectory, the ground truth.

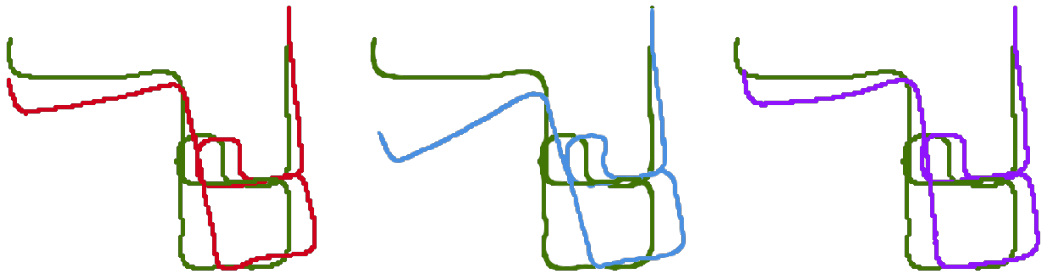
SIFT neither results in a good odometry for color images nor for the feature images. Figure 37 shows random turns and barely straight trajectories for any conversion.



(a) Flexion images **(b)** Bearing-Angle images **(c)** Color Images

Fig. 37: *SIFT odometry trajectories compared.* The presented odometry trajectories for the benchmark dataset from Zhang et al. [87] are calculated with an in-house implementation of a basic visual odometry system using SIFT.

AKAZE on the other hand performs much better in all cases, as can be seen in Figure 38. The principal shape of the trajectory is reconstructed well. The Bearing-Angle images have higher drift compared to the Flexion images or colored images, but only in certain areas where all inputs suffer from similar issues. The findings show that the higher number of keypoints results in better performance. Because the benchmark dataset is synthetic, the colored image do not contain the same level of texture as in real life. This explains the bad performance of SIFT for the colored images. The range data conversions on the other hand are representative for real world cases. A higher number of keypoints, combined with false positive rejection through RANSAC are enough to yield odometry performance similar to colored images.



(a) Flexion images **(b)** Bearing-Angle images **(c)** Color Images

Fig. 38: *AKAZE odometry trajectories compared.* The presented odometry trajectories for the benchmark dataset from Zhang et al. [87] are calculated with an in-house implementation of a basic visual odometry system using AKAZE.

6.2 Keypoint Detector Discussion

The presented results for the different state-of-the-art keypoint detectors imply that good performance on classical camera footage does not automatically yield good performance in the proposed feature images. Detectors that solely work on corner detection, as ORB does with FAST or the Harris-Corner-Detector, have a hard time on the feature images. The experiments on real sensor data show, that they strongly respond to missing measurements, that create a sharp black to gray edge. Solely relying on such areas is unstable and undesirable. Flat out banning such keypoints on the other hand might not be the right choice either. Shadow effects and reflective properties of surfaces can produce missing range values, too. Areas with such consistent black patches can and should be exploited as feature.

SURF's results are the most surprising, especially in comparison to SIFT. SURF is designed as a faster SIFT, using the same principles and underlying theory. One reason for the bad results might be too many simplifications and approximations. The use of a box-filter compared to a Gaussian in combination with the other assumptions are justified with empirical analysis on colored images. In these cases SURF performs good enough, but the bigger deviation from the theoretical background and human perception compared to SIFT seem to hurt effectiveness in the novel feature images. After all, SURF does find true correspondences in some instances.

SIFT and AKAZE provide the most effective keypoint detectors tested. AKAZE's keypoint sizes are not as diverse as SIFT's, but the absolute number of keypoints is a clear advantage for example in visual odometry tasks. The surprisingly low number of AKAZE keypoints in the *Laserscan* (Appendix C.2) could be due to a bad implementation regarding image scaling of very wide images. This effect requires further research.

6.3 Feature-Descriptor Discussion

SIFT again proves as a very good descriptor that has over 80% recall in even challenging conditions. This unmet performance comes at the cost of high computational complexity and memory footprint, simply recreating some of the reasons why alterna-

tives to SIFT were developed in classical computer vision. Still, for global place recognition, loop closure and other tasks requiring such descriptive power, SIFT seems to be the first choice for the feature images. The odometry results are a clear indication, that SIFT's keypoint detector does not produce enough features, at least for classical pinhole cameras and geometrically clean looking scenes.

The very similar descriptors M-LDB and BRIEF, used in AKAZE and ORB respectively, proof their potential, too. They function very different compared to SIFT's brightness histograms. The brightness sampling around the keypoint and construction of the binary descriptor is effective. Using an averaged value for the area around the sampling point improves the robustness of M-LDB. The combination of SIFT keypoint detector and descriptor has a slightly lower false positive rate compared to AKAZE's detector and M-LDB, but the difference in the number of keypoints needs to be considered, too. The much lesser computational complexity and drastically higher matching speed of M-LDB is promising for real-time applications, like visual odometry.

The Haar-Wavelet response-based descriptor SURF falls flat for both Flexion images and Bearing-Angle images. Similar to the FAST detector, the lack of strong edges typically induced by texture in color images seems to be the main reason for this shortcoming. Another indication for this finding is, that KAZE has similar Random-Guess performance like SURF. KAZE's descriptor is not discussed further, but part of the AKAZE tests and is based on the SURF descriptor.

It is interesting to see, that the false positive rate is around 35%$\pm$5% for all evaluated descriptors. This can be partly explained by inherent similarity of objects in the scenes. As only the geometric shape serves as information source, similarity between boxes and edges will always exist. The more nuanced shading details, complex textures and color differences from color images can not influence the result. An additional factor to such a similar false positive rate is the rudimentary matching scheme applied. A reasonable maximum matching distance and additional heuristics in the matching system will produce less false positives. Such heuristics do not exist yet for the proposed feature images and only this evaluation brings an empirical foundation for such heuristics.

6.4 Error Discussion

The datasets are affected by different errors and simplifications, that are discussed in this section. *Synthetic* is a noise free dataset with very simple scene structure. Findings and conclusions based on this dataset are only of principle nature but do not reflect real world performance.

The *Mine* and *Office* datasets are both gathered using the same Kinectv2 technology, but under different conditions and environments. *Mine*, an underground mining environment with partially wet walls, contains drastically more missing depth measurements. For parts of the whole trajectory, it is very challenging to make out the basic shapes of the scene as a human. The computer vision algorithms can not magically perform better at these places. Parts of the scene are not within the measurement range, the walls and wide areas are not fully lit by the sensor's infrared LED or the light is absorbed by water. Missing range measurements also occur in *Office*, but the overall influence of bad conditions is not as high as in *Mine*.

Another major consideration is the rectification of the depth images before being processed by the proposed pipeline. The native resolution of the Kinectv2 depth sensor is 512×424 px [30], which is lower than the input depth images used for the feature image conversions. This change in resolution is artifact of ROS' image rectification, that maps each depth pixel to its matching color pixel. The experiments show no qualitative difference of results between different image scales, but unrectified depth images experience distortion which will change the results at the image borders. The sensors depth image is noticeably affected by smaller waves in the conversion images for stand images. Their effect is considered small, because different filtering algorithms show only slight changes in the final result. The much finer resolution of the *Laserscan* dataset on the other hand showed that prior filtering with median blur can improve the quality of the final feature image drastically (Appendix B.3).

There are no errors expected from the implementation of the software as all components are thoroughly tested. The usage of integers to store the range information on the other hand creates visible artifacts. Their effect is neglectable because every feature algorithm employs some form of blurring and detail reduction.

7 Conclusion and Future Work

This thesis evaluated the use of state-of-the-art computer vision feature detection and description algorithms to recognize salient points in a depth image. Instead of direct application of the algorithm to the depth image, it is transformed into a derived image constructed from local geometric properties of the depth values. The approach is introduced by Scaramuzza et al. [8] with their Bearing-Angle image and the usage of SURF to detect keypoints on those by Lin et al. [9]. To overcome the Bearing-Angle image's limitation, the lack of rotation invariance, this work proposed four additional feature image conversions. Only the Flexion image is judged as viable. The performance of SIFT, SURF, ORB and AKAZE were evaluated for both the Bearing-Angle and the Flexion image. Surprisingly, the evaluation showed that SURF lacks descriptive power to achieve good recognition performance. ORB's corner-based keypoint detector failed to produce stable keypoints. SIFT and AKAZE were the only algorithms achieving acceptable detection and recognition performance. The findings of this thesis — the novel Flexion image and its performance for SIFT and AKAZE — may be the foundation for a descriptor for dense, structured pointcloud data, a problem not solved satisfactory yet.

The proposed conversion has been tested with a Kinectv2 depth sensor and the Riegl Z300 LiDAR scanner. Preprocessing the depth data with edge-preserving filtering was part of the evaluation and has proven to be useful for the Riegl Z300, but unnecessary for the Kinectv2 sensor. Applying a median blur is enough for consistent smooth results and a more advanced bilateral filtering was not beneficial.

