Robust and Accurate Detection of Mid-level Primitives for 3D Reconstruction in Man-Made Environments

Dominik Wolters

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel
eingereicht im Jahr 2018

Kiel Computer Science Series (KCSS) 2019/1 dated 2019-01-02

URN:NBN urn:nbn:de:gbv:8:1-zs-00000348-a8
ISSN 2193-6781 (print version)
ISSN 2194-6639 (electronic version)

Electronic version, updates, errata available via https://www.informatik.uni-kiel.de/kcss

The author can be contacted via https://www.mip.informatik.uni-kiel.de

Published by the Department of Computer Science, Kiel University

Multimedia Information Processing Group

Please cite as:

▻ Dominik Wolters. *Robust and Accurate Detection of Mid-level Primitives for 3D Reconstruction in Man-Made Environments.* Number 2019/1 in Kiel Computer Science Series. Department of Computer Science, 2019. Dissertation, Faculty of Engineering, Kiel University.

```
@book{wolters_robust_2019,
  author    = {Dominik Wolters},
  title     = {Robust and Accurate Detection of Mid-level Primitives
               for 3D Reconstruction in Man-Made Environments},
  publisher = {Department of Computer Science, Kiel University},
  year      = {2019},
  number    = {2019/1},
  doi       = {10.21941/kcss/2019/1},
  series    = {Kiel Computer Science Series},
  note      = {Dissertation, Faculty of Engineering, Kiel University.}
}
```

About this Series

The Kiel Computer Science Series (KCSS) covers dissertations, habilitation theses, lecture notes, textbooks, surveys, collections, handbooks, etc. written at the Department of Computer Science at Kiel University. It was initiated in 2011 to support authors in the dissemination of their work in electronic and printed form, without restricting their rights to their work. The series provides a unified appearance and aims at high-quality typography. The KCSS is an open access series; all series titles are electronically available free of charge at the department's website. In addition, authors are encouraged to make printed copies available at a reasonable price, typically with a print-on-demand service.

Please visit `http://www.informatik.uni-kiel.de/kcss` for more information, for instructions how to publish in the KCSS, and for access to all existing publications.

1. Gutachter:	Prof. Dr.-Ing. Reinhard Koch Christian-Albrechts-Universität Kiel
2. Gutachter:	Prof. Dr.-Ing. Helmut Mayer Universität der Bundeswehr München Neubiberg

Datum der mündlichen Prüfung: 28. November 2018

Zusammenfassung

Die Detektion von geometrischen Primitiven wie Punkten, Linien und Bögen ist ein elementarer Verarbeitungsschritt für viele Techniken des maschinellen Sehens wie Bildanalyse, Mustererkennung und 3D-Szenenrekonstruktion.

In dieser Arbeit wird eine Methode vorgestellt, die eine zuverlässige Detektion von geometrischen Primitiven in Bildern ermöglicht. Der Fokus liegt auf der Anwendung in urbanen Umgebungen, wobei der Prozess nicht darauf beschränkt ist.

Die Methode ermöglicht eine robuste und subpixelgenaue Detektion von Punkten, Linien und Bögen und erstellt einen Graphen, der die topologischen Beziehungen zwischen den detektierten Merkmalen beschreibt. Die Detektionsmethode kann direkt auf verzeichnete perspektivische Bilder und Fischaugenbilder angewendet werden. Die zusätzliche Erkennung sich wiederholender Strukturen in Bildern gewährleistet die Eindeutigkeit der Merkmale in ihrer lokalen Umgebung.

Das neu entwickelte Verfahren erreicht eine hohe Lokalisierungsgenauigkeit, die dem Stand der Technik entspricht und gleichzeitig robuster gegenüber Störungen durch Rauschen ist. Darüber hinaus ermöglicht das Verfahren, mehr Details in den Bildern zu extrahieren. Die Detektionsrate ist bei dem neuen Verfahren auf den realen Datensätzen stets höher als bei dem aktuellen Stand der Technik. Darüber hinaus kann das neue Verfahren zuverlässig zwischen Linien- und Bogensegmenten unterscheiden.

Die durch das neue Verfahren gewonnenen zusätzlichen topologischen Informationen sind weitgehend konsistent über mehrere Bilder einer Szene und können somit eine Unterstützung für nachfolgende Verarbeitungsschritte wie Matching und Korrespondenzsuche sein.

Die Detektionsmethode wird in eine vollständige merkmalsbasierte 3D-Rekonstruktionspipeline integriert und es wird eine neuartige Rekonstruktionsmethode vorgestellt, die die topologischen Beziehungen der Merkmale nutzt, um ein abstraktes, aber zugleich semantisch reichhaltiges

3D-Modell der rekonstruierten Szenen zu erstellen, in dem komplexere geometrische Strukturen leicht detektiert werden können.

Abstract

The detection of geometric primitives such as points, lines and arcs is a fundamental step in computer vision techniques like image analysis, pattern recognition and 3D scene reconstruction.

In this thesis, we present a framework that enables a reliable detection of geometric primitives in images. The focus is on application in man-made environments, although the process is not limited to this.

The method provides robust and subpixel accurate detection of points, lines and arcs, and builds up a graph describing the topological relationships between the detected features. The detection method works directly on distorted perspective and fisheye images. The additional recognition of repetitive structures in images ensures the unambiguity of the features in their local environment.

We can show that our approach achieves a high localization accuracy comparable to the state-of-the-art methods and at the same time is more robust against disturbances caused by noise. In addition, our approach allows extracting more fine details in the images.

The detection accuracy achieved on the real-world scenes is constantly above that achieved by the other methods. Furthermore, our process can reliably distinguish between line and arc segments.

The additional topological information extracted by our method is largely consistent over several images of a scene and can therefore be a support for subsequent processing steps, such as matching and correspondence search.

We show how the detection method can be integrated into a complete feature-based 3D reconstruction pipeline and present a novel reconstruction method that uses the topological relationships of the features to create a highly abstract but semantically rich 3D model of the reconstructed scenes, in which certain geometric structures can easily be detected.

Acknowledgements

First of all I would like to thank Prof. Reinhard Koch for the opportunity to work on a PhD under his supervision. Without his constant and encouraging guidance, the completion of this work would not have been possible.

Furthermore I would like to thank all the members of the Multimedia Information Processing Group, Johannes Brünger, Sascha Clausen, Sandro Esquivel, Oliver Fleischmann, Yuan Gao, Andreas Jordt, Anne Jordt, Daniel Jung, Falko Kellner, Tim Michels, Luca Palmieri, Arne Petersen, Stefan Reinhold, Simon-Martin Schröder, Tobias Schwede, Renate Staecker, Torge Storm, Yu Tang, Claudius Zelenka, for their great social interaction and academic cooperation.

Special thanks go to Jan-Friso Evers-Senne and the whole TVS team, with whom I worked for four exciting and challenging years on the development of innovative measuring instruments. Especially my colleagues from the image processing and measurement group, Hellen Altendorf, Felix Born, Pablo Bueno Plaza, Jens Einighammer, Jelena Ivanova, Andreas Jordt, Jan Tenfelde, Robert Wulff, have contributed to the success of this work through several discussions, telephone conversations and of course all the great social events.

I thank Prof. Helmut Mayer for reviewing my thesis as well as Prof. Wilhelm Hasselbring and Prof. Hauke Schramm for their participation in the assessment commission.

Finally, I would like to thank my family, especially my parents and sister, for their outstanding support over so many years.

Contents

Chapter 1

Introduction

1.1 Motivation

Image features bridge the gap between low-level pixel intensities and semantic meanings. They are a fundamental step in computer vision techniques such as image processing, image analysis and pattern recognition.

Feature detection denotes the identification of geometric primitives, i. e., points, lines and arcs, in digital images. It is a low-level processing step, which has the pixel intensities as input and information about the image structure as output.

The aim of feature detection is to identify important events and changes in the properties of the captured scene by examining changes in the intensity values of the image. This can be discontinuity in depth, changes in material properties or different scene illumination. A well-designed feature detector must be able to handle different imaging conditions and extract visual clues that meet human interests.

Ideally, the image features allow an abstract description of the objects of the scene and their boundaries. The less relevant information is filtered out, but the important information about the structure of the scene is retained. This leads to a significant reduction of the data to be processed. If the relevant structures of the scene are successfully extracted, the subsequent task of interpreting the scene is significantly simplified.

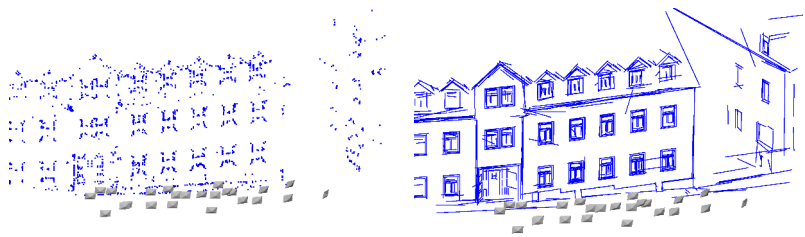
One of the major challenges in image feature detection is to explore the relationships between different feature types. The definitions of the image features are topologically connected, but in the current procedures these topological relationships are hardly exploited. There is already a variety of algorithms to extract image features. However, most approaches are specialized in one feature type. Although there are some approaches that

allow simultaneous detection of multiple primitives [För94], a framework that provides accurate and robust detection of various feature types for comprehensive image analysis is lacking.

The types of primitives extracted in the images affect the complete subsequent processing and its performance. The geometric primitives commonly used are points, lines and arcs. Many works are concerned with point features and their applications in computer vision. In contrast, line segments and arc segments are less researched.

The different geometric primitives provide complementary information about the image. In many scenes where point features are missing, reliable and sufficient lines are available. A typical example for this are low-textured man-made environments. In addition, lines offer a higher level of structural information about the scene. The contours of objects can be well described by line and arc segments, especially for artificial objects.

Figure 1.1 shows a comparison between a point-based and a line-based reconstruction of the same scene. The line-based reconstruction helps to understand the geometry of the scene much easier. Furthermore, additional geometric conditions can often be applied more easily to line features. A condition that is often used in man-made environments is the Manhattan world assumption. It states that the scene has three mutually orthogonal dominant directions. For lines, unlike points, it is easy to check whether they are either parallel or orthogonal to each other.



(a) Point-based reconstruction **(b)** Line-based reconstruction

Figure 1.1. Comparison between a point-based and a line-based reconstruction of the same scene.

A disadvantage of line segments is that the matching of lines, as required for feature-based 3D reconstruction, is much more complicated than for points. The reasons for this are that the positions of the end points are often inaccurately localized, that the lines are fragmented into several subsegments and that there are no strong geometric conditions, such as epipolar conditions.

A combined detection of lines, arcs and points from intersections can be used to solve these issues. The points obtained as intersections of line and arc segments have a higher localization accuracy and greater relevance. At the same time, the intersection points represent robust and reliable end points for the line segments, which can significantly simplify further processing. Since the intersections of the line and arc segments correspond to the corners of the real objects, they are very useful for further processing steps and applications such as 3D scene reconstruction and image analysis.

1.2 Contribution and Overview

We have developed a framework for an accurate and robust detection of the most important image features. We propose a method that enables a combined detection of points, lines and arcs. The focus is on application in man-made environments, although the process is not limited to this.

The intersections of the extracted line and arc segments are used as point features. This results in higher localization accuracy and greater relevance of the point features compared to conventional corner detection methods.

Current problems that occur in the detection process of geometric primitives, such as the fragmentation of lines into several segments and misdetection, which do not belong to interesting structures of the scene, are addressed by the proposed method.

The method we propose enables the detection of geometric primitives independent of the underlying mapping function of the camera. Thus, the method can also be used for the detection of geometric primitives in distorted perspective or fisheye images without the need to generate software-corrected images with an inverse distortion. This can improve both the localization accuracy and the detection speed.

The reliability of the feature detection can be significantly increased by considering the sensor noise of the camera in the detection process. This enables the detection of faint features in low-contrast regions without increasing the number of misdetection.

The additional recognition of repetitive structures in images ensures the unambiguity of the features in their local environment. This prevents errors in subsequent image processing steps, e. g., due to incorrect image feature correspondence.

A novelty of the method is that the topological relationships of the features to each other are extracted from the images and stored in a graph structure. This topological information can make meaningful contributions to support subsequent image processing steps, as we demonstrate with concrete application examples.

By successfully detecting and extracting the relevant structures of the scene and their topological relationship, the following task of interpreting the scene is significantly simplified

Many aspects of this research have already been published in the following articles:

- D. Wolters. "Automatic 3D Reconstruction of Indoor Manhattan World Scenes Using Kinect Depth Data". In: *Pattern Recognition*. Ed. by X. Jiang, J. Hornegger, and R. Koch. Springer International Publishing, 2014, pp. 715–721

- D. Wolters and R. Koch. "Precise and Robust Line Detection for Highly Distorted and Noisy Images". In: *Pattern Recognition*. Ed. by B. Rosenhahn and B. Andres. Springer International Publishing, 2016, pp. 3–13

- D. Wolters and R. Koch. "Combined Precise Extraction and Topology of Points, Lines and Curves in Man-Made Environments". In: *Pattern Recognition*. Ed. by V. Roth and T. Vetter. Springer International Publishing, 2017, pp. 115–125

- D. Wolters and R. Koch. "Topology-Based 3D Reconstruction of Mid-level Primitives in Man-Made Environments". In: *Pattern Recognition*. Ed. by T. Brox, A. Bruhn, and M. Fritz. Springer International Publishing, 2018

1.3 Outline

This thesis is structured as follows. We start in Chapter 2 with a detailed overview of the different types of image features and the current state-of-the-art methods for detecting them. In addition, we give an overview of related methods for feature-based 3D reconstruction and facade interpretation as the primary field of application for the detected image features.

In Chapter 3 we introduce our method for the detection of image features. We explain the process flow for the combined detection of points, lines and arcs and describe how the topological relationships between the detected features can be extracted.

In Chapter 4 we perform an extensive evaluation of our proposed method based on numerous experiments with synthetic and real-world datasets. Furthermore, we compare our approach with the current state-of-the-art methods.

In Chapter 5 we demonstrate the integration of our detection method into a complete feature-based reconstruction pipeline and present a novel method that uses the extracted topological relationships to create a semantically rich and at the same time abstract model of the scene.

Finally, in Chapter 6 we draw a conclusion and present ideas for future work.

Chapter 2

Related Work

2.1 Feature Detection

Feature detection refers to extracting interesting image structures that provide salient visual cues in images. These structures are primitives such as points, lines and curves but also regions. Many researches are concerned with the detection of a large number of different features as they are widely used in computer vision techniques. Typical applications are object recognition, tracking or structure from motion. Figure 2.1 illustrates the classification of feature detection methods.

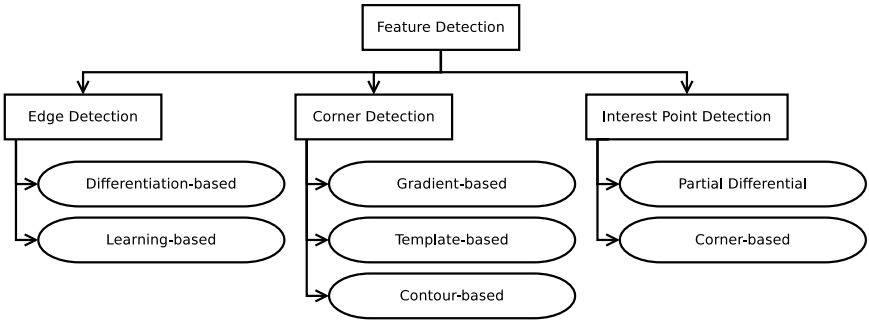


Figure 2.1. Classification of feature detection methods (adapted from [LWT+15]).

Despite the many different applications, the underlying objectives are always the robust and efficient detection of features that are highly stable and unambiguous. An extensive survey of recent developments of visual feature detection can be found in [LWT+15].

Closing the gap between low-level visual cues and high-level concepts, e. g., in the reconstruction, is an important research topic. Accurate and meaningful image features can significantly affect the performance of computer vision based applications. Although progress has been made in this area in recent years, there are still many challenges. The main challenge in detecting image features is a robust handling of different imaging conditions caused by changes in illumination, scaling, viewpoint, noise, etc.

The visual features are strongly related with the nature of human perception. The Gestalt law describes in psychology human perception as the ability to identify structures and principles of order in sensory impressions. The aims of computer vision are generally the computer-aided solution of tasks that are based on the capabilities of the human visual system. Therefore, feature detection employs many concepts that are inspired by human perception. [LWT+15]

2.1.1 Edge Detection

Edge detection attempts to identify the boundaries between different regions in a digital image if they differ sufficiently in brightness. For this purpose, mathematical methods are used which detect discontinuities in the image. The boundaries at which the image brightness changes sharply usually consist of a set of line and curve segments called edges.

Sharp changes in image brightness are usually associated with important events in the scene. These may be, for example, discontinuities in depth, changes in the material properties, discontinuities in surface orientation or changes in the scene illumination.

Differentiation-based Edge Detection

Classical methods for edge detection use differentiation-based filters to identify discontinuities in brightness.

Gradient operators based on first-order differentiation are used in pairs with different orientations. Examples of such filters are Sobel and Prewitt [ZT98]. The local maxima of the gradient magnitude correspond to the edges.

Zero-crossing based methods use second-order differentiation filters such as Laplacian to find edges [ZT98].

The differentiation-based edge detection is very simple but sensitive to noise [LWT+15]. Therefore, a smoothing filter, typically Gaussian smoothing, is normally used as the preprocessing step.

The Canny edge detector [Can86] models edge detection as a three-criteria optimization problem. These are detection of edges with low error rate, accurate localization on the center of the edge and minimizing multiple responses to a single edge. The Canny edge detection process consists of several steps. First, the image is smoothed to reduce the noise. For this purpose, a Gaussian filter is used. Subsequently, the partial derivatives are determined in the horizontal direction and in the vertical direction. From this, the gradient magnitude and the gradient direction can be calculated. Non-maximal suppression is applied to suppress all the gradient values except the local maxima, that correspond to the locations with the sharpest change of the intensity value. The edges are then traced and determined by hysteresis thresholds. The Canny edge detector is still widely used and outperforms several new detectors [LWT+15].

In recent years, the research topics in the field of differentiation-based edge detection have been the use of multiple resolution levels, selection of thresholds, and subpixel accurate detection [LWT+15].

The motivation for multi-resolution edge detection is that the detection results are scale-dependent and that human perception is multi-resolutional. Gaussian smoothing, Sobel operators and coarse-to-fine edge tracking are used to detect multi-scale edges [LDB+13].

The selection of the threshold values is decisive for the results of the edge detection. Hysteresis thresholding helps to generate connected edges. Various publications deal with the selection of upper and lower thresholds. Some methods analyze the histogram of the gradient magnitudes to determine the thresholds [SP10].

In order to increase the detection accuracy, subpixel accurate edge detection is used. There are several approaches to achieve subpixel accurate detection. Fitting-based approaches attempt to determine the subpixel position by fitting the gray values of the image to an assumed edge model [YFP05]. Interpolation-based methods determine the subpixel position by interpolating the image data or their derivatives [HBB+08]. The

interpolation-based methods are very efficient but sensitive to noise. Therefore, moment-based methods, which are not sensitive to noise, are often used today [DZ10].

Learning-based Edge Detection

One of the main problems of differentiation-based edge detection is that in highly textured areas, many edges are detected within the textured image regions. This makes it difficult to detect the boundaries between the different image regions. Learning-based edge detection methods try to solve this problem.

These edge detectors use a machine learning based framework to extract the edges between the different regions while suppressing edges within the regions. For this, natural images and corresponding boundaries marked by humans are used in the training and validation set. These approaches look at image patches and determine the likelihood that the center pixel is part of an edge. Normally, statistical learning algorithms and multiple low-level cues are used and combined into a model for edge response prediction [LWT+15].

The methods differ in the cues extracted from the images and the statistical learning algorithms used. First approaches such as the Pb edge detector [MFM04] consider multiple local cues such as the brightness gradient, the texture gradient, and the color gradient. Logistic regression is used to combine the cues and train the model for the edge response prediction. Based on this approach, multi-scale edge detectors [Ren08] have been developed because edge detection is scaling dependent.

Other promising approaches, such as the structured forests edge detector [DZ15], are based on learning of the randomized decision trees. In the method, color and gradient channels are used as input features. In addition, pairwise difference features are used. Structured labels are used to determine the splitting function at each branch in the tree. Each forest predicts a patch of edge pixel labels that are aggregated across the image to calculate the final edges.

The edges detected by the learning-based approaches are closer to human perception. Edges within textures can be effectively suppressed. The methods are, however, usually more computationally expensive. Because

the methods analyze subregions instead of pixels, the localization accuracy is lower. Subpixel accurate localizations are so far only achieved in the differentiation-based methods. Therefore, in order to extract mid-level features, differentiation-based methods are still generally used [LWT+15]. The learning-based approaches are used primarily for image segmentation tasks.

Subsequent steps are the building of connected pixel chains from the detected edge pixels as well as the grouping of the contours to determine the semantic object boundaries.

2.1.2 Corner Detection

Corners are defined as the intersection of two connected edges. This means that a corner is a point in the image where two dominant and different gradient directions occur [LWT+15].

The methods to detect corners in images can be divided into three classes [LWT+15]: The gradient-based methods use local gradient information to determine the corners. The template-based methods perform pixel comparisons, and the contour-based methods use the results of edge detection techniques to determine intersections of the edges.

Gradient-based Corner Detection

The first approaches to detect corners are based on the detection of gradients in the images. A well-known example is the Harris corner detector [HS88]. The assumption is that a corner is a point that has a low self-similarity. For each pixel, it is determined how similar a patch centered on the pixel is to overlapping patches in the local environment. The similarity is measured by taking the sum of squared differences (SSD). The Harris corner detection is based on the autocorrelation of gradients on shifting windows.

The Förstner corner detector [FG87] uses a similar approach. Instead of the autocorrelation matrix, the covariance matrix is considered. To determine the corners, the size and roundness of the error ellipse are used. Corners have a small and round error ellipse. A comprehensive framework

for the consistent low-level feature extraction of points, edges and segment regions is described by Förstner in the subsequent work [För94].

Other gradient-based corner detectors are Kanade-Lucas-Tomasi (KLT) operator [TK91] and Shi-Tomasi corner detector [ST94]. The main difference between the methods is the function used to determine the corner strength of a pixel.

Typical problems of the gradient-based approaches are that the detection is sensitive to noise and the algorithms are computationally expensive.

More recent publications, such as S-LOCOCO [MYL+11], deal with the possibility of reducing the computational costs and at the same time achieving a high accuracy in the detection. The approach uses a box kernel to approximate the Gaussian derivative kernel and an integral image to speed up the computation of the cornerness response. To achieve a subpixel accurate localization, the position of the corners are interpolated.

Template-based Corner Detection

Template-based edge detectors compare the intensity values within a mask around the pixel with the center pixel to determine the positions of corners.

A classic detector of this category is SUSAN (smallest univalue segment assimilating nucleus) [SB97]. The intensity of each pixel within a circular mask is compared to the center pixel. Subsequently, the area of the mask is determined which has the same or a similar intensity as the center pixel. The points where this area is smallest are selected as corners.

Unlike the gradient-based methods, no derivatives have to be calculated and no noise reduction has to be performed. In recent years, approaches have been increasingly integrated machine learning algorithms into the template-based corner detection to accelerate the detection.

The FAST (features from accelerated segment test) detector [RD06] uses a circle of 16 pixels. If more than n contiguous pixels of the circle are lighter or darker than the center pixel intensity and a threshold, that point is considered as a corner. To speed up the detection process, a learned decision tree is used to determine the order of pixels for comparison.

The AGAST (adaptive and generic accelerated segment test) [MHB+10] detector is a variant of the FAST detector. By combining specialized de-

cision trees, the corner detector is more generic and does not have to be adapted to new environments.

The template-based corner detectors are usually much faster than the gradient-based methods. A disadvantage of template-based corner detectors is that effective and precise methods are lacking to determine the corner strength [LWT+15].

Contour-based Corner Detection

Corners are defined as the intersection of two edges. A number of methods use this by finding the corners based on contour detection. These methods usually search for the points with the maximum curvature along the contours. An overview and a comparison of different contour-based corner detectors can be found in [ALF12].

The contour-based approaches first perform edge detection (see Section 2.1.1) and build planar curve, i. e., open or closed contours. Then they analyze the planar curves to find the locations of rapid changes in direction. The contour-based detection methods can be divided into three different classes based on the strategy for selecting of the final corners [ALF12]. A number of methods use fixed thresholds to select strong corners and discard weak ones [MS98]. A second group directly determines the curvature extrema as corners, without the additional curvature thresholds [PB10]. The third group consists of algorithms that approximate the contour by a polygon and use the intersections of the adjacent line segments as corners [PG10].

The terms corner and junction are not used consistently in the literature. The term junction is typically used when the point is assigned with information about the intersecting edges. The intersecting edges may be connected via T-, L- or X-junctions. In [MAF+08] an approach for a combined contour and junction detection based on combination of local and global cues and statistical learning is presented.

The contour-based corner detectors differ significantly in the detection procedure and the application areas from the gradient-based and template-based methods. The contour-based corner detectors are highly dependent on the contours generated by edge detection and linking. Traditionally, the contour-based corner detectors are more commonly used in shape

analysis and shape-based image compression and not for wide baseline matching [LWT+15].

2.1.3 Interest Point Detection

An interest point is a point in the image that has a well-defined position and can be robustly detected. The point is unique in its local environment. An interest point can be defined as a local extremum in three-dimensional scale space with the locations and the scale as axes [LWT+15].

Therefore, an interest point can also be a corner with a fixed scale. However, it can also be a local intensity maximum or minimum, the end point of a line or a point of maximum curvature on a contour.

The classical interest point detectors use a Gaussian pyramid together with Laplacian of Gaussian (LoG), difference of Gaussian (DoG) or Hessian-Laplacian [HAA16]. These approaches are still widely used in image processing algorithms.

A common interest point detector is SIFT (scale-invariant feature transform) [Low99]. The candidates for interest points are the maxima and minima of the difference of Gaussian function applied in the scale space. In further developments of the SIFT detector, such as SURF (speeded-up robust features) [BET+08], the computational effort has been reduced and the robustness of the detector has been improved.

In recent years, new interest point detectors have been developed which, unlike the classical methods, are not based on partial differentiation on Gaussian scale space. Based on the assumption that Gaussian filtering results in blurred images, KAZE [ABD12] uses nonlinear diffusion filtering to generate the scale space of an image. This preserves the natural image boundaries.

Another group of methods combines corner detection, which allows fast computation, with image pyramids to extract interest points. ORB (oriented FAST and rotated BRIEF) [RRK+11] uses the FAST corner detector at each level of an image pyramid.

2.1.4 Geometric Primitives: Lines, Arcs and Ellipses

The detection of geometric primitives is one of the basic tasks of image processing, as they are often the prerequisite for higher-level procedures. Most of the existing methods to detect geometric primitives are limited to a single primitive type, i. e., line segments or ellipses. The detection methods can roughly be divided into two classes: Hough-based methods or edge chaining methods. Most methods require as input an edge map generated by edge detection (see Section 2.1.1).