The implementation of the evaluation software and conversion code was done with state-of-the-art software engineering principles ensuring high quality. It can be used for follow up research and various binary artifacts are produced that run on any Linux system and can be compiled from source if required. The proposed Flexion image bridges the gap between pointcloud processing and classical computer vision algorithms. Common processes, such as visual odometry, global localisation and maybe even optical flow algorithms should be evaluated. The foundational insights on the descriptor performance of SIFT and AKAZE help to fine-tune heuristics, such

as the maximum matching distance, for these tasks. The feature image conversions implementations can be optimized by better SIMD support and caching intermediate results like the spherical coordinates for pixels. Each conversion is suited for GP-GPU computation and drastic speed-ups are expected.

New formulations for visual odometry and relative pose calculation become possible because the range value for each pixel is available. This additional information reduces the required number of true correspondences to reconstruct the relative pose. Investigating the viability of a new formulation for this problem and potential speed and efficiency gains seems to be the highest value research target. The classical ICP formulation relies on establishing explicit correspondences between points, too. Using the Flexion image and classical feature descriptors might result in a new variation of this algorithm. Both problems are related and need further research.

A Derivation of the Bearing-Angle Formula

This section provides the exact derivation of the Bearing-Angle formula. The literature on Bearing-Angle disagrees on the formula and have mathematical errors. The full chain of derivation and graphical presentations goal is to provide a trustworthy and checkable background for the provided formula.

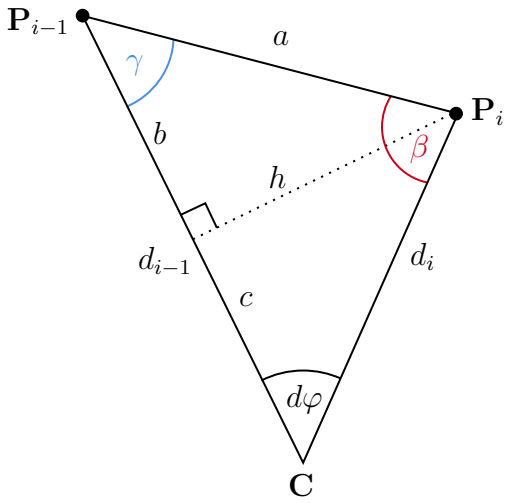


Fig. A39: *Derivation of the Bearing-Angle with all support variables.* Deriving the bearing angle requires additional support variables that are introduced with this figure. Using fundamental trigonometry allows derivation of the formula.

The cosine theorem holds true for all triangles and is the central equality for the derivation.

Ansatz:

$$d_{i-1}^2 = d_i^2 + a^2 - 2d_i a \cos\beta \qquad \text{(Cosine Theorem)} \tag{36}$$

Introducing supporting variables at the positions given in Figure A39 allows substituting the missing quantity a.

$$\begin{aligned}
h &= d_i \sin d\varphi \\
c &= d_i \cos d\varphi \\
b &= d_{i-1} - d_i \cos d\varphi \\
a^2 &= h^2 + b^2 \\
a^2 &= (d_i \sin d\varphi)^2 + (d_{i-1} - d_i \cos d\varphi)^2 \\
a^2 &= d_i^2 \sin^2 d\varphi + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi + d_i^2 \cos^2 d\varphi \\
a^2 &= d_{i-1}^2 \sin^2 d\varphi + d_{i-i}^2 \cos^2 d\varphi + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi \\
a^2 &= d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi
\end{aligned} \tag{37}$$

For a holds the condition $a > 0$ for all values, because the constraints $d_i > 0$, $d_{i-1} > 0$ and $0 < d\varphi < \pi$ result from the properties of depth sensors.

The variable a is now substituted in the ansatz. Consequent simplification of the equations yields the final formula for the bearing angle.

$$\begin{aligned}
d_{i-1}^2 &= d_i^2 + a^2 - 2d_i a \cos\beta \\
d_{i-1}^2 &= d_i^2 + d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi - 2d_i\sqrt{d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi}\cos\beta \\
0 &= 2d_i^2 + 0 - 2d_{i-1}d_i \cos d\varphi - 2d_i\sqrt{d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi}\cos\beta \\
0 &= d_i^2 - d_{i-1}d_i \cos d\varphi - d_i\sqrt{d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi}\cos\beta \\
0 &= d_i - 1d_{i-1} \cos d\varphi - 1\sqrt{d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi}\cos\beta \\
\cos\beta &= \frac{d_i - d_{i-1}\cos d\varphi}{\sqrt{d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi}} \\
\beta &= \arccos \frac{d_i - d_{i-1}\cos d\varphi}{\sqrt{d_i^2 + d_{i-1}^2 - 2d_{i-1}d_i \cos d\varphi}}
\end{aligned} \tag{38}$$

B Converted Images for Datasets

This section presents exemplaric feature images for each dataset. The effects of filtering and the visual difference between sensors and environments become apparent.

B.1 Synthetic

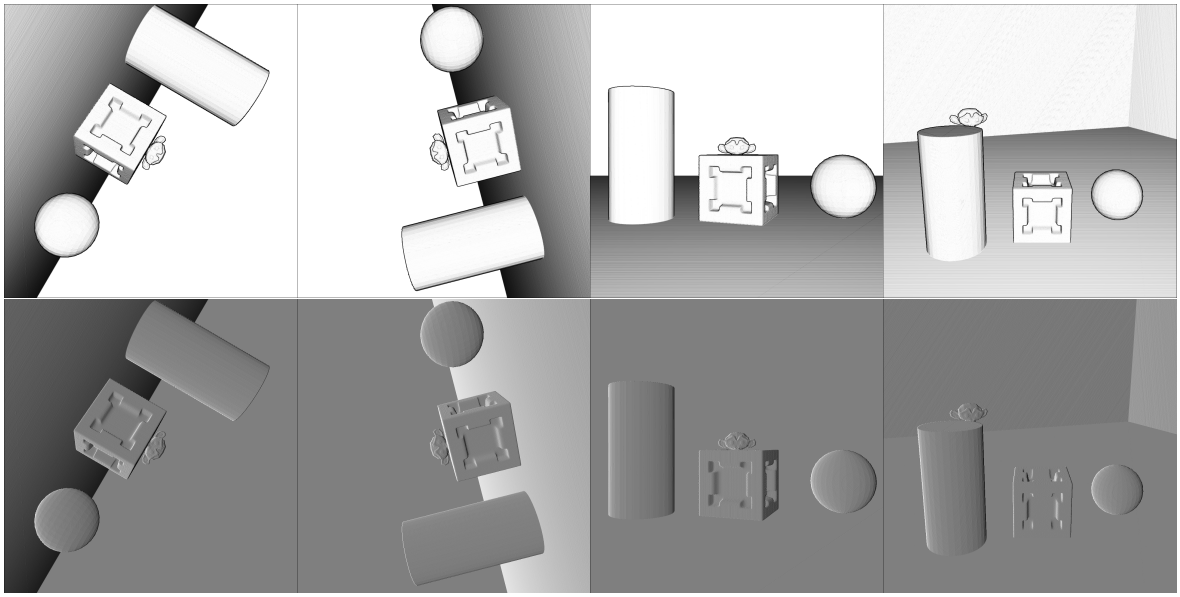


Fig. B40: *Examples of the Synthetic scene feature image conversions.* The *Synthetic* scene's conversion to Flexion images and Bearing-Angle images. No filtering is applied, because the range data has no error.

B.2 Office

Fig. B41: Flexion images and Bearing-Angle images for the *Office* dataset.

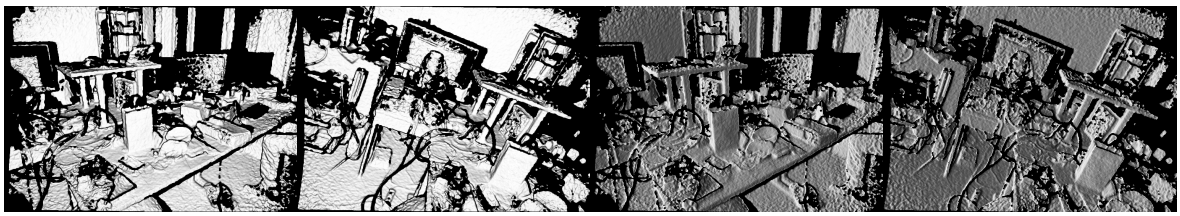


Fig. B42: Effect of median blur filtering on the *Office* data.

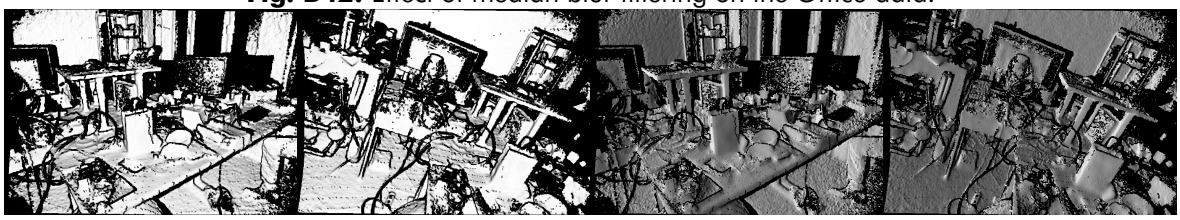


Fig. B43: Effect of bilateral filtering on the *Office* data.