Hough-based Methods

The first methods to detect geometric primitives use the Hough transform [Bal81] to determine a set of primitives from an edge map. Typically, the Canny detector [Can86] is used to detect the edges.

The Hough transform is a robust global method to detect objects that can be represented in closed form, e. g., straight lines, circles and ellipses. First, primitive candidates are sampled from the primitive parameter space in an exhaustive manner. Subsequently, the edge points vote for the candidates according to a defined criterion. Candidates with a sufficient number of edge pixels lying on them are selected as detections.

These methods usually require an accurate adjustment of the parameters depending on the image data and suffer from a high memory consumption and a long execution time.

To improve the execution time, randomized methods [XO93] are used that take into account only a fraction of the edge pixels in the voting process. In addition, probability methods are used to automatically select meaningful thresholds [MGK00].

A general problem of Hough-based line detection methods is that usually infinitely long straight lines are detected. The straight lines must be split into individual line segments in a post-processing step.

Edge Chaining Methods

The edge chain methods exploit the geometric properties of the searched primitives, e. g., straightness for lines or the curvature of ellipses. The first step is usually to build chains of connected edge pixels starting from one

or a group of seed pixels. Subsequently, the geometric primitives are fitted on the edge chains. For this purpose, different methods are used, e. g., least squares fitting techniques or RANSAC-like approaches.

LSD (line segment detector) [vGJM+10] works directly on the intensity values of the input image. The first step is to group connected pixels that have a similar gradient direction. Then rectangles are fitted into the grouped pixels. To control the false detections, a validation step based on an a-contrario approach and the Helmholtz principle [DMM08] is used.

EDLines [AT11] also uses the a-contrario theory and performs similarly to LSD. By using a very fast edge detection algorithm [TA12] that simultaneously detects edges and builds edge chains, the EDLines detector is up to ten times faster. This makes the detector suitable for real-time applications.

A typical problem of edge chaining methods is that noise or lack of contrast causes line segments, which are actually connected, to fragment into several short sub-segments. Multi-scale approaches [SMM16] are used to avoid this over-segmentation.

An algorithm that detected not only lines but also circular arcs was already presented in [Ete92]. The algorithm is parameterless and uses an edge map as input. The algorithm usually produces accurate results but has no ability to remove false positives.

In [WF13] an adaption to the Douglas-Peucker algorithm is used to approximate given pixel chains by a sequence of lines and elliptical arcs.

The recently proposed ELSDc [PGG17] allows a parameterless detection of line segments and ellipses. They use a fitting algorithm for the ellipses that uses both the algebraic distance from the conic equation and the deviation from the gradient direction. For model validation and model selection they use statistical criteria based on the a-contrario theory.

2.1.5 Line Detection in Distorted Images

Most image processing algorithms are based on the pinhole camera model. Here, it is assumed that a 3D line in space is projected on a 2D line in the image plane. Especially wide-angle cameras introduce a variety of optical distortions that need to be corrected to meet this assumption. Causes

of such distortions may be, for example, inaccuracies of the lenses or misalignment of the optical system.

If the calibration is known, a distortion correction is usually applied as a preprocessing step before the image processing algorithms. In the case of highly distorted images, e. g., fisheye cameras, this is not always possible.

Normal line detection algorithms cannot be applied to distorted images because the lines are not straight. In [AAG+14], a detection method is proposed in which the Hough transform is extended by a parameter describing the lens distortion. The search space is a three-dimensional space consisting of the orientation, the distance from the origin and the distortion. Based on the detected lines, the lens distortion can be estimated and corrected.

In the case of a catadioptric camera, special detection methods exist [VM04]. The detection takes place in the space of the equivalence sphere, since this is the uniform domain of central catadioptric sensors. Real lines are projected on great circles on this sphere, which are detected via a Hough transformation.

A similar approach is used by [BSB+13]. Instead of the Hough transform, they use a RANSAC-based method to fit lines into edge chains projected onto the equivalence sphere.

2.1.6 Repetitive Structures

In architecture and man-made objects often repetitive structures occur. In image processing algorithms, such as matching, repetitive structures are a challenge because they violate the assumption that features are unique in their local environment. For this reason, repetitive structures should be detected in images, so that can be treated separately in the subsequent processing steps.

The existing methods focus mainly on detecting planar patterns in single images. The approaches usually extract a sparse set of corresponding local image features and identify symmetries by growing or tracking from the initial feature set to their spatial neighbors [LM96; WDF08].

An entirely different approach is used in a recently introduced salience-based line detector [BWG15]. The approach takes into account the surroundings of the lines and thus detects the regions of significant divergence

between pixel intensity or color statistics. The detector thereby avoids the repetitive parts of a scene and recognizes the strong, discriminating lines.

2.2 Point-based 3D Reconstruction

2.2.1 SfM Approaches

An important field of application for image features is image-based 3D reconstruction. The processing of real and large datasets to realistic 3D models has only become possible with the introduction of efficient and robust methods for the detection and description of image features.

The reconstruction of a three-dimensional structure from an image sequence with different viewpoints is called structure from motion (SfM). For the three-dimensional reconstruction of objects, correspondences between images must be found. For this purpose, image features, such as corners and lines, are tracked from one image to the next. The result of a SfM procedure usually consists of the camera poses at which the images were taken and a sparse three-dimensional point cloud of the scene.

First methods that enabled the reconstruction of sights from large unsorted Internet photo collections [SSS06] have contributed to the popularity of SfM applications. In the following years, the efficiency of the processes was further increased so that complete cities could be reconstructed from huge data sets [ASS+09; FFG+10].

There are numerous different approaches for SfM [ÖVB+17]. The most common groups include global approaches [COS+11; SSH+15], hierarchical approaches [GFF10] and incremental approaches [SSS06; ASS+09; FFG+10; SF16]. The incremental SfM approaches are the most popular methods for unsorted image datasets [SF16]. In the following section, we explain the basic structure of an incremental SfM pipeline.

2.2.2 Basic Incremental SfM Pipeline

The most common SfM pipeline design consists of five steps [Hof16]: feature detection, matching, geometric verification, reconstruction and bundle adjustment. The different steps are described in the following sections.

Feature Extraction

The first step is the detection of image features. In addition, descriptors are extracted that describe the feature based on certain properties of the local environment. These can be, e. g., image intensities or the orientation of the gradients. Common point descriptors are SIFT [Low04], SURF [BET+08], BRISK [LCS11] and FREAK [AOV12]. SIFT achieves the highest matching rates in most evaluations. In contrast, binary descriptors such as BRISK and FREAK generally offer higher efficiency at the cost of reduced robustness [HDF12].

Matching

The next step is matching. In this step, pairs of visually similar features are determined from two different images. The extracted descriptors are used to measure the similarity of two features. In the matching process, in general, for each feature from the first image, a maximum of one corresponding feature is determined from the second image.

A comparison of every image pair of the image sequence to test the images for overlap is generally too time-consuming for large image datasets. Therefore, many approaches are concerned with methods that enable efficient and scalable matching [ASS+09; FFG+10; HSD+15].

Geometric Verification

In the third step, the matched image pairs are verified. Since matching is based only on appearance, a transformation between the images is estimated using projective geometry for verification.

Usually the relative pose between the images is determined based on the matched image features. Since the results of the matching often contain outliers, robust methods such as RANSAC [FB81] are used.

The result of this step is a so-called scene graph. It contains the images as nodes and the verified image pairs as edges [HSD+15; SF16].

Reconstruction

In the incremental reconstruction step, the camera poses are estimated and the scene is reconstructed as a sparse point cloud.

The process begins with the initialization of the model from the first camera pair. The robustness and accuracy of the reconstruction is affected by the choice of the pair, so different strategies have been developed for the choice of a suitable pair [SF16].

Images from further camera positions are added incrementally to a consistent 3D model. To register new images, a Perspective-n-Point (PnP) problem [FB81] is solved using the feature correspondences to already triangulated points. This estimates the camera position for the new image. RANSAC and minimal pose solvers [LMF09; SF16] are used to handle outliers.

Since the new camera must have observed existing 3D points, a re-triangluation is then performed. In addition, new 3D points can be triangulated if the new part of the scene has been covered by at least one more image with a different pose. There are numerous methods for robust and efficient multi-view triangulation [HS97; KWY14; SF16].

Bundle Adjustment

The last step consists of a global bundle adjustment. In a combined process, the camera parameters and the 3D points are refined by minimizing the reprojection error of the entire system. The bundle adjustment is often integrated in the incremental reconstruction step.

Nonlinear least squares algorithms are used for minimization. Levenberg-Marquardt is often used because it is easy to implement and enables fast convergence [TMH+00].

2.3 Line-based 3D Reconstruction

The state-of-the-art SfM methods generally use point features. There are only a small number of line-based reconstruction methods that can handle realistic datasets.

One of the reasons for this is that pose estimation based on line segments is much more complex [Hof16].

The line-based reconstruction is nevertheless useful, because especially in man-made environments there are only few textured surfaces, which are necessary for a point-based reconstruction, and at the same time the structure of the scenes can be much better represented by line segments. A comprehensive overview of the development of line-based reconstruction methods and current approaches can be found in [Hof16].

The line based reconstruction methods can be divided into two groups: On the one hand, methods that perform a complete SfM pipeline based on lines and on the other hand, methods that assume the camera poses are given.

Line based reconstruction has been researched for many years. One of the first methods was proposed by [YH83] and allows the estimation of the 3D motion parameters of a rigid body using three images from a static camera. However, the corresponding line features between the images are assumed to be known.

The procedures were further improved in the following years. Promising approaches have been presented, especially in recent years.

In [EE11], a method is proposed for calculating the relative pose between two images using triplets of 2D lines, where two lines must be parallel and one orthogonal to them. Using synthetic and real data, the authors show that the approach works robustly and has advantages over point-based methods for indoor scenes and can be integrated into any SfM pipeline.

A complete line-based SfM pipeline is presented in [ZK14]. They use their proposed Line Band Descriptor (LBD) [ZK13] for line matching and their Robust Perspective-n-Line (RPnL) algorithm [ZXL+12] to estimate the camera pose from n correspondences between 3D features and the 2D observations. The LBD is calculated from the gradients within several band-like regions that are in the local environment and parallel to the line segment. In addition to the appearance, several geometric constraints between line pairs are used to improve the matching. The newly introduced Cayley representation for 3D lines allows a description with the minimum four parameters and simplifies optimization by bundle adjustment, since no additional constraints have to be enforced. The reconstruction begins

with an initial model from three images. Incrementally new images are added to this using 3D-2D correspondence.

Another purely line-based SfM pipeline is presented in [MW14]. The basic procedure is similar and begins with an initial reconstruction from three images, which is incrementally extended. But the approach used to match the line segments is completely different. They extract fixed sized SIFT features at the end points of the line segments and use them for matching. The method can therefore only be used for small camera motions [Hof16]. The authors' evaluations show that the trajectories can be estimated as good as with classical point-based SfM methods and at the same time more meaningful wiry 3D models can be reconstructed for indoor scenes.

Because point-based SfM methods are widely used and enable reliable and robust post estimation, many approaches use a point-based reconstruction as a prerequisite. If the camera positions are known, line-based reconstruction is substantially simplified and is used as a post-processing step to improve the SfM results with 3D lines.

Many of the approaches, which assume that the camera poses are known, are concerned with the reconstruction of building outlines from aerial image sequences. Such a method is presented in [HF01]. For matching and reconstructing the line segments, they use only geometric conditions.

Another method that belongs to the group of approaches that assume the camera poses as given is presented in [JKT+10]. They establish connections between neighboring 3D lines and can improve reconstruction quality through these additional geometric conditions. A special aspect of this method is that no explicit matching of 2D line segments is performed. Instead, for each 2D line segment, all possible 3D hypotheses are analyzed by projecting them into neighboring images and evaluating them based on the gradients in the image. This allows reconstructions of scenes with high complexity, e. g., due to strong illumination changes. At the same time, this approach results in an incredibly high computational complexity, so that the calculation takes several hours even for small data sets.

To solve this issue, [Hof16] proposes a combination of 3D hypothesis evaluation and geometric line matching. They propose a method that determines a set of matches for each 2D line segment using weak epipolar

constraints. 3D hypotheses are only generated and evaluated for this limited set. This approach can significantly accelerate the performance. At the same time, they can show that the procedure has no negative effect on the completeness or accuracy of the reconstruction.

2.4 Reconstruction of Higher-level Primitives

A number of methods that are concerned with line-based reconstruction use the lines as starting point to build a piecewise planar 3D model of the scene. These methods usually focus on the reconstruction of buildings. Using 3D lines to create planes has advantages compared to 3D points, since two 3D lines are sufficient to create a plane hypothesis and the reconstructed lines normally represent the intersection of two 3D planes.

The method presented in [WZ02] uses the reconstructed lines to determine the main directions of the scene, which are used to detect the dominant planes.

A similar procedure is proposed by [SSS09]. Starting from a sparse 3D reconstruction of points and lines, planes are extracted and then piecewise planar depth maps are generated by graph-cut based minimization.

There are currently only a small number of methods for reconstructing curves. The reason for this is that the matching and reconstruction of curves is much more complex than with line segments. Most existing methods are generally not applicable to arbitrary datasets, but are specialized for a specific application, e. g., the reconstruction of wire art [LCL+17].

In [FK10], a framework for multi-view reconstruction of curves is presented, which enables the reconstruction of 3D curve sketches as a supplement to point-based reconstruction.

An approach for a curve-based SfM method is proposed by [NF15]. They use splines to describe the 3D curves and curve-based bundle adjustment to optimize points, curves, and camera poses simultaneously. A limitation of the work is that the correspondences between the image curves are assumed to be known.

2.5 Facade Image Interpretation

The final goal of urban 3D reconstruction is to create detailed and semantically meaningful 3D models of the buildings. For this reason, automatic interpretation and reconstruction of building facades has been researched for many years. A comprehensive overview of the research topics in the field of urban reconstruction is given in [MWA+13].

One of the first methods enabling automatic detection and reconstruction of windows from single view rectified images was presented in [LR04]. They take advantage of the regularity of the vertical and horizontal window arrangement for detection.

In the following years, a lot of work was done on automatic window detection. Most approaches require prior knowledge [Wen16]. Often, either assumptions of a typical grid structure of windows in facades [LR04; WF08; RL10] or assumptions about the geometry and appearance of windows [CS08] are used.

The method presented in [TŠ11] offers comparatively high flexibility with regard to the grid structure of the windows. The authors use stochastic grammar with pairwise attribute constraints to detect windows in single rectified images.

There are different methods to detect regions of interest, such as windows, in images [Wen16]. The two most important techniques are gradient projection to find aligned edges [LR04; RL10] and the use of classifiers that detect regions of interest within the image [ASJ+07; WF08]. The gradient projection uses the fact that the windows in the facades are usually aligned horizontally and vertically. Detection using a classifier is more tolerant of perspective distortions or loose structure of the facade elements, but often achieves worse results in terms of completeness [Wen16].

Mayer and Reznik [MR05] introduce a method that enables the recognition of windows from image sequences. The authors train a window classifier and use the Markov chain Monte Carlo (MCMC) method with an abstraction hierarchy for detection. First, they determine the facade planes by means of robust least squares matching. The recognition of window hypotheses is then done by learning an implicit shape model. In the subsequent work [RM08], the original approach was extended by a validation by self-diagnosis based on the similarity of windows and model

selection to determine the most suitable configuration regarding vertical and horizontal arrangement of the windows.

A method for the detection of windows and doors in dense 3D point clouds of facades is presented in [NDM14]. The method is based on Monte Carlo Simulation. Templates containing control points of curves are used for possible shapes of windows and doors and a 2D shape-space is introduced to match similar shapes.

Another group of methods aims to detect not only windows but also other facade objects [MMW+12; CSP14; KGZ+15; GMP16]. For this purpose, pixelwise labelling or facade segmentation is performed. Typical classes are window, wall, balcony, door, roof, chimney, sky and shop.

Cohen et al. [CSP14] propose a method that uses dynamic programming, but also strong architectural constraints for facade interpretation, to achieve consistent results.

In [MMW+12], weak architectural prior knowledge is used to ensure that the reconstruction results are architecturally plausible and consistent. The authors use a three-layer approach for semantic segmentation of building facades. Starting from an oversegmentation of the facade, probabilistic interpretations for each segment are generated using Recursive Neural Networks (RNN). These labels are merged with the output of facade component detectors. In their subsequent work [MMV16], the process is improved. In the bottom layer they evaluate different segmentation and classification algorithms and in the other layers optimizations are introduced.

Koziński et al. [KGZ+15] use architectural prior knowledge, which they transfer into the structure of a Markov Random Field (MRF). By the additional handling of occlusions, they are able to restore partially occluded facades.

Gadde et al. [GMP16] propose a method for automatically learning grammars for facade segmentation from images with ground truth annotations. The authors use split grammars and first generate a large set of rules from the training set, which is then compressed and generalized.

An approach for the interpretation of facades based on a Convolutional Network is presented in [SM16]. The network is trained with patches of images and the facades are classified into wall, window and door.

Recently, Rahmani and Mayer [RM18] proposed a method that uses

Structured Random Forest (SRF), Region Proposal Network (RPN) based on a Convolutional Neural Network (CNN) and rectangular fitting to provide high-quality semantic segmentation of building facades. The authors use the features created by the RPN as channels in the SRF. Furthermore, architectural constraints are used, e. g., that the facade objects have a rectangular shape and the roof and wall are separated by a horizontal line.

In the recent work of Gadde et al. [GJM+18], a method for facade segmentation for 2D images and 3D point clouds is presented. The authors propose a generic framework that uses auto-context features. The approach does not require strong architectural prior knowledge and achieves high-quality segmentation results.

2.6 Summary

In this chapter, we have given an overview of the different areas of feature detection in images and presented the most important methods for each area. Furthermore, we have presented the related fields in the area of three-dimensional reconstruction, which are able to reconstruct a three-dimensional scene based on the detected features and depend on a robust and accurate detection of the relevant image features. Finally, we have provided a brief overview of the research work in the related field of facade interpretation, which aims to create semantically meaningful models of buildings.

In the following chapter, the ideas and the individual process steps of the approach developed by us are presented in detail.

Chapter 3

Robust and Accurate Detection of Image Features

In this chapter, we present our combined detection method for points, lines and arcs. First, we describe the basics and prerequisites. Then the individual steps of the method for detecting the image features are explained in detail.

3.1 Prerequisites

3.1.1 Camera Model

Camera models are used to mathematically describe the physical properties of cameras. They describe the mathematical relationship between a point in three-dimensional space and its projection onto the image plane.

Pinhole Camera Model

The pinhole camera model is the simplest mathematical description of a physical camera. The aperture of the camera is a point and no lenses are used. This is called an ideal pinhole camera.

All points in three-dimensional space are projected along a straight line through the projection center onto the image plane. This type of projection is called perspective projection. By projecting a three-dimensional point in space onto a two-dimensional point on the image plane, one dimension is lost. From a projected point in the image, therefore, no point in three-dimensional space can be reconstructed, but an infinite ray on which it lies.

Figure 3.1 shows the geometric relationship between a point M in three-dimensional space and its projection m onto the image plane. Unlike a real pinhole camera, the projection center C is located behind the image plane, as this allows a simpler description without changing the perspective properties. The origin of the camera coordinate system (x_c, y_c, z_c) is the projection center and the z_c-axis corresponds to the optical axis of the camera. The distance between the projection center and the image plane is called the focal length f. The image plane is parallel to the $x_c y_c$-plane. The intersection of the optical axis with the image plane is called the principal point p.

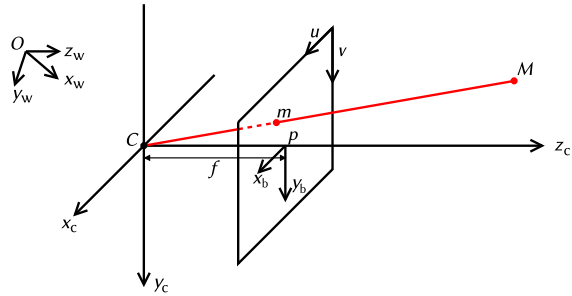


Figure 3.1. Geometry of a pinhole camera.

Camera Parameters

The camera parameters allow the calculation of the projection of a three-dimensional point in space in the world coordinate system (x_w, y_w, z_w) onto a point in the pixel-based image coordinate system (u, v). A distinction is made here between the intrinsic and the extrinsic camera parameters.

The extrinsic camera parameters determine the spatial position and orientation of the camera in the world coordinate system. These are described by a rotation matrix R, which determines the orientation of the camera, and a translation vector $\vec{C}$, which indicates the origin of the camera coordinate system in the world coordinate system. Together they are referred to as a pose of the camera.

The homogeneous world coordinates $\widetilde{M}_{\mathrm{w}}$ of a three-dimensional point in space can be converted via Equation 3.1.1 into the homogeneous coordinates $\widetilde{M}_{\mathrm{c}}$ of the camera coordinate system.

$$\widetilde{M}_{\mathrm{c}} = \begin{bmatrix} R^{\mathrm{T}} & -R^{\mathrm{T}}\vec{C} \\ \vec{0}_3^{\mathrm{T}} & 1 \end{bmatrix} \cdot \widetilde{M}_{\mathrm{w}} \tag{3.1.1}$$

The intrinsic camera parameters describe the internal geometric properties of the camera and allow the projection of the three-dimensional points in the camera coordinate system onto the two-dimensional image coordinate system. In a digital camera, they are determined by the focal length f, the pixel coordinates of the principal point $p = \left(p_x, p_y\right)^{\mathrm{T}}$, and the pixel scaling in the horizontal k_x and vertical k_y directions. The pixel scaling is used to convert the metric coordinates into pixel coordinates. The intrinsic parameters are summarized in the calibration matrix K (Equation 3.1.2).

$$K = \begin{bmatrix} f \cdot k_x & 0 & p_x \\ 0 & f \cdot k_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1.2}$$

A point in homogeneous three-dimensional camera coordinates can then be converted via Equation 3.1.3 into homogeneous pixel coordinates.

$$\widetilde{m}_{\mathrm{p}} = \begin{bmatrix} K & \vec{0}_3 \end{bmatrix} \cdot \widetilde{M}_{\mathrm{c}} \tag{3.1.3}$$

When projecting a scene four coordinate systems must be considered. First, a point in the world coordinate system has to be converted to the three-dimensional camera coordinate system using the extrinsic camera parameters. Then, a perspective projection is made into the two-dimensional normalized image coordinate system with the origin at the principal point of the image. Subsequently, the transformation into the pixel-based image coordinate system with the origin in the upper left corner of the image takes place using the intrinsic parameters.

The entire transformation from the world coordinate system to the pixel coordinate system can be combined into a single matrix multiplication using homogeneous coordinates. Using the projection matrix P given in Equation 3.1.4, a homogeneous point $\widetilde{M}_{\mathrm{w}} = \left(X, Y, Z, 1\right)^{\mathrm{T}}$ in world coordinates can be projected to a homogeneous point $\widetilde{m}_{\mathrm{p}} = \left(x, y, w\right)^{\mathrm{T}}$ in

pixel coordinates. The searched pixel coordinates in the image are then $m_\mathrm{p} = (x/w, y/w)^\mathrm{T}$.

$$\widetilde{m}_p = P \cdot \widetilde{M}_\mathrm{w} \quad \text{with} \quad P = K \cdot \begin{bmatrix} R^\mathrm{T} & -R^\mathrm{T}\vec{C} \end{bmatrix} \tag{3.1.4}$$

Distortion

Real lens systems do not match the ideal pinhole camera model. The term distortion is used in optics to describe optical aberrations caused by the deviation from the ideal perspective projection. The distortion causes straight lines in the scene to be no longer straight lines in the image.

In image processing, methods have been developed to model and calculate the distortion of a camera lens [Con19; Bro66]. By inverting the model, the distortion of an image can be compensated and a software-corrected image without distortion can be calculated.

The two main types of distortion are radial and tangential distortion. The curvature of the lens leads to nonlinear distortions, especially with small focal lengths. As the radial distance from the focal point increases, a non-linear mapping occurs, causing the straight line to appear curved. This distortion is rotationally symmetric and is therefore called radial distortion. If the camera lens is not aligned completely parallel to the image plane, it can lead to a tangential distortion.

By introducing an additional nonlinear transformation, the distortion can be generally described. In normalized image coordinates, the transformation of the undistorted coordinates $m_u = (x_u, y_u)^\mathrm{T}$ into the distorted coordinates $m_d = (x_d, y_d)^\mathrm{T}$ can be described as follows:

$$\begin{aligned} \begin{bmatrix} x_d \\ y_d \end{bmatrix} = & \underbrace{(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \begin{bmatrix} x_u \\ y_u \end{bmatrix}}_{\text{radial distortion}} \\ & + \underbrace{\begin{bmatrix} 2\kappa_4 x_u y_u + \kappa_5 (r^2 + 2x_u^2) \\ 2\kappa_5 x_u y_u + \kappa_4 (r^2 + 2y_u^2) \end{bmatrix}}_{\text{tangential distortion}}, \end{aligned} \tag{3.1.5}$$

where $r = \sqrt{x_u^2 + y_u^2}$ is the distance to the principal point. The five parameters κ_1, κ_2, κ_3, κ_4 and κ_5 are called distortion coefficients. For normal

distortions, the coefficients κ_1 and κ_2 dominate. Therefore, the remaining parameters κ_3, κ_4 and κ_5 are often assumed to be zero.

The mapping from the distorted image coordinate system into the pixel coordinate system then takes place via the calibration matrix K as above:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = K \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{3.1.6}$$

Generic Camera Model for Wide-Angle and Fisheye Lenses

The pinhole camera model with the lens distortion extension is suitable for modeling most conventional cameras with narrow-angle or even wide-angle lenses. However, fisheye cameras cannot be modeled satisfactorily.

A fisheye lens has a very large field of view. The lens has a strong visual distortion to create wide panoramic or hemispherical image. Straight lines that do not run through the center of the image are mapped curved. A fish-eye lens usually reproduces area ratios more faithfully than a conventional, gnomonically projecting wide-angle lens.

Since it is not possible to project a hemispheric field of view onto a finite image plane using a perspective projection, a different projection model is needed to model fisheye lenses.

For fisheye lenses, we introduce the camera model proposed by [KB06]. This is a generic camera model for fisheye lenses and conventional wide-angle lenses.