B.3 Laserscan

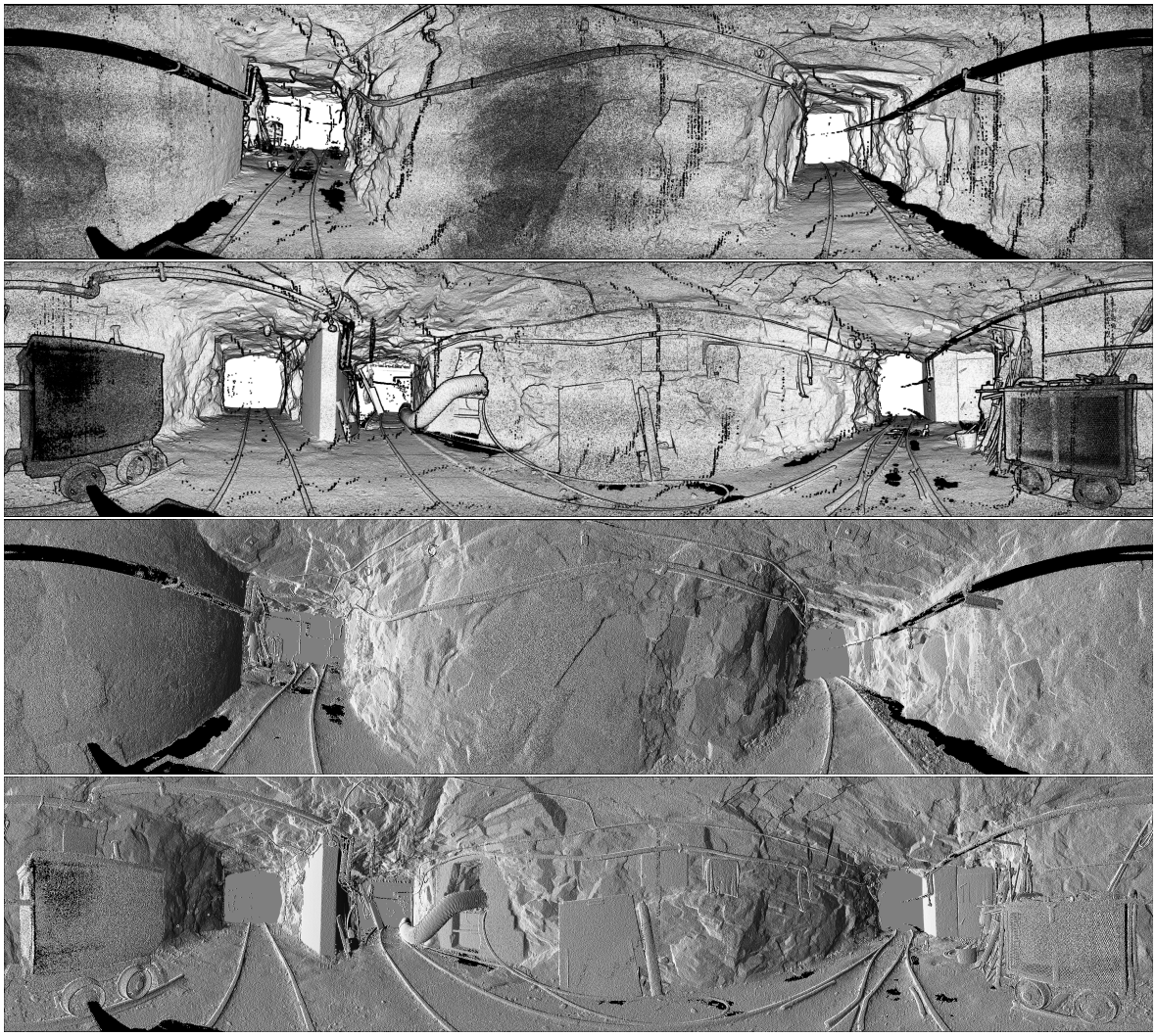


Fig. B44: Example Flexion images and Bearing-Angle images of the Laserscan dataset without any filtering.

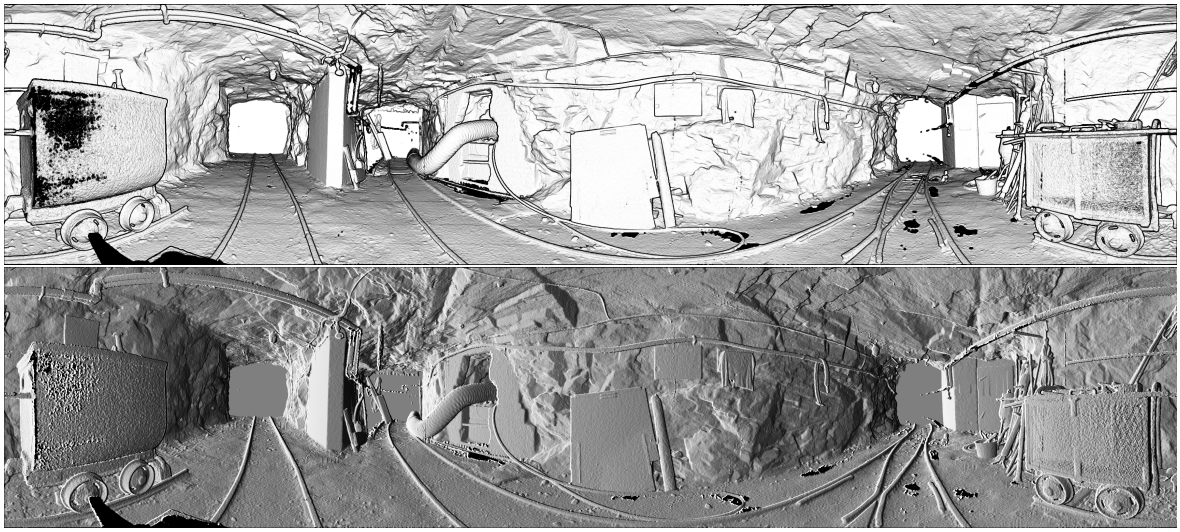


Fig. B45: Effect of median blur filtering on the *Laserscan* data.

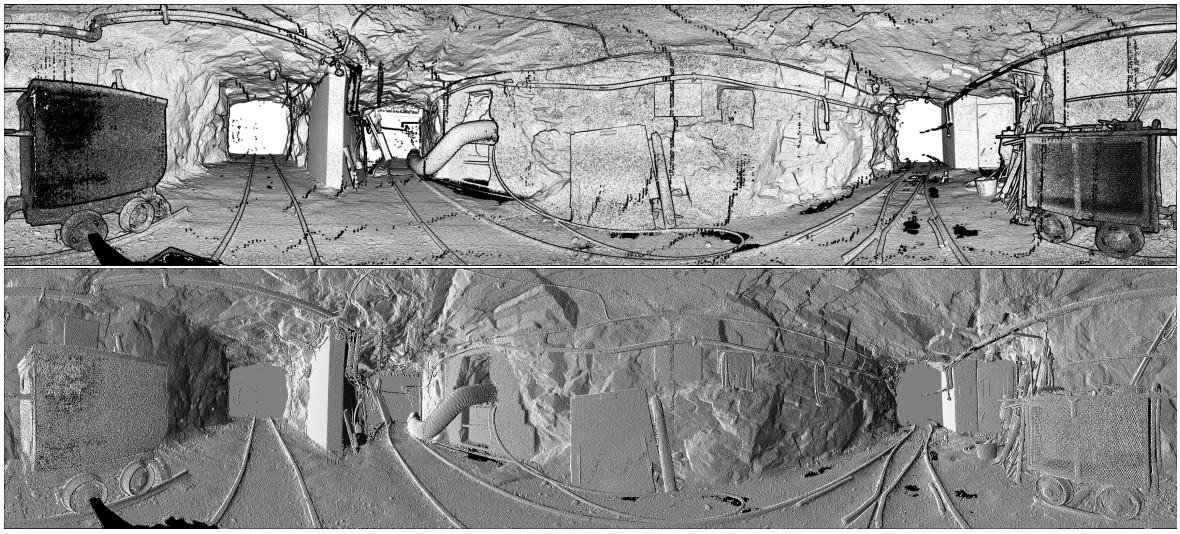


Fig. B46: Effect of bilateral filtering on the *Laserscan* data.