The imaging properties of a lens can be described using the mapping function. The distance r of an image point from the principal point depends on the focal length f and the angle θ from the optical axis (Figure 3.2 on the next page). In the case of a perspective projection of a pinhole camera, the following applies:

$$r = f \tan\theta \tag{3.1.7}$$

Fisheye lenses can have different mapping functions: These are stereographic projection (Equation 3.1.8), equidistant projection (Equation 3.1.9), equisolid angle projection (Equation 3.1.10) and orthographic projection

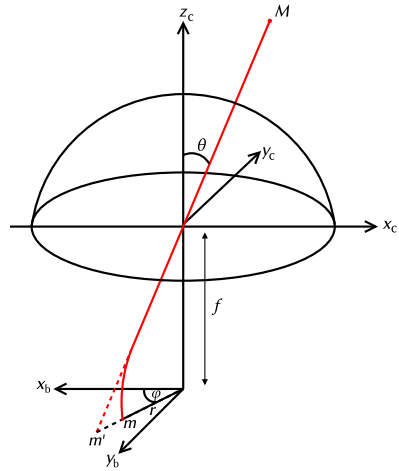


Figure 3.2. Geometry of a fisheye camera (adapted from [KB06]). The projection of the point M is m and with a pinhole camera it would be m'.

(Equation 3.1.11).

$$r = 2f \tan(\theta/2) \tag{3.1.8}$$

$$r = f\theta \tag{3.1.9}$$

$$r = 2f \sin(\theta/2) \tag{3.1.10}$$

$$r = f \sin(\theta) \tag{3.1.11}$$

The most common projection model is the equidistant projection (Equation 3.1.9). Since the imaging functions of real fish eyes deviate from the fundamental projection models, the general form is used for the projections:

$$r(\theta) = \kappa_1\theta + \kappa_2\theta^3 + \kappa_3\theta^5 + \kappa_4\theta^7 + \kappa_4\theta^9 \tag{3.1.12}$$

Here, only the first five terms are used, as these are usually sufficient to approximate the different projection curves with adequate accuracy [KB06]. For the mapping of a three-dimensional point M over the incoming ray onto the normalized image coordinates $m = (x_d, y_d)$, the following applies:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = r(\theta) \begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix} \tag{3.1.13}$$

where $\Phi = (\theta, \varphi)$ is the direction of the incoming ray and $r(\theta)$ is given by Equation 3.1.12. For real lenses, the function $r(\theta)$ increases monotonically in the interval $[0, \theta_{\max}]$. Here $\theta_{\max}$ corresponds to the maximum viewing angle.

3.1.2 Modulation Transfer Function

When an object is captured with a real optical system, inevitable aberrations and diffraction phenomena lead to a reduction in the quality of the resulting image. The modulation transfer function (MTF) describes the resolution performance of an optical system by the ratio of the relative image contrast to the relative object contrast. The MTF is used for the objective evaluation of the imaging performance of optical systems. Based on the MTF, system performance and image quality can be reliably predicted. To describe the quality of an imaging system, the terms resolution and contrast are noteworthy.

Resolution

Resolution is the ability of an imaging system to distinguish closely spaced object details. The resolution is often expressed by the number of line pairs per millimeter that can be resolved separately. For the measurement, special test targets are used, which show a series of alternating white and black stripes with the same distance. When imaging such a test target with an optical system, the perfect edges of the stripes are blurred to some degree (Figure 3.3 on the following page).

Contrast

Contrast refers to the difference between light and dark areas of an image. Mathematically, the contrast M is calculated as follows:

$$M = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}} \tag{3.1.14}$$

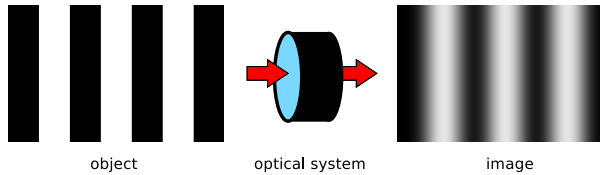


Figure 3.3. Test target for determining the resolution of an optical system (left) and the image of the target through the optics (right).

where $I_{\max}$ and $I_{\min}$ represent the highest and lowest luminance. This definition of contrast is also known as modulation because it is an effective way to quantify contrast for periodic functions. For the test target from Figure 3.3, the contrast describes how faithfully the minimum and maximum intensity values of the object are transferred to the image.

If an imaging lens with the same resolution is used to image objects with different line-pair frequencies, it can be seen that as the spatial frequency of the lines increases, the contrast of the image decreases. The MTF generally describes the contrast as a function of the resolution.

3.1.3 Image Noise

Electronic image sensors, such as CCD and CMOS sensors, suffer from image noise, resulting in deterioration of a captured image due to unwanted disturbance unrelated to the actual image signal. The noise of image sensors is varies greatly, as it is caused by several different effects.

The linear camera model, which is defined in the EMVA 1288 standard [Eur16], allows a description of the relationship between light incidence at the sensor and the resulting image signal. The input values are the mean value and variance of the number of photons integrated over the exposure time and the output are the mean value and the variance of the digital image signal.

The camera's physical model (Figure 3.4 on the facing page) is based on the assumption that a number of the photons hitting the pixel area during the exposure time produce a number of electrons that form a charge which is converted to a voltage. This is then amplified and digitized, resulting in a digital gray value.

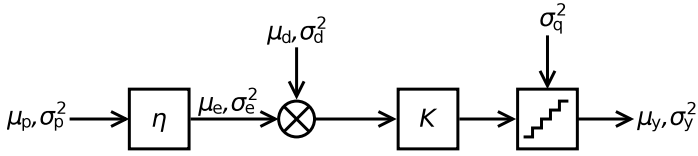


Figure 3.4. Linear camera model (adapted from [Eur16]). Here, μ_p is the number of photons hitting the pixel area during exposure time, η the quantum efficiency, μ_e the number of electrons, μ_d the dark signal, K the system gain, σ_q^2 the quantization noise and μ_y the digital gray value.

The model has only three parameters, the quantum efficiency η, the dark noise σ_d and the system gain K. The quantum efficiency η describes the fraction of the average number of photons μ_p incident during the exposure time, which are absorbed and converted into μ_e electrons:

$$\eta(\lambda) = \frac{\mu_e}{\mu_p} \tag{3.1.15}$$

where λ is the wavelength of the incident light. Over the exposure time, the dark signal μ_d is added, which is caused by thermally induced charge carriers and leakage currents. The entire signal is amplified and quantized by an analog to digital converter to a gray value:

$$\mu_y = K(\mu_d + \mu_e) = \mu_{y.dark} + K\mu_e \tag{3.1.16}$$

The linear camera model takes into account three sources of noise: photon noise, dark noise, and quantization noise. The photon noise describes the statistical fluctuation of the incident photons and is Poisson distributed. The variance is equal to the mean. The same applies to the variance of the electrons accumulated over the exposure time.

$$\sigma_e^2 = \mu_e \tag{3.1.17}$$

All noise sources that are caused by the transport or the amplification of the charge carriers are combined into a single noise source, the dark noise σ_d^2. This noise is amplified by the gain K. After amplification, only the quantization noise σ_q^2 is added. With sufficiently small quantization intervals, it can be considered as uniform distributed.

Since the variances of different noise sources add up linearly according

to the error propagation law, the following overall noise results:

$$\sigma_y^2 = K^2(\sigma_d^2 + \sigma_e^2) + \sigma_q^2 \tag{3.1.18}$$

$$= K^2\sigma_d^2 + \sigma_q^2 + K(\mu_y - \mu_{y.dark}) \tag{3.1.19}$$

Therefore, there is a linear relation between the total temporal variance of the digital signal and the mean photon-induced gray value.

All unknown parameters, the quantum efficiency η, the dark noise σ_d and the system gain K, can be determined experimentally for real camera sensors. This makes it possible for a known camera sensor to specify the expected noise level for each pixel as a function of the gray value.

3.2 Detection of Points, Lines and Arcs

The focus of our work is the extraction of image features in urban and man-made environments, which on the one hand are accurate and on the other hand describe the relevant geometric primitives.

The different image features are closely related (see Section 2.1) and have advantages and disadvantages for use in urban scenes. Robust interest point detectors, such as SIFT [Low99] or SURF [BET+08], enable reliable wide-baseline matching of features across multiple images, but generally do not have high localization accuracy and are usually not located at physical object corners.

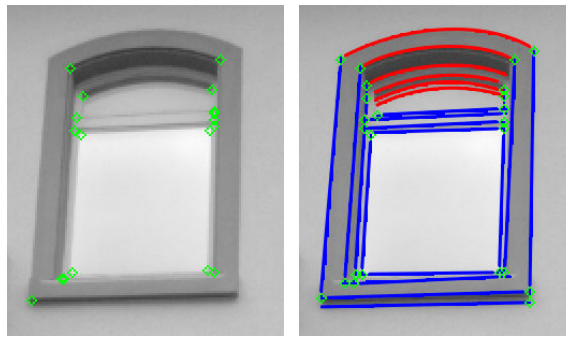
Classical corner detectors, e. g., Shi-Tomasi corner detector [ST94], have higher localization accuracy but are also often found in non-relevant locations, e. g., on structured surfaces or on trees. They are often missing on man-made structures which do not have a high corner strength. These may be e. g., windows in the shadow. In addition, the localization accuracy of these points deteriorates in the presence of noise.

Edge and line segments offer a higher level of structure information about the scene and are usually located in relevant places in urban scenes. However, start and end points of line features are often only inaccurately localized.

Corner points generated by the intersecting of line segments have a higher localization accuracy than classical point features and a greater

robustness against noise. They are usually located at relevant geometric structures in urban scenes.

Combined detection of arc segments, line segments and points by calculating intersections offer many possibilities. A higher localization accuracy and greater relevance of the point features can be achieved. At the same time, accurate end points for the line and arc segments can be determined in this way. These end points can be used to support triangulation of the line segments.



(a) Corner detection **(b)** Intersections

Figure 3.5. Comparison between a classical corner detector and the detection of arcs, lines and intersections.

Figure 3.5 shows a comparison of the detection of points with the Shi-Tomasi corner detector [ST94] and the detection of the intersections of detected line and arc segments. The structure of the window can be better represented by the line and arc segments. At the same time, the intersections are located at the real corner points of the window.

The topology between the different image features can be used to support and verify the matching of multiple images in a structure from motion application. Image features, which are connected in one image, are probably also connected in other images. Simple contours or polygons can thus be recognized and reconstructed directly. In this way, a highly abstract but at the same time semantically rich description of the image can be generated.

3.2.1 Overview of the Detection Process

Our approach to extracting image features consists of a multi-step process (Figure 3.6). In the first step, edges are detected in the images. Subsequently, geometric primitives, such as line and arc segments are fitted to the extracted edges. The intersections between the geometric primitives are calculated to extract accurate point features. At the same time, a graph is constructed which describes the topology of the image features. The steps of the detection process are described in more detail in the following sections.

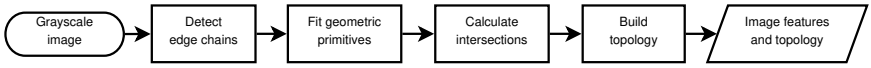


Figure 3.6. Flowchart of the detection process.

3.2.2 Extraction of Edge Chains

To extract the edges, we use a customized version of the Edge Drawing method [TA12]. Edge Drawing is a differentiation-based edge detection algorithm that runs in real-time and produces contiguous pixel chains that are exactly one pixel wide.

The Edge Drawing algorithm consists of four steps. In the first step, noise is reduced by using a Gaussian filter; then the gradient magnitude and gradient direction are calculated for each pixel. In the next step, anchor points are extracted. These are the peaks in the gradient image. These peaks are linked using smart routing to extract the edges.

Figure 3.7 on the next page shows the steps of the process using an example. As is customary in the case of the differentiation-based edge detection methods, the image is first smoothed with a Gaussian filter in order to suppress noise. We use a 5×5 filter kernel with $\sigma = 1$, as suggested by the authors of the original paper.

In the next step, the gradient magnitude G and the gradient direction D are determined for each pixel. For this purpose, the horizontal and vertical gradients, G_x and G_y, are determined using the Sobel operator.

(a) Smoothed image **(b)** Gradient map **(c)** Anchor points **(d)** Edge chains

Figure 3.7. Process flow of the edge extraction with Edge Drawing.

The direction-independent gradient magnitude is calculated by combining both results:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.2.1}$$

To simplify the calculation, the absolute values of the two directional gradients are used instead of the root.

$$G = |G_x| + |G_y| \tag{3.2.2}$$

When calculating the edge direction, only a distinction is made between horizontal and vertical direction. For this purpose, the gradient in vertical and in horizontal direction is compared per pixel.

$$D = \begin{cases} 0 & \text{for } |G_x| \geqslant |G_y| \\ 1 & \text{else} \end{cases} \tag{3.2.3}$$

The restriction to only two directions leads to a significant acceleration of the routing process without a deterioration of the results [TA12]. To suppress clutter caused by image noise or slight texture changes in homogeneous regions, a fixed threshold is applied to the gradient map. The gradient magnitude of all pixels below this threshold is set to zero.

In the next step, the points at which the extraction of the edge chains begins are determined. These are called anchor points. Anchor points are simply the local maxima of the gradient map (Figure 3.7 (b)).

Finally, the extracted anchor points are linked together via the ridges in the gradient map during the smart routing process. For our application, we have adapted the smart routing process in such a way that not only individual contiguous edge chains are extracted, but also the connecting

points between the individual edge chains are stored. The background for this adjustment is that it allows us to determine the connections between the different edges, as we need them in a subsequent processing step to handle the junctions between different edge chains.

Another reason is that during the extraction of edge chains, these sometimes tear off by sharp changes of the direction of the gradient or the edge kinks through a textured background (Figure 3.8). These cases can later be resolved by examining the stored connections between different edge chains.

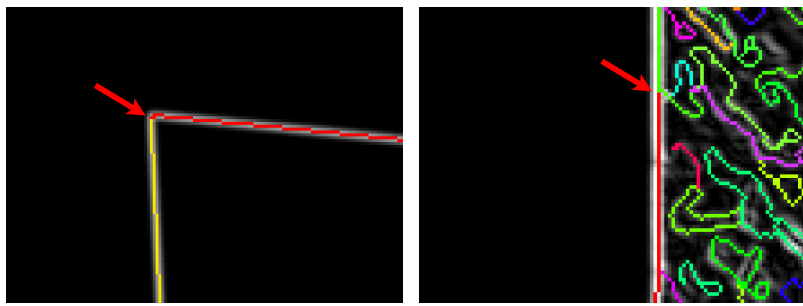


Figure 3.8. Examples of extracted edge chains in real images where connected edge chains have actually been extracted separately (marked with a red arrow).

The adapted smart routing process is as follows: Beginning with an anchor point, the direction of the gradient is considered. The pixel with the strongest gradient magnitude is then selected orthogonally to the gradient direction and used to iteratively build up a pixel chain. The process is terminated if all neighboring pixels have a gradient magnitude value of zero or a previously detected edge pixel is reached. In the latter case, we also store the connection point for the two edges.

Figure 3.9 (a) illustrates the sequence of the routing process on the basis of a small cutout of a gradient map. The numbers are the intensity values of the pixels. The extracted anchor points are highlighted in red and the created chain of edge pixels in yellow. The process starts at the upper right anchor point. Since the pixel has a horizontal gradient direction, the structure of the chain runs from the pixel to the left and right. The

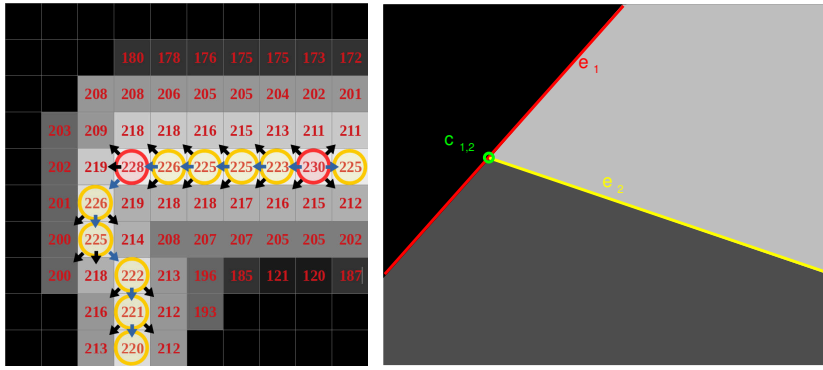


Figure 3.9. (a) Illustration of the smart routing process (adapted from [TA12]) and (b) detected edge chains and connections of a simple example image.

arrows indicate the neighboring pixels that are selected based on the gradient direction of the pixel and compared in the smart routing process to determine the next pixel of the edge chain. The pixel with the largest gradient magnitude is always selected.

As a final result, our customized Edge Drawing algorithm provides a set of chains of edge pixels L and a set of tuples J consisting of a junction point and references to the two edge chains that join up at that point.

For the simple example image in Figure 3.9 (b), in which two edge chains, e_1 and e_2, have been extracted and one junction at point $c_{1,2}$, the result of the edge detection is:

$$L = \{e_1, e_2\} \tag{3.2.4}$$

$$J = \{(c_{1,2}, \widetilde{e_1}, \widetilde{e_2})\} \tag{3.2.5}$$

where $\widetilde{e_1}$ and $\widetilde{e_2}$ are the references to the corresponding edge chains.

Overall, one thus obtains a set of exactly one-pixel wide chains of edge pixels as well as the connection point between the individual chains.

3.2.3 Fitting the Geometric Primitives

In the next step, geometric primitives are fitted into the extracted edges. We use an approach similar to that used by [AT11]. Their approach allows the detection of line segments based on the edge chains extracted by Edge Drawing.

The process presented by us allows the extraction of different geometric primitives. We limit ourselves to lines and parabolas as these are the predominant forms in our application, which are urban scenes. However, an extension which allows the extraction of further geometric shapes, e. g., circles or ellipses, is easily possible. Parabolas were chosen as the model for the arcs because they are easy to parameterize and a robust fitting is possible.

The starting point for the detection of the lines and arcs are the edge chains from the first step. The extracted pixel chains are decomposed into one or more geometric primitives. The basic idea is to walk along the pixel chain and to fit geometric primitives.

If the edge has a minimum length, fitting an initial line segment is attempted using a least squares line fitting method. For this purpose, the model function is first selected based on the gradient direction prevailing in the current edge segment.

Equation 3.2.6 is used for horizontal edges and Equation 3.2.7 for vertical edges. This distinction is necessary because Equation 3.2.6 is not suitable for representing lines that are parallel to the y-axis.

$$y = \alpha_0 + \alpha_1 x \tag{3.2.6}$$

$$x = \alpha_0 + \alpha_1 y \tag{3.2.7}$$

The data points to which the function is fitted are the first n pixel coordinates $p_i = (x_i, y_i)$, $i = 1, ..., n$ of the edge chain. The fit of the model to the data points is measured by the residuals, which are defined as differences between the values of the model function and the data. The least squares method finds the optimal parameter values by minimizing the sum of squared residuals. For the model function of Equation 3.2.6, therefore, the parameter α_0 and α_1 are searched with the smallest sum of

squared residuals:

$$\min_{\alpha_0,\alpha_1} \sum_{i=1}^{n} (\alpha_0 + \alpha_1 x_i - y_i)^2 \tag{3.2.8}$$

If an initial line segment was found where the mean error is below the specified threshold value, it is tried to extend this line segment. For this purpose, further pixels of the edge chain are added to the line until the deviation exceeds a threshold value.

If a smaller deviation is achieved when fitting a parabola than when fitting a line, the parabola is selected as model. The corresponding model functions for a parabola are:

$$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \tag{3.2.9}$$

$$x = \alpha_0 + \alpha_1 y + \alpha_2 y^2 \tag{3.2.10}$$

When selecting a parabola as model the curvature of the parabola is additionally checked. The parabola is selected only if it has a sufficient curvature. Otherwise very flat parabolas could be fitted into line segments, i. e., with $\alpha_2 \approx 0$.

The curvature of the point of a curve can be determined by the radius of the osculating circle. The osculating circle of a point p on a curve is defined as the circle that best approximates the curve at that point. The circle passes through p and another two points on the curve infinitesimal close to p. The radius of curvature r of a function f at the location x can be calculated as follows:

$$r(x) = \left| \frac{\left(1 + f'(x)^2\right)^{\frac{3}{2}}}{f''(x)} \right| \tag{3.2.11}$$

We determine the curvature of the parabolic segment at three points, the starting point, the center, and the end point of the parabolic segment. Only when the ratio of the mean curvature at the three points to the Euclidean distance between start and end point is less than a threshold, the model is switched from line to parabola. If the model is changed, the parabolic segment is extended by further pixels from the edge chain until the threshold for the maximum deviation is reached. The extracted element is then stored. Additional models such as ellipses and circles can

be considered at this point.

Similarly, the remaining pixels of the edge chain are processed to extract further geometric primitives. The complete algorithm for extraction of lines and arcs from the chains of edge pixels is summarized in Listing 3.1 in pseudocode.

Listing 3.1. Pseudocode for the extraction of geometric primitives.

```
ExtractLinesAndArcs:
  while number of edge pixels greater than or equal n
    fit line to first n pixels
    if residual of fit greater than threshold
      remove first edge pixel
    else
      add edge pixel until deviation from line is above threshold
      if mean deviation with model parabola smaller and curvature greater threshold
        add edge pixel until deviation from line is above threshold
        store arc
      else
        store line
      end
    end
  end
```

3.2.4 Refinement of the Geometric Primitives

So far, the detection of the edge pixel is only done with pixel accuracy. The line and parabola equations are, however, set as the best-fit through all discrete pixel positions so that the geometric primitives can be specified with subpixel positions.

In the detection step, only the pixels with the maximum gradient in the local environment are considered. If the local environment of the geometric primitive is taken into account, a higher detection accuracy could be achieved. Possible approaches include an analysis of the distribution of the gradient magnitude perpendicular to the primitive or an interpolation-based subpixel localization of each pixel before the fit. We use the second approach because it is independent of the model function, which is used for fitting.

For each pixel of the geometric primitive, the local environment (4-neighborhood or 8-neighborhood) is considered. The subpixel position is

the average of the positions of the pixels in the neighborhood weighted by the gradient magnitude of the pixels.

The subpixel refined position of a pixel $(u, v)^{\mathrm{T}}$ is calculated using the following formula:

$$\begin{aligned}\begin{pmatrix} u_{\mathrm{sub}} \\ v_{\mathrm{sub}} \end{pmatrix} = &\frac{g(u-1,v)}{g_{\mathrm{sum}}} \begin{pmatrix} u-1 \\ v \end{pmatrix} + \frac{g(u+1,v)}{g_{\mathrm{sum}}} \begin{pmatrix} u+1 \\ v \end{pmatrix} \\ &+ \frac{g(u,v)}{g_{\mathrm{sum}}} \begin{pmatrix} u \\ v \end{pmatrix} \\ &+ \frac{g(u,v-1)}{g_{\mathrm{sum}}} \begin{pmatrix} u \\ v-1 \end{pmatrix} + \frac{g(u,v+1)}{g_{\mathrm{sum}}} \begin{pmatrix} u \\ v+1 \end{pmatrix}\end{aligned} \tag{3.2.12}$$

$$\begin{aligned}g_{\mathrm{sum}} = &g(u-1,v) + g(u+1,v) + g(u,v) \\ &+ g(u,v-1) + g(u,v+1)\end{aligned} \tag{3.2.13}$$

where $g(u, v)$ is the gradient magnitude at the pixel position $(u, v)^{\mathrm{T}}$.

In this case, the 4-neighborhood is considered, in the same way the 8-neighborhood can be used for the weighting. These subpixel accurate positions are then used to refine the geometric primitive by re-fitting.

3.2.5 Calculation of Intersection Features

In the next step, the intersections of the extracted geometric primitives are calculated. Only the intersections of successive elements from the same edge chain are calculated. In this way, it is ensured that the elements used for intersecting are connected over a common edge and are thus, with a high probability, part of a contiguous object in the scene.

In addition, intersections are calculated between elements, which have been extracted from different edge chains, if the edges have a connection and the elements have been extracted in the region of the junction point. To determine the edges with connections and the corresponding junction points, the set of tuples J created in the customized Edge Drawing algorithm (see Section 3.2.2) is used.

When calculating the intersection, three cases must be distinguished: Intersections between two lines, intersections between parabolas and lines, and intersections between two parabolas.

For the intersection point $I = (I_x, I_y)$ of two line segments l_1 and l_2, where l_1 is defined by the two end points (x_1, y_1) and (x_2, y_2) and l_2 is defined by the end points (x_3, y_3) and (x_4, y_4), applies:

$$I_x = \frac{(x_4 - x_3)(x_2 y_1 - x_1 y_2) - (x_2 - x_1)(x_4 y_3 - x_3 y_4)}{(y_4 - y_3)(x_2 - x_1) - (y_2 - y_1)(x_4 - x_3)} \tag{3.2.14}$$

$$I_y = \frac{(y_1 - y_2)(x_4 y_3 - x_3 y_4) - (y_3 - y_4)(x_2 y_1 - x_1 y_2)}{(y_4 - y_3)(x_2 - x_1) - (y_2 - y_1)(x_4 - x_3)} \tag{3.2.15}$$

If the two lines are parallel or congruent, the denominator is zero.

For the intersection $I = (I_x, I_y)$ between a line $x = \alpha_0 + \alpha_1 y$ and a parabola $y = \beta_0 + \beta_1 x + \beta_2 x^2$, the following applies:

$$I_x = \begin{cases} \alpha_0 & \text{for } \alpha_1 = 0 \\ -\frac{\alpha_1\beta_1 - 1}{2\alpha_1\beta_2} \pm \sqrt{\left(\frac{\alpha_1\beta_1 - 1}{2\alpha_1\beta_2}\right)^2 - \frac{\alpha_1\beta_0 + \alpha_0}{\alpha_1\beta_2}} & \text{else} \end{cases} \tag{3.2.16}$$

To calculate I_y, the value I_x must be substituted into the parabola equation. If the line and parabola have more than one intersection point, the correct one must be determined from the end points of the primitives used for the intersection.

The last case is the intersection of two parabolas, $y = \beta_0 + \beta_1 x + \beta_2 x^2$ and $y = \gamma_0 + \gamma_1 x + \gamma_2 x^2$. Here the intersection can be determined as follows:

$$I_x = -\frac{\beta_1 - \gamma_1}{2(\beta_2 - \gamma_2)} \pm \sqrt{\left(\frac{\beta_1 - \gamma_1}{2(\beta_2 - \gamma_2)}\right)^2 - \frac{\beta_0 - \gamma_0}{\beta_2 - \gamma_2}} \tag{3.2.17}$$

It is important that arc segments are also extracted from the edges because the approximation of curved contours through line segments would lead to inaccurate intersections.

If three or more geometric primitives meet at one point, the mean of the pairwise intersection points is set as the joint intersection of all the primitives. This is to prevent multiple points at such corners.

In case of noise, lines or arcs might fragment into several subsegments, which are separately detected. We therefore check for successive line and arc segments whether they have similar parameters and the end points are close to each other. If this is the case, the elements are fused.

3.2.6 Building a Topology

The connections of the intersections and geometric primitives are stored in a graph $T = (V, E)$, which models the topological relations. The nodes V of the graph correspond to the image features, i. e., points, lines and arcs. The relationships between the image features are modeled over the edges E of the graph. It is a undirected graph without multiple edges. The number of nodes $n(T)$ of the graph corresponds to the sum of all detected points, lines and arcs.