B.4 Odometry Benchmark

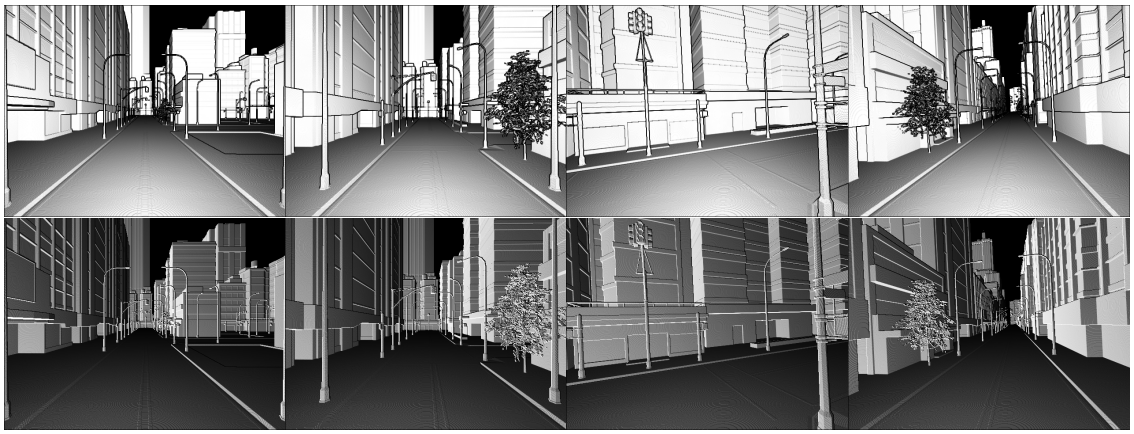


Fig. B47: Flexion images and Bearing-Angle images for the *Odometry* dataset.

B.5 Mine

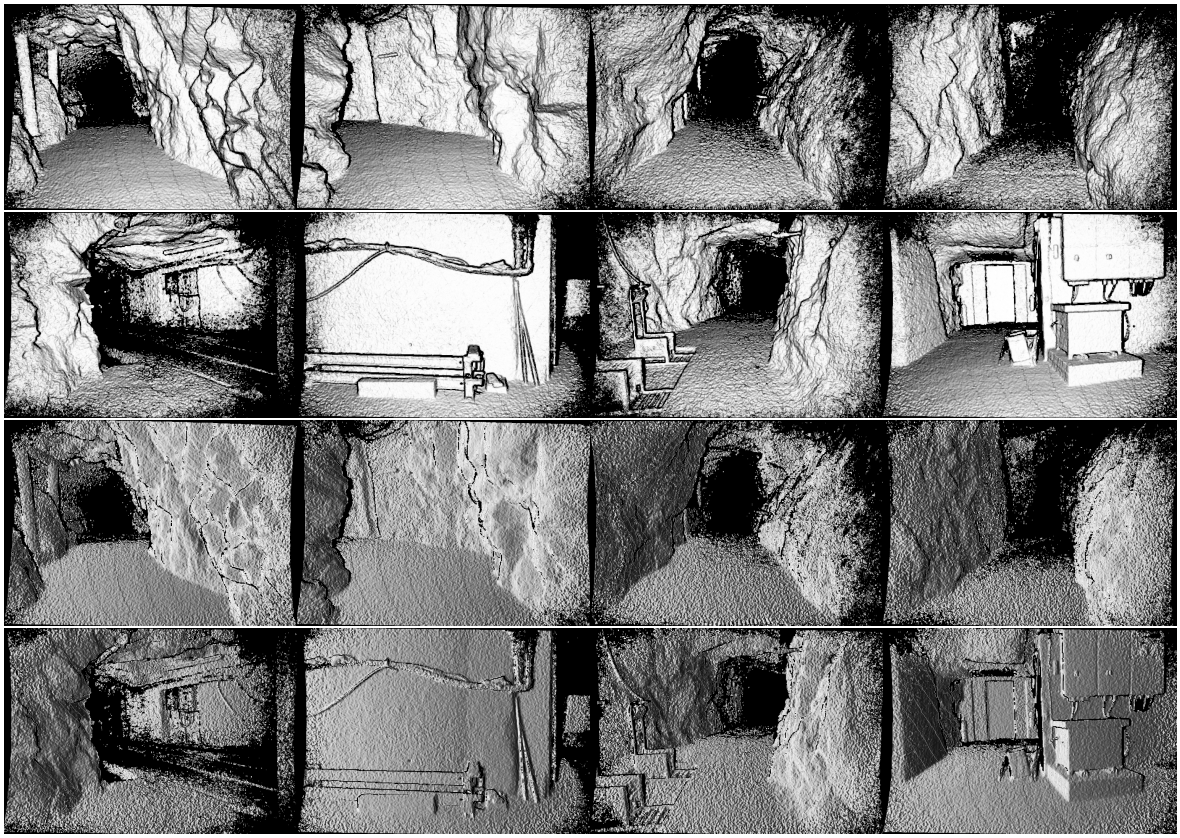


Fig. B48: Example Flexion images and Bearing-Angle images of the *Mine* dataset without any filtering.

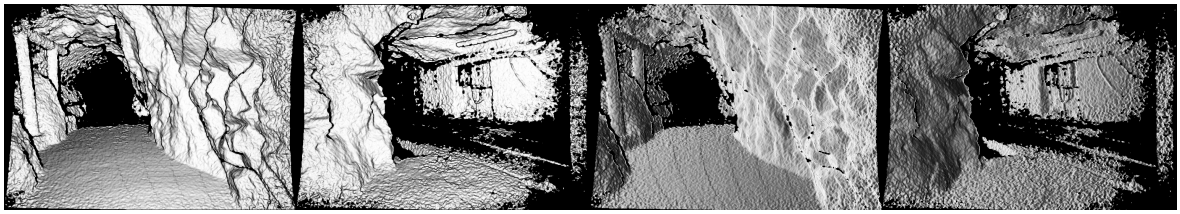


Fig. B49: Effect of median blur filtering on the *Mine* data.

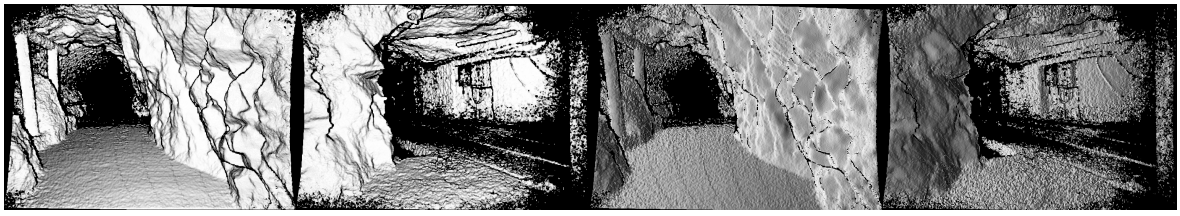


Fig. B50: Effect of bilateral filtering on the *Mine* data.

C Keypoints and Matchings for SIFT and AKAZE

The plots in this section show the combined keypoint and matching performance. Green points are a identified true positives and purple points false negative, making up the total correspondences. Orange lines connect false positives.