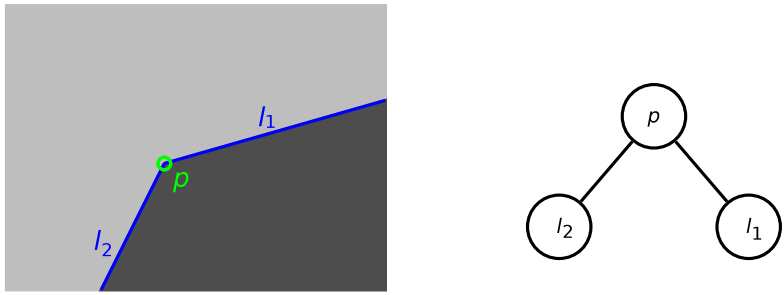


Figure 3.10. Simple sample image with two detected line segments, l_1 and l_2, and an intersection point p (left) and the corresponding graph describing the topology of the features (right).

Figure 3.10 shows the correlation between the detected image features and the generated graph, which describes their topological relationships, using a simple example. In the graph, the feature point p created by intersecting lines l_2 and l_2 corresponds to a node with label p and two edges to two nodes corresponding to the two lines l_1 and l_2.

Adjacent nodes in the graph always correspond to features of different types. This means that two nodes that represent lines are never directly connected to each other, but always via another node, that represents the intersection of the two lines.

In this way, geometric primitives that are more complex can be described over the individual elements. The connected components of the graph correspond to more complex geometric shapes in the image. If the connected component has a cycle, a closed contour is usually present in the image. This may be, e. g., a window in a facade.

3.2.7 Summary of the Detection Process

Figure 3.11 summarizes the steps of the detection process from the input image to the generated topology for a synthetic image.

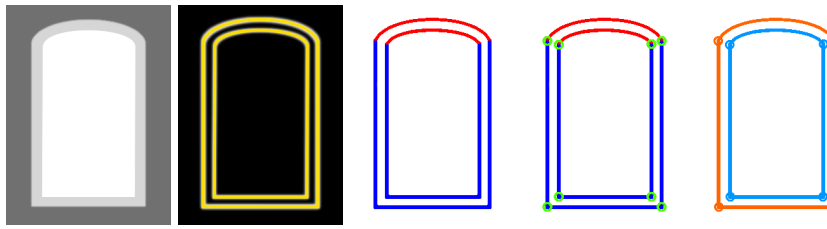


Figure 3.11. The steps of the detection process: (a) input image, (b) gradient image with extracted edge chains, (c) fitted geometric primitives (red: arcs, blue: line segments), (d) calculated intersections, (e) connected components of the graph (a random color for each component).

First, the input image is smoothed with a Gaussian filter to suppress noise (Figure 3.11 (a)). Then edge detection with Edge Drawing is performed. The extracted edge chains are marked yellow in Figure 3.11 (b). Geometric primitives are fitted into the edge chains (Figure 3.11 (c)) and the intersection point between them is calculated (Figure 3.11 (d)). At the same time, a graph is created that describes the topological relationships of the characteristics. In Figure 3.11 (e), the connected components of the graph are highlighted in different colors. As expected, both objects are recognized as different connected components of the graph.

3.3 Detection in Highly Distorted Images

3.3.1 Standard Methods

The standard methods used for detecting geometric primitives in images expect images without distortion as input. Especially wide-angle cameras introduce a variety of optical distortions. This causes the standard line

detection methods to fail because the lines are not straight. Therefore, the conventional approach to detect geometric primitives, especially lines, in distorted perspective or fisheye images is to correct those distortions by warping the image with a reverse distortion and use this software-corrected images as input (Figure 3.12).

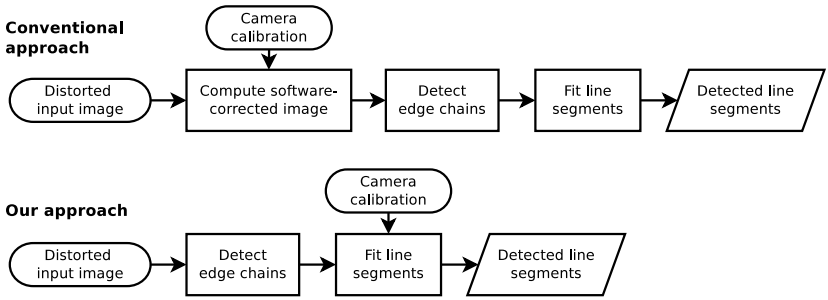


Figure 3.12. Application of primitive detection for distorted images.

To use a software-corrected image instead of the original image for the detection has many disadvantages and should be avoided. The reason for this is on the one hand the performance, because the image has to be warped, and on the other hand, the accuracy, because the original image is transformed e. g using a bilinear interpolation. Without adjusting the focal length, either the software-corrected fisheye image gets very large or large parts of the images are cut. An adjustment of the focal length means that the image center, which usually contains the relevant image content, is scaled down, so that information is lost here.

Another aspect is that errors in the calibration directly affect the detection accuracy of the lines, if the image is warped with an inaccurate reverse distortion. If the intrinsic camera parameters are corrected in a later processing step, for example, by a bundle adjustment, the complete detection including the software correction of all images and the calculation of the descriptors must be performed again.

For these reasons, we do not use a software-corrected image but use the original image as input to the line detection (Figure 3.12).

3.3.2 Adjustments of the Detection Process

In order to perform our detection approach to fisheye images, some adjustments are needed. The edge detection step extracts any contours so that it can be applied directly to distorted perspective images or fisheye images. However, for the line and arc fitting step undistorted pixel coordinates are needed.

We undistort the edge coordinates for the fitting step instead of fitting the lines with conic sections. We do this because in our case the calibrations of the cameras are already known and so any distortions can be considered. The approach can be used both for perspective images as well as fisheye images and it is ensured that only straight line segments are detected.

The camera models presented in Section 3.1.1 can be used to calculate a reverse distortion for distorted perspective and fisheye images. With this it is possible to convert the distorted pixel coordinates into undistorted normalized pixel coordinates.

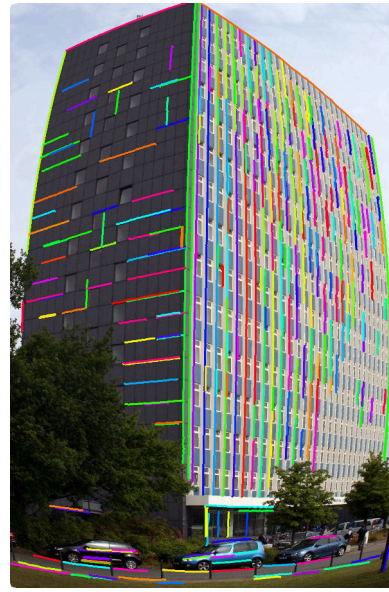


Figure 3.13. Detected features on a real image of the tower of Kiel University.

For the detection of geometric primitives in fisheye images, in our approach, the pixel coordinates are undistorted during the fitting step. All pixels of an edge chain are undistorted with the reverse distortion before the geometric primitives are fitted. The subsequent steps of the algorithm can remain unchanged. Figure 3.13 on the facing page shows the application of the detection method to a strongly distorted image. Despite the strong distortion, the lines in the image are reliably detected.

3.4 Detection of Faint Features in Noisy Images

A general problem with the construction of edge chains is that the pixel chains tear off due to image noise or weak gradients. At the same time, false edges can be caused by noise in homogeneous image regions.

Most edge detection methods use fixed threshold values to suppress weak edge pixels and thus false detection of image features. If the threshold value is set too high, faint edges are lost which are then below the threshold. If the threshold value is set too low, many wrong edges are created, e. g., by noise in homogeneous image regions.

The edge drawing approach uses a fixed lower threshold. The Canny Edge detector uses a hysteresis method, which improves the handling of noise. Some publications also deal with the automatic determination of threshold values, e. g., via histogram analysis (see Section 2.1.1). Usually, however, the same threshold values are used for the entire image.

Figure 3.14. Gradient maps with different threshold values for the minimum gradient magnitude. In order to improve the visualization, the images are contrast enhanced by gamma correction.

Figure 3.14 on the previous page shows the results of edge detection with different thresholds for the minimum gradient magnitude. A large number of edge pixels caused by image noise and the slight irregularities of the texture on the homogeneous wall surfaces and in the sky can be suppressed by increasing the threshold value. However, relevant structures are also lost as a result.

Therefore, current publications in the field of geometric primitive detection [AT11; PGG17] often use a validation step based on an a-contrario approach and the Helmholtz principle [DMM08] to control the false detections. The a-contrario approach evaluates each detected primitive based on its degree of structuredness in comparison to a stochastic model for unstructured data. The evaluation of the structuredness is carried out by the distribution of the gradient directions along the primitive. The noise is generally assumed to be constant over the entire image. This approach is computationally complex, since a large number of primitives has to be detected first in order to determine which ones are correct and which are false detections. In addition, the gradient directions of faint structures are often arbitrary due to image noise, so that these structures are erroneously removed.

We have therefore developed an approach that suppresses the edge pixels created by noise before the extraction of the geometric primitives and at the same time preserves faint structures. As described in Section 3.1.3, the noise of a camera depends on several factors. Our approach takes into account the noise level of the camera during feature detection.

3.4.1 Noise Model

The linear camera model described in Section 3.1.3 can be used to determine the expected standard deviation of the noise for each pixel in the image based on its intensity. The parameters of the model can be easily determined for a camera by taking images with different exposure times of a homogeneous surface, such as a light table [Eur16].

If the variance of the individual pixels of the intensity image is known, the variance of the gradient magnitude, which is caused by the camera noise, can be determined for each pixel in the gradient image. This value

can then be used to filter gradient images to suppress edge pixels caused by noise.

For the variance of the weighted sum of uncorrelated random variables $X = a_1X_1 + \ldots + a_nX_n$ applies generally:

$$\mathrm{Var}\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n} a_i^2 \mathrm{Var}(X_i) \tag{3.4.1}$$

Therefore, the following applies to the variance of the gradient approximations calculated with the Sobel operator for a pixel $P_{i,j}$ at position i, j in the image:

$$\begin{aligned}\mathrm{Var}(G_x(i,j))) = {} & \mathrm{Var}(P_{i-1,j-1}) + 4 \cdot \mathrm{Var}(P_{i-1,j}) + \mathrm{Var}(P_{i-1,j+1}) \\ & + \mathrm{Var}(P_{i+1,j-1}) + 4 \cdot \mathrm{Var}(P_{i+1,j}) + \mathrm{Var}(P_{i+1,j+1})\end{aligned} \tag{3.4.2}$$

$$\begin{aligned}\mathrm{Var}(G_y(i,j))) = {} & \mathrm{Var}(P_{i-1,j-1}) + 4 \cdot \mathrm{Var}(P_{i,j-1}) + \mathrm{Var}(P_{i+1,j-1}) \\ & + \mathrm{Var}(P_{i-1,j+1}) + 4 \cdot \mathrm{Var}(P_{i,j+1}) + \mathrm{Var}(P_{i+1,j+1})\end{aligned} \tag{3.4.3}$$

For the gradient magnitude of a pixel determined by Equation 3.2.2, the variance can be approximated as follows.

$$\mathrm{Var}(G(i,j))) = \mathrm{Var}(G_x(i,j))) + \mathrm{Var}(G_y(i,j))) \tag{3.4.4}$$

This is only an approximation, since the variances of the gradient approximation, G_x and G_y, are not completely independent.

3.4.2 Integration into the Detection Process

In the detection process, the Edge Drawing step is adjusted so that no fixed threshold value is used to suppress clutter through noise in homogeneous regions. Instead, we compute a filtered gradient map G_F where each pixel is set to zero if it has a gradient magnitude smaller than twice the standard deviation.

$$G_\mathrm{F}(i,j) = \begin{cases} 0 & \text{for } G(i,j) < 2 \cdot \sqrt{\mathrm{Var}\left(G(i,j)\right)} \\ G(i,j) & \text{else} \end{cases} \tag{3.4.5}$$

If the noise is approximately Gaussian distributed, filtering with twice the standard deviation would suppress about 95% of the noise. The remaining

Figure 3.15. Gradient map after filtering based on the modeled image noise.

steps of the detection process do not need to be changed.

Figure 3.15 shows a gradient map filtered in this way. A comparison with the images in Figure 3.14 on page 51 shows that large parts of the edge pixels on the homogeneous wall surfaces and in the sky are suppressed, while at the same time the faint structures, e. g., the basement windows, are still present in the image.

3.5 Recognition of Repetitive Line Features

Another challenge are repetitive structures. If the detected features are to be used for wide-baseline reconstruction, they should be unambiguous in their local environment in order to enable a reliable determination of correspondences between image pairs. Repetitive structures do not fulfill this assumption. Therefore, they must be considered separately in the matching step.

Numerous approaches already exist in the literature that allow the recognition of repetitive structures (see Section 2.1.6). Many of these methods are specifically designed for urban environments and allow the detection of repetitive rectangles, e. g., windows in building facades. Often the images are rectified to the facades plane in a first step.

Repetitive line segments are a challenge for which, to our knowledge, there is no adequate solution yet. Such structures often occur in man-made environments, e. g., on roller shutters or balcony handrails. In

wide-baseline reconstruction, these structures lead to problems. If one of the lines is not correctly detected, this can cause all correspondence to be shifted by one element. The result is a faulty three-dimensional reconstruction of the line segments. Therefore, we would like to introduce a method that allows the detection of repetitive line segments.

To detect repetitive lines, we use both photometric and geometric properties and combine them in a recognition and clustering process.