C.1 SIFT

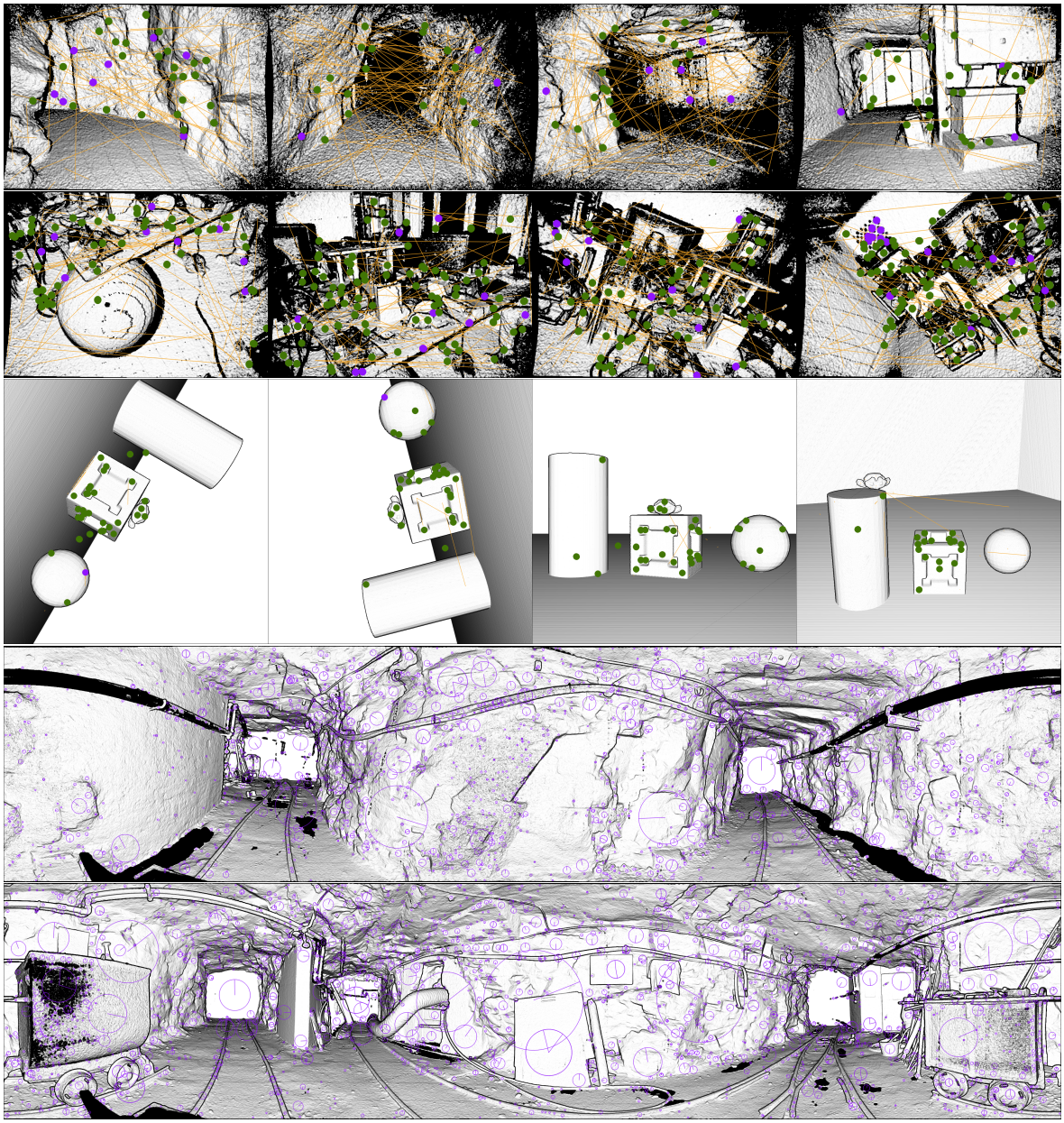


Fig. C51: SIFT backprojections for `raw/default` and keypoints of `median-blur/default` for *Laser-scan*

C.2 AKAZE

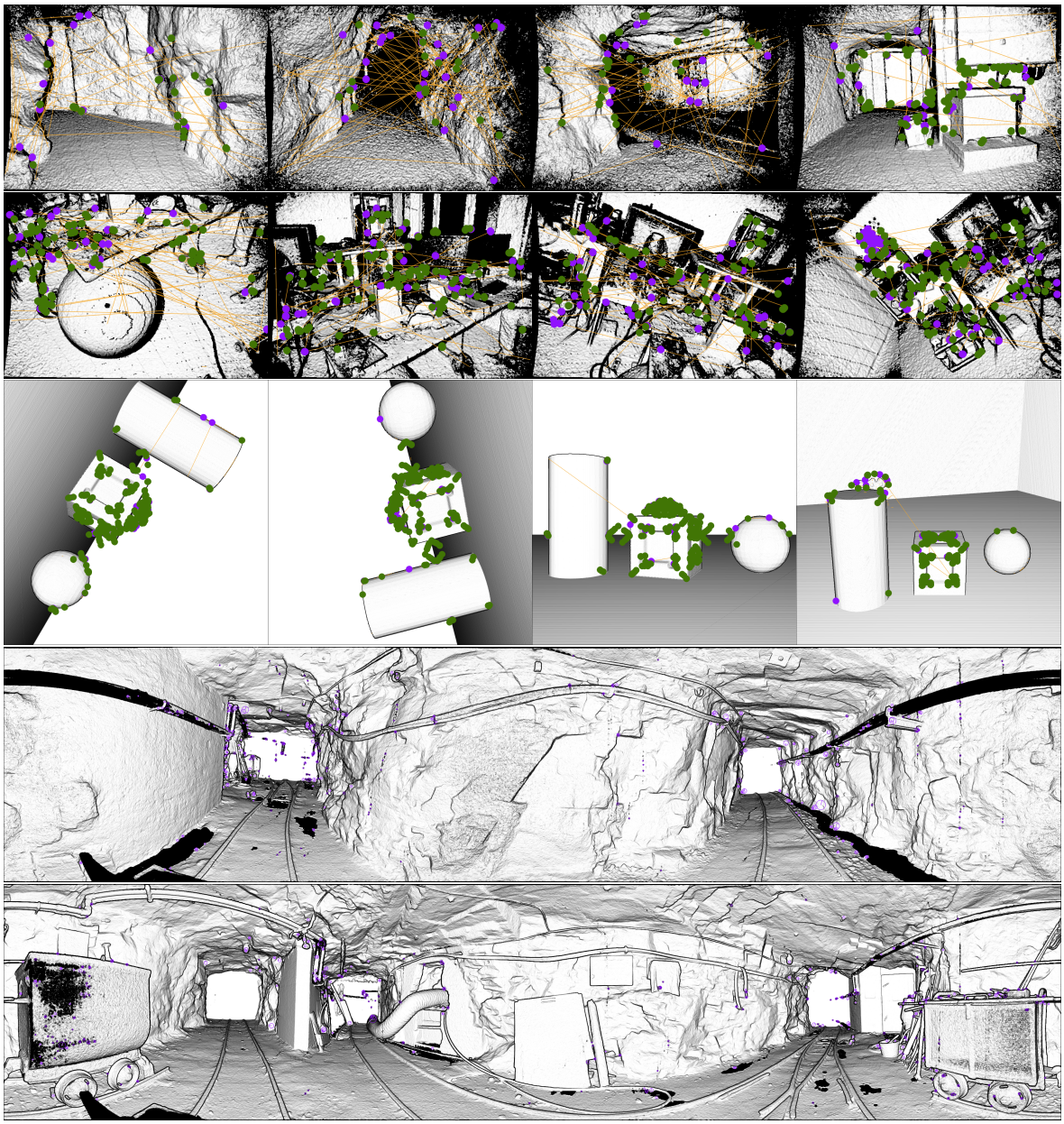
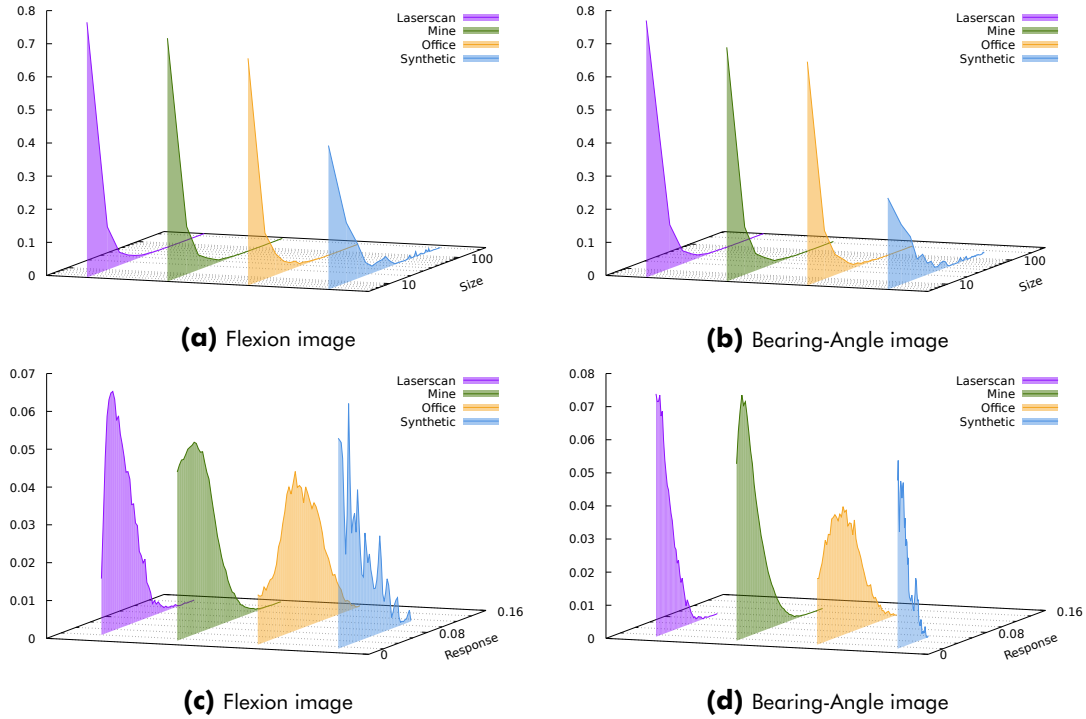


Fig. C52: AKAZE backprojections for `raw/best-only` and keypoints of `median-blur/best-only` for *Laserscan*

D Additional Keypoint Statistics

D.1 SIFT



(a) Flexion image **(b)** Bearing-Angle image

(c) Flexion image **(d)** Bearing-Angle image

Fig. D53: Overview of keypoint sizes and responses for the SIFT detector.

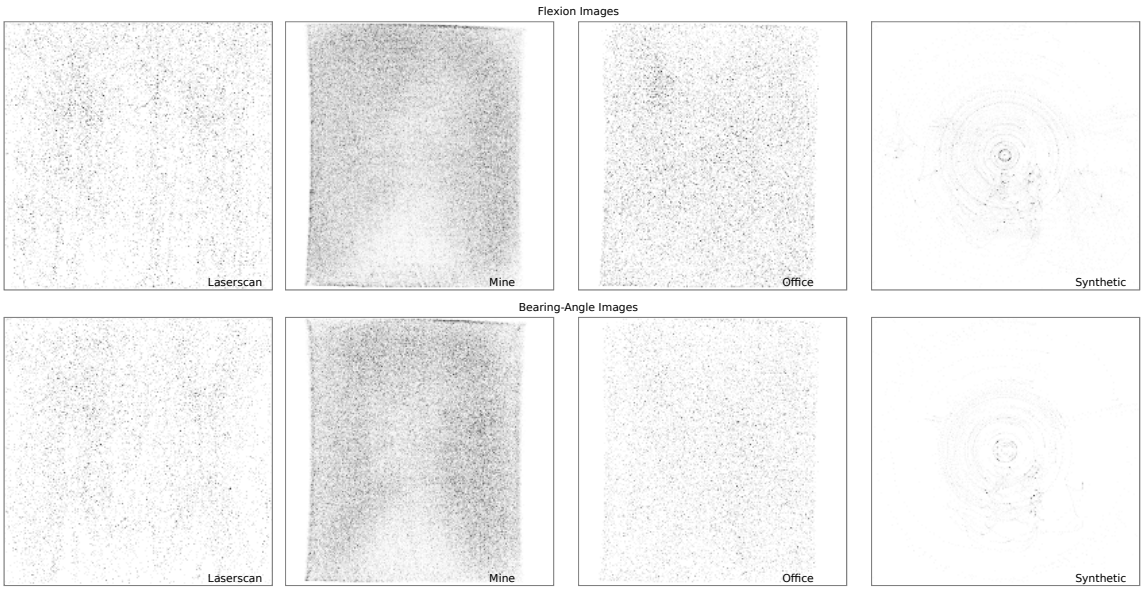
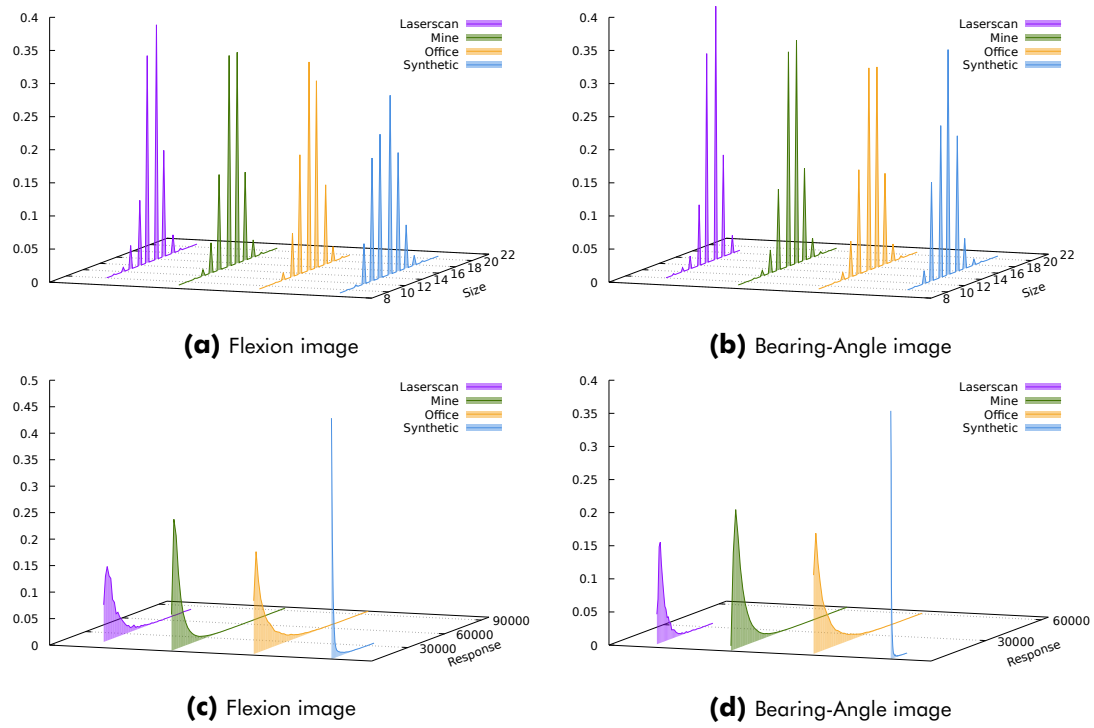


Fig. D54: Keypoint distribution for the SIFT detector.

D.2 SURF



(a) Flexion image **(b)** Bearing-Angle image

(c) Flexion image **(d)** Bearing-Angle image

Fig. D55: Overview of keypoint sizes and responses for the SURF detector.

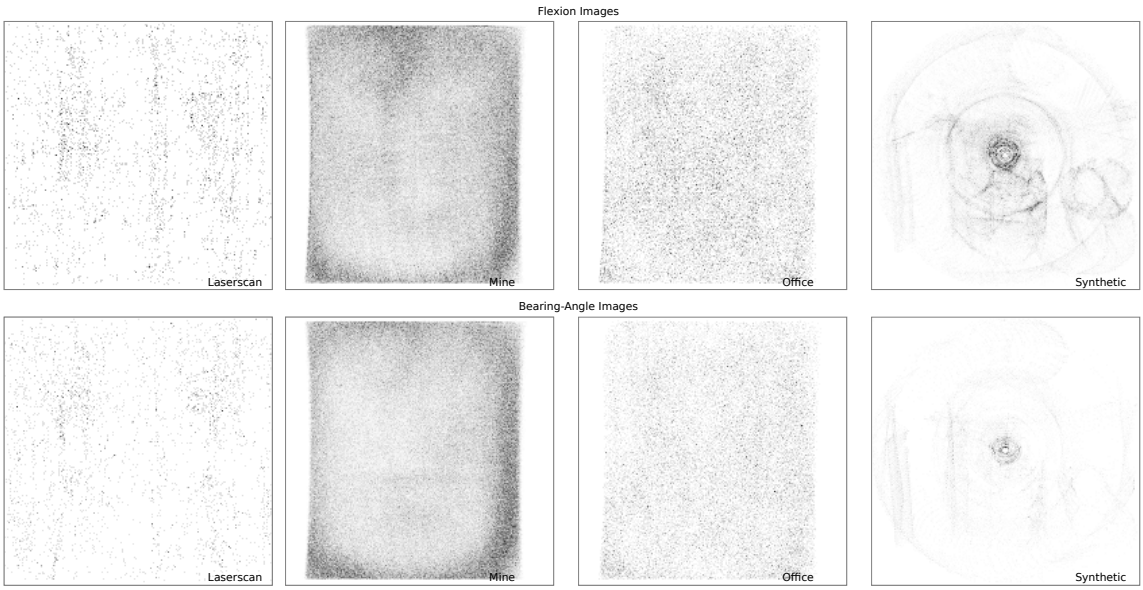
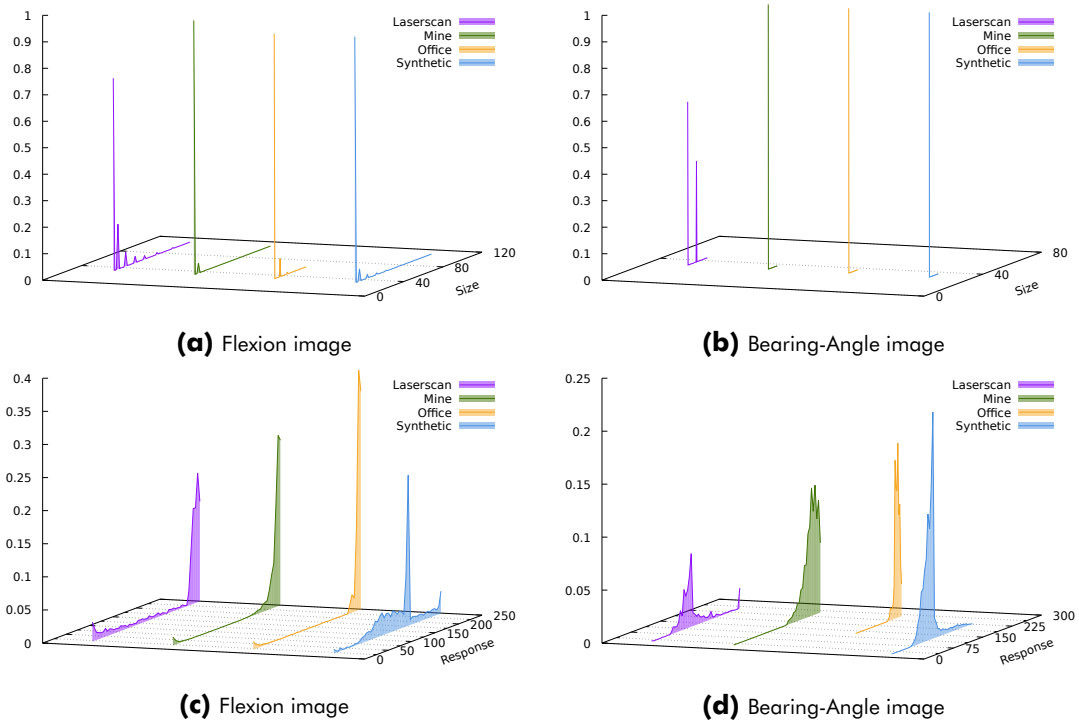


Fig. D56: Keypoint distribution for the SURF detector.

D.3 ORB



(a) Flexion image **(b)** Bearing-Angle image

(c) Flexion image **(d)** Bearing-Angle image

Fig. D57: Overview of keypoint sizes and responses for the ORB detector.

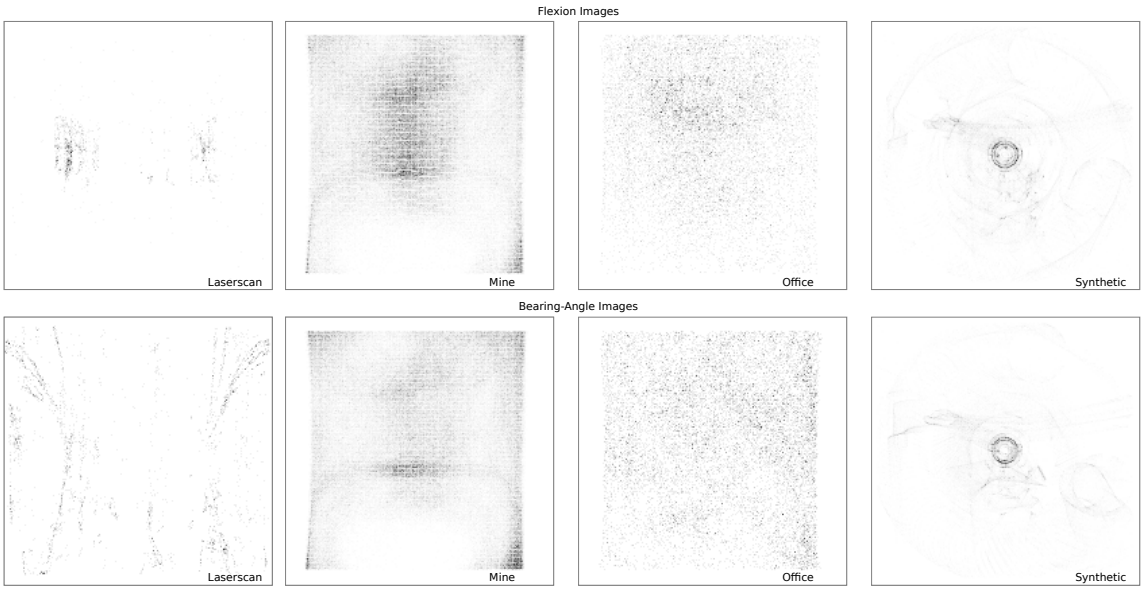
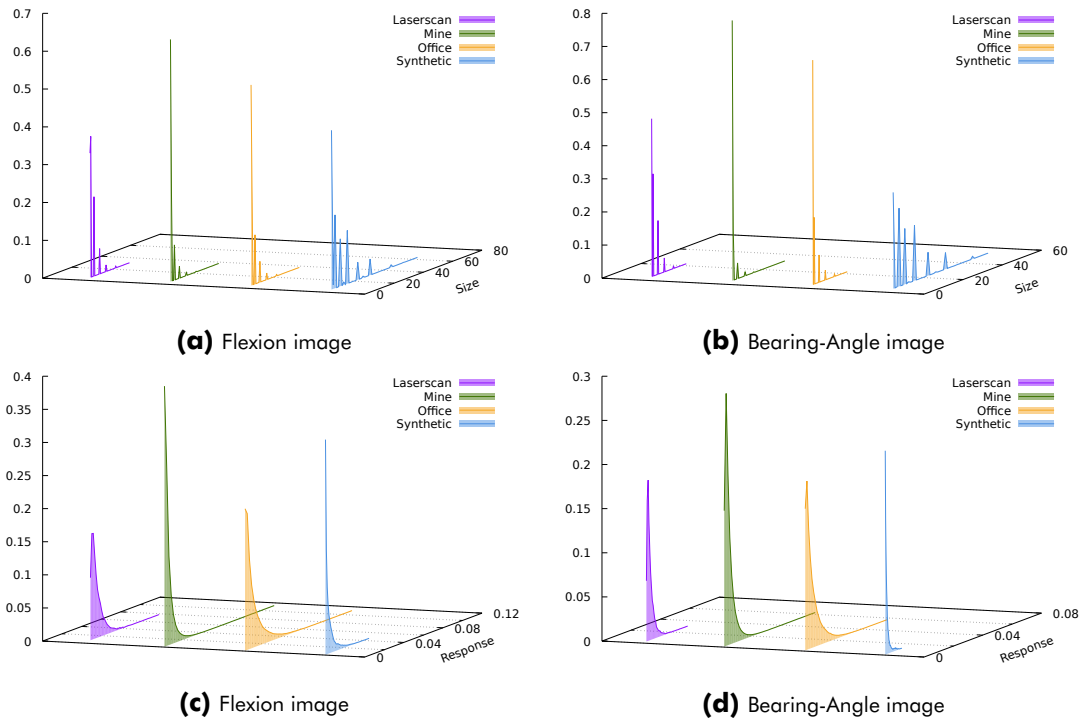


Fig. D58: Keypoint distribution for the ORB detector.

D.4 AKAZE



(a) Flexion image **(b)** Bearing-Angle image

(c) Flexion image **(d)** Bearing-Angle image

Fig. D59: Overview of keypoint sizes and responses for the AKAZE detector.

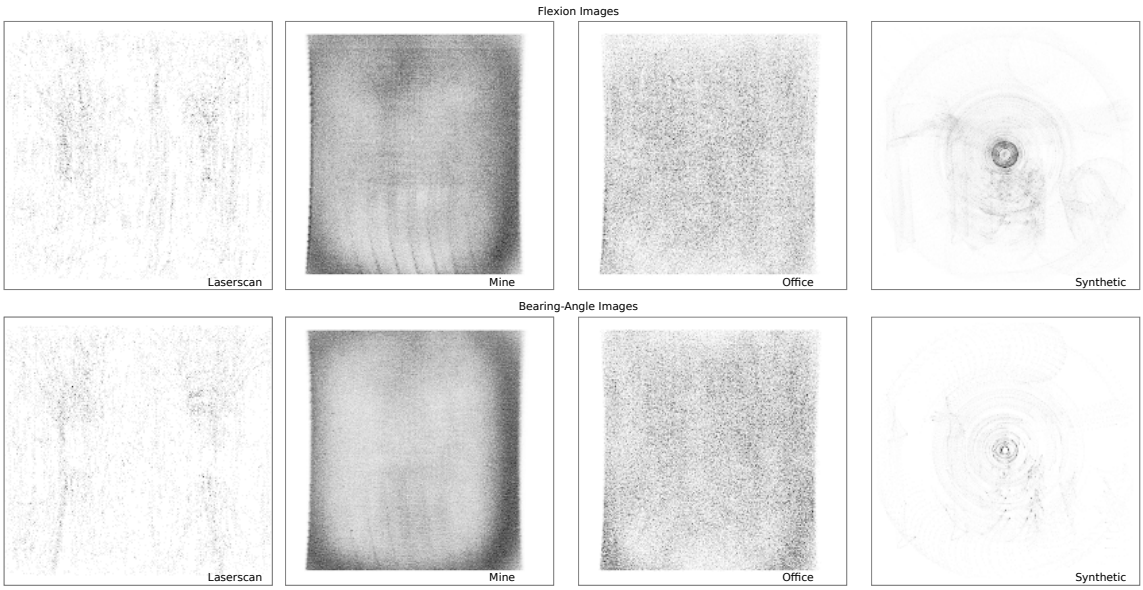


Fig. D60: Keypoint distribution for the AKAZE detector.

References

[1] M. He, C. Zhu, Q. Huang, B. Ren, and J. Liu, "A review of monocular visual odometry," *The Visual Computer*, 06 2019.

[2] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, "Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8601–8610, 06 2018.

[3] P. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

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

[5] K. L. Ho and P. Newman, "Loop closure detection in SLAM by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740 – 749, 2006, selected papers from the 2nd European Conference on Mobile Robots (ECMR '05).

[6] T. Sattler, B. Leibe, and L. Kobbelt, "Towards Fast Image-Based Localization on a City-Scale," *Proceedings of the 15th international conference on Theoretical Foundations of Computer Vision: outdoor and large-scale real-world scene analysis*, pp. 191–211, 06 2011.

[7] O. Andersson and S. R. Marquez, "A comparison of object detection algorithms using unmanipulated testing images: Comparing SIFT, KAZE, AKAZE and ORB," Stockholm, Sweden, 2016.

[8] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4164–4169, 10 2007.

[9] C. C. Lin, Y. C. Tai, J. J. Lee, and Y. S. Chen, "A novel point cloud registration using 2D image features," *Eurasip Journal on Advances in Signal Processing*, vol. 2017, no. 1, 12 2017.

[10] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," *Conference: Proceedings of the 9th European conference on Computer Vision - Volume Part I*, 2006.

[11] Y. Zhuang, N. Jiang, H. Hu, and F. Yan, "3-d-laser-based scene measurement and place recognition for mobile robots in dynamic indoor environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 2, pp. 438–450, 2013.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[13] A. Segal, D. Hähnel, and S. Thrun, "Generalized-icp," *Proc. of Robotics: Science and Systems*, 06 2009.

[14] M. Korn, M. Holzkothen, and J. Pauli, "Color Supported Generalized-ICP," *Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, vol. 3, pp. 592–599, 01 2014.

[15] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2743 – 2748 vol.3, 11 2003.

[16] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[17] B. Bellekens, V. Spruyt, and R. B. Maarten Weyn, *A survey of rigid 3D pointcloud registration algorithms*. IARA, 2014.

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

[19] G. Elbaz, T. Avraham, and A. Fischer, "3D point cloud registration for localization using a deep neural network auto-encoder," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Januar, pp. 2472–2481, 11 2017.

[20] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3d range data based on point features," *2010 IEEE International Conference on Robotics and Automation*, pp. 1400–1405, 2010.

[21] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization," 07 2015.

[22] J. Sivic and A. Zisserman, "Efficient Visual Search of Videos Cast as Text Retrieval," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 591 – 606, 05 2009.

[23] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *SpringerPlus*, vol. 5, 2016.

[24] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, pp. 1615–1630, 11 2005.

[25] C. G. Harris, M. Stephens *et al.*, "A combined corner and edge detector." *Alvey vision conference*, vol. 15, no. 50, pp. 10–5244, 1988.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[27] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *Proceedings of the British Machine Vision Conference*, 2013.

[28] M. Gesto-Diaz, F. Tombari, D. Gonzalez-Aguilera, L. Lopez-Fernandez, and P. Rodriguez-Gonzalvez, "Feature matching evaluation for multimodal corre-

spondence," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 129, pp. 179–188, 07 2017.

[29] F. Blais, "Review of 20 years of range sensor development," *Videometrics VII*, vol. 5013, pp. 62 – 76, 2003.

[30] O. Wasenmüller and D. Stricker, "Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision," *Asian Conference on Computer Vision Workshop*, 11 2016.

[31] Intel, "Intel RealSense," https://intelrealsense.com, Accessed: 2020-04-05.

[32] Y. Zhou, D. Zhao, Y. Yu, J. Yuan, and S. Du, "Adaptive color calibration based one-shot structured light system." *Sensors 12, no. 8*, vol. 12(8), pp. 10 947–10 963, 2012.

[33] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time-of-Flight Cameras: Principles, Methods and Applications*. London: Springer, 2012.

[34] P. Frydlewicz, "LiDAR and ToF Cameras – Technologies explained," https://tof-insights.com/time-of-flight-lidar-and-scanners-technologies-explained/, 11 2018, Accessed: 2019-03-27.

[35] T. Taylor, *Introduction to Laser Science and Engineering*. CRC Press, 2020.

[36] D. J. Natale, R. L. Tutwiler, M. S. Baran, and J. R. Durkin, "Using full motion 3d flash lidar video for target detection, segmentation, and tracking," *2010 IEEE Southwest Symposium on Image Analysis & Interpretation (SSIAI)*, pp. 21–24, 2010.

[37] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*, 2nd ed. Springer, 2011.

[38] OpenCV Calibration Toolbox, "Pinhole Camera Model," https://docs.opencv.org/4.3.0/pinhole_camera_model.png, Accessed: 2020-04-07.

[39] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[40] J. P. Snyder, "Map projections: A working manual," Washington, D.C., 1987.

[41] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[42] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, 06 1981.

[43] T. Sattler, B. Leibe, and L. Kobbelt, "SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter," *2009 IEEE 12th International Conference on Computer Vision*, pp. 2090–2097, 2009.

[44] O. Chum and J. Matas, "Matching with PROSAC - progressive sample consensus," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 220–226 vol. 1, 2005.

[45] A. Agresti, *Introduction to Categorical Data Analysis*, 2nd ed. Hoboken: Wiley, 2007.

[46] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, 2006.

[47] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, pp. 32–35, 1950.

[48] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[49] B. R. Frieden, "A new restoring algorithm for the preferential enhancement of edge gradients," *JOSA*, vol. 66, no. 3, pp. 280–283, 1976.

[50] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 1, pp. 13–18, 1979.

[51] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *Sixth International Conference on Computer Vision*, pp. 839–846, 1998.

[52] W. Kühnel, *Differentialgeometrie*, 4th ed. Wiesbaden: Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH, 2008.

[53] "Programming languages — C++," International Organization for Standardization, Geneva, CH, Standard, 12 2017.

[54] "Information technology — Programming languages — C++," International Organization for Standardization, Geneva, CH, Standard, 09 2011.

[55] B. Meyer, "Applying 'design by contract'," *Computer*, vol. 25, no. 10, pp. 40–51, 1992.

[56] B. Stroustrup, *The C++ Programming Language*. Addison-Wesley, May 2013.

[57] K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, "AddressSanitizer: A Fast Address Sanity Checker," *USENIX ATC 2012*, 2012.

[58] D. Hutchins, A. Ballman, and D. Sutherland, "C/C++ Thread Safety Analysis," *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*, pp. 41–46, 2014.

[59] T. Kremenek, "Finding software bugs with the clang static analyzer," *Apple Inc*, 2008.

[60] B. Babati, G. Horváth, V. Májer, and N. Pataki, "Static analysis toolset with clang," *Proceedings of the 10th International Conference on Applied Informatics (ICAI 2017)*, pp. 23–29, 2017.

[61] M. Fowler and M. Foemmel, "Continuous integration," https://martinfowler.com/articles/originalContinuousIntegration.html, 09 2000.

[62] "Programming Languages – C++ Extensions for Concepts," International Organization for Standardization, Geneva, CH, Technical Specification, 08 2015.

[63] H. Schreiner, P. Top, M. Brinkmann, and C. Bachhuber, "CLIUtils/CLI11," 01 2020. [Online]. Available: https://github.com/CLIUtils/CLI11

[64] A. Gauniyal, "Rang - A Minimal, Header only Modern C++ library for terminal goodies," 2018. [Online]. Available: https://github.com/agauniyal/rang

[65] T.-W. Huang, C.-X. Lin, G. Guo, and M. S. Wong, "Cpp-Taskflow: Fast Task-Based Parallel Programming Using Modern C++," *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 974–983, 2019.

[66] V. Zverovich and J. Müller, "fmt - A modern formatting library," 2020. [Online]. Available: https://github.com/fmtlib/fmt

[67] Microsoft, "GSL - Guidelines Support Library," 2019. [Online]. Available: https://github.com/microsoft/GSL

[68] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[69] B. Organization, "The Boost Libraries," 2020. [Online]. Available: https://www.boost.org/

[70] V. Kirilov, "doctest - The fastest feature-rich C++11/14/17/20 single-header testing framework for unit tests and TDD," 2020. [Online]. Available: https://github.com/onqtam/doctest

[71] R. M. Fernandes, "nonius - A C++ micro-benchmarking framework," 2019. [Online]. Available: https://github.com/libnonius/nonius

[72] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[73] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, pp. 77–116, 09 1998.

[74] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2911–2918, 2012.

[75] E. Hellinger, "Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen." *Journal für die reine und angewandte Mathematik*, vol. 1909, no. 136, pp. 210 – 271, 1909.

[76] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

[77] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," *Mathematische Annalen*, vol. 71, pp. 38–53, 1911.

[78] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision – ECCV 2006*, pp. 430–443, 2006.

[79] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," *Computer Vision – ECCV 2010*, pp. 778–792, 2010.

[80] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291 – 307, 1999.

[81] A. Ziegler, E. Christiansen, D. Kriegman, and S. J. Belongie, "Locally Uniform Comparison Image Descriptor," *Advances in Neural Information Processing Systems 25*, pp. 1–9, 2012.

[82] M. G. Kendall, "A NEW MEASURE OF RANK CORRELATION," *Biometrika*, vol. 30, no. 1-2, pp. 81–93, 06 1938.

[83] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," *Computer Vision – ECCV 2012*, pp. 214–227, 2012.

[84] Xin Yang and Kwang-Ting Cheng, "LDB: An ultra-fast feature for scalable Augmented Reality on mobile devices," *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 49–57, 2012.

[85] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pp. 127–136, 10 2011.

[86] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2020.

[87] Zichao Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 801–808, 2016.

[88] A. Singh, "Monocular visual odometry," IIT Kanpur, 2015. [Online]. Available: https://avisingh599.github.io/assets/ugp2-report.pdf