3.5.1 Appearance Similarity

Line descriptors can be extracted and compared to determine the appearance similarity of lines. We use the Line Band Descriptor (LBD) [ZK13]. The LBD describes the line by the distribution of the gradient directions in band-like regions that are parallel to the line.

The LBD considers a rectangular region centered on the line. This region is divided into several bands that are parallel to the line. In addition, two directions are introduced which define a local coordinate system. These are on the one hand the direction of the line d_{L} and on the other hand the clockwise orthogonal direction $d_{\perp}$. The direction of the line is defined in such a way that $d_{\perp}$ corresponds to the global gradient direction of the line segment.

The local gradient directions are summed up in rows. A global as well as a local weighting function is applied to each row to make the descriptor robust against small shifts and to prevent abrupt changes of the descriptor due to boundary effects.

For each line of the image, we generate a LBD descriptor vector d. The distance δ between the descriptor vectors, d_r and d_q of two lines, l_r and l_q, is a measure of the appearance similarity of the lines:

$$\delta(l_r, l_q) = \left| d_r - d_q \right| \tag{3.5.1}$$

3.5.2 Geometric Similarity

In addition to the visual similarity of lines, the similarity can also be determined by the geometric properties (Figure 3.16 on the following page). We consider three geometric properties to measure the geometric

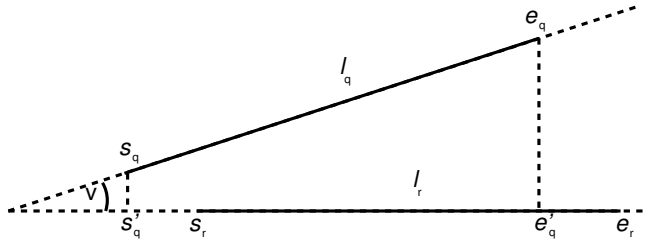


Figure 3.16. Geometric similarity between line l_r with start point s_r and end point e_r and line l_q with start point s_q and end point e_q and their projections, s'_q and e'_q, on line l_r.

similarity: The angle between the lines, the length difference of the lines and the overlap ratio of the line segments when they are projected onto each other.

The angle ν between two lines, l_r and l_q, can be calculated using the following formula:

$$\nu(l_r, l_q) = \arccos\left(\frac{\overrightarrow{s_r e_r} \cdot \overrightarrow{s_q e_q}}{|\overrightarrow{s_r e_r}| \cdot |\overrightarrow{s_q e_q}|}\right) \tag{3.5.2}$$

where s_i is the starting point and e_i the end point of the line l_i. The start and end points of the line are selected in such a way that most of the line pixel gradients point from the left to the right side of the line.

The length difference of the lines in relation to the total length of the shorter line is used as a further criterion:

$$\tau(l_r, l_q) = \frac{||\overrightarrow{s_r e_r}| - |\overrightarrow{s_q e_q}||}{\min(|\overrightarrow{s_r e_r}|, |\overrightarrow{s_q e_q}|)} \tag{3.5.3}$$

The third criterion is the overlap ratio between the two line segments. For this purpose, the start and end points of line segment l_q are projected onto the straight line on which the line segment l_r is located. We define the ratio of the distance on which both line segments are located to the maximum extent on the straight line as an overlap ratio ρ. The overlap ratio ρ is calculated as follows:

$$\rho(l_r, l_q) = \begin{cases} \frac{|\overrightarrow{s'_q e'_q}|}{|\overrightarrow{s_r e_r}|} & \text{if } \left|\overrightarrow{s'_q s_r}\right| + \left|\overrightarrow{s'_q e_r}\right| = |\overrightarrow{s_r e_r}| \text{ and } \left|\overrightarrow{s'_q s_r}\right| + \left|\overrightarrow{e'_q e_r}\right| = |\overrightarrow{e_r e_r}| \\ \frac{|\overrightarrow{s'_q s_r}|}{|\overrightarrow{e'_q e_r}|} & \text{else if } \left|\overrightarrow{s'_q s_r}\right| + \left|\overrightarrow{s'_q e_r}\right| = |\overrightarrow{s_r e_r}| \text{ and } \left|\overrightarrow{e'_q s_r}\right| \leqslant \left|\overrightarrow{e'_q e_r}\right| \\ \frac{|\overrightarrow{s'_q e_r}|}{|\overrightarrow{s_r e'_q}|} & \text{else if } \left|\overrightarrow{s'_q s_r}\right| + \left|\overrightarrow{s'_q e_r}\right| = |\overrightarrow{s_r e_r}| \text{ and } \left|\overrightarrow{e'_q s_r}\right| > \left|\overrightarrow{e'_q e_r}\right| \\ \frac{|\overrightarrow{e'_q s_r}|}{|\overrightarrow{s'_q e_r}|} & \text{else if } \left|\overrightarrow{s'_q s_r}\right| + \left|\overrightarrow{e'_q e_r}\right| = |\overrightarrow{e_r e_r}| \text{ and } \left|\overrightarrow{s'_q s_r}\right| \leqslant \left|\overrightarrow{s'_q e_r}\right| \\ \frac{|\overrightarrow{e_r e'_q}|}{|\overrightarrow{s_r s'_q}|} & \text{else if } \left|\overrightarrow{s'_q s_r}\right| + \left|\overrightarrow{e'_q e_r}\right| = |\overrightarrow{e_r e_r}| \text{ and } \left|\overrightarrow{s'_q s_r}\right| > \left|\overrightarrow{s'_q e_r}\right| \\ \frac{|\overrightarrow{s_r e_r}|}{|\overrightarrow{s'_q e'_q}|} & \text{else if } \left|\overrightarrow{s'_q s_r}\right| \leqslant \left|\overrightarrow{s'_q e_r}\right| \text{ and } \left|\overrightarrow{e'_q s_r}\right| > \left|\overrightarrow{e'_q e_r}\right| \\ \frac{|\overrightarrow{s_r e_r}|}{|\overrightarrow{s'_q e'_q}|} & \text{else if } \left|\overrightarrow{s'_q s_r}\right| > \left|\overrightarrow{s'_q e_r}\right| \text{ and } \left|\overrightarrow{e'_q s_r}\right| \leqslant \left|\overrightarrow{e'_q e_r}\right| \\ 0 & \text{else} \end{cases} \tag{3.5.4}$$

3.5.3 Repetitive Line Detection Method

The appearance similarity can already be used to determine lines that are visually similar. However, the lines do not necessarily represent a repetitive structure. Therefore, we additionally consider geometric criteria in order to identify the repetitive lines and in particular to find the boundaries of the repetition. The local boundaries, i. e., the first and the last line of the repetitive structure, can be useful for matching the repetitive structure across multiple images.

A line l_q is similar to a reference line l_r if the line l_q is located in a local environment of l_r and meets the following criteria:

$$\nu(l_r, l_q) < t_\nu \tag{3.5.5}$$

$$\tau(l_r, l_q) < t_\tau \tag{3.5.6}$$

$$\rho(l_r, l_q) > t_\rho \tag{3.5.7}$$

$$\delta(l_r, l_q) < t_\delta \tag{3.5.8}$$

where t_v, t_τ, t_ρ and t_δ are empirically determined threshold values.

Figure 3.17 shows all lines that are similar to the blue line based on these criteria. Red lines are on the left-hand side of the reference line and green lines are on the right-hand side of the reference line.

Figure 3.17. Detection of similar lines. The red and green marked lines are the lines, which have a high appearance and geometric similarity to the blue line within a local environment.

The detection procedure works as follows. For each detected line, the algorithm checks whether lines exist in their local environment that are similar based on the geometric and photometric criteria. If at least two similar lines are found, the line is marked as part of a repetitive structure. If the current line is not yet part of a repetitive cluster, a new cluster is created that contains the line and similar lines. If the line is already part of a repetitive cluster, the similar lines are added to the cluster. If the similar lines are placed only on one side of the line, the line is marked as the boundary of the repetitive structure.

Figure 3.18 on the facing page shows the repetitive clusters detected in the image. A total of four clusters are found. Two interleaved clusters are detected on each louver. The reason for this is that two lines are detected on each blade of the louver, corresponding to the upper and lower edge of the blade. These lines belong to different clusters because they have different gradient directions and descriptors.

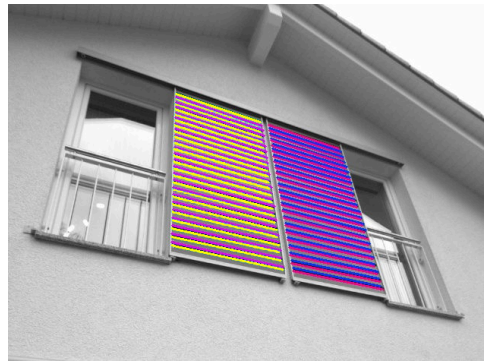


Figure 3.18. Detected repetitive line clusters. All lines of a cluster are displayed with the same color.

3.6 Summary

In this chapter, we have introduced our method for the detection of image features. The method enables robust and subpixel accurate detection of points, lines and arcs, and builds up a graph describing the topological relationships between the detected features. The detection method works directly on distorted perspective and fisheye images and can detect repetitive structures in the images.

In the next chapter, we evaluate our method using several challenging datasets and compare it with the current state-of-the-art methods.

Chapter 4

Evaluation and Results

In this chapter, we show the results of our detection method on several challenging datasets compared to the current state-of-the-art methods. We perform qualitative and quantitative evaluation on natural and synthetic datasets.

4.1 Preliminary Remarks

4.1.1 Test Environment

For the evaluation and the comparison of the methods, we use as a test system a mid-range PC with the following components:

- Intel Core i7-2600K, 4×3.4 GHz
- 8 GB RAM
- 1 TB HDD

4.1.2 Implementation and Parameters

The approach we developed is implemented in C++. Furthermore, the software library OpenCV [Bra00] is used for basic image processing operations.

In Chapter 3, we presented the parameters for each step of the algorithm. Through numerous experiments, we have empirically determined a set of default values for the parameters that provide reliable results over a wide range of different scenes. These are not necessarily the best parameters for all scenes, but the default values were chosen so that suitable results can also be expected on new and unknown scenes. The default

values for the parameters are given in Table 4.1. These fixed default values are always used for the evaluations and comparisons.

Table 4.1. The default values for the parameters of our detection method.

Parameter	Value	Description
L_{min}	15 px	Minimum number of pixels for fitting a primitive
d_{max}	1.2 px	Maximum deviation of edge pixels from the primitive
L_{final}	30 px	Minimum length for final primitives
c_{max}	3.0	Maximum curvature ratio for arcs
t_ν	10°	Maximum angle between repetitive lines
t_τ	2.0	Maximum length difference between repetitive lines
t_ρ	0.75	Minimum overlap ratio between repetitive lines
t_δ	0.6	Maximum descriptor distance between repetitive lines

4.1.3 Baseline Algorithms

Since the approach we proposed allows the simultaneous detection of points, lines, and curves, we use several publicly available image feature detectors for the comparative analysis.

We compare the results of our corner detector with the Harris corner detector [HS88] and the Shi-Tomasi corner detector [ST94]. Both detectors are well known and are still frequently used. They belong to the group of gradient-based corner detectors. We use the implementations from the OpenCV library. In addition, we use the corner refinement from OpenCV to determine the corners with subpixel accuracy.

To evaluate the detection results of the higher-order primitives, i. e., lines and arcs, we compare our results with the results of the widely used LSD (line segment detector) [vGJM+10] and EDLines [AT11] and the recently proposed ELSDc [PGG17]. For the LSD detector we use the implementation from the OpenCV library. For ELSDc we use a publicly available implementation of the authors and for EDLines we use a version we implemented ourselves based on the published paper.

4.1.4 Test Datasets

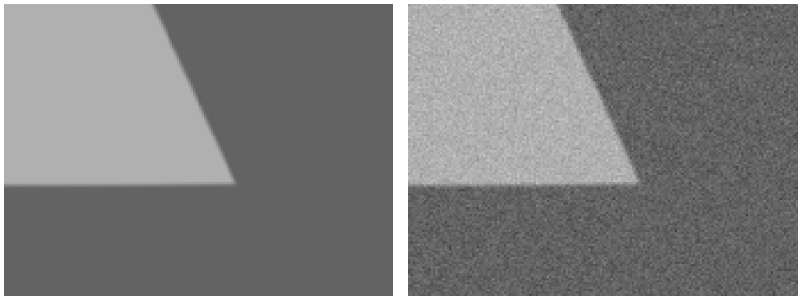
For the evaluation, we use our own and public available synthetic computer-generated and real-world datasets.

Geometric Shapes

The first dataset consists of 50 images with a resolution of 2064×1544 pixels, which contain simple geometric shapes, i. e., triangles and rectangles. The individual forms do not overlap. We use 8-bit images, i. e., the pixel values are between 0 and 255.

The images show a distortion which corresponds to a fisheye lens with an angle of view of 120°. For algorithms that require image without distortion as input, the images are corrected with the inverse distortion in a preprocessing step. These generated software-corrected images are then used as input for the algorithms.

We disturb the images with different levels of additive white Gaussian noise. The standard deviation of the noise is varied from 0 to 15. The dataset thus contains a total of 800 images. Figure 4.1 shows two cutouts of images from the dataset. The left one is without noise and the right one is disturbed by noise.



(a) (b)

Figure 4.1. Detailed view of a corner in an image from the dataset with geometric shapes: (a) Without noise and (b) disturbed by additive white Gaussian noise with a standard deviation of 15.

Synthetic Scenes

For a more realistic evaluation of the detection quality when using the detectors for images of man-made environments, we use computer-generated images of synthetic building scenes.

The images have a resolution of 2064×1544 pixels. Again, we use the model of a fisheye camera with an angle of view of 120°. Figure 4.2 shows some examples of such images.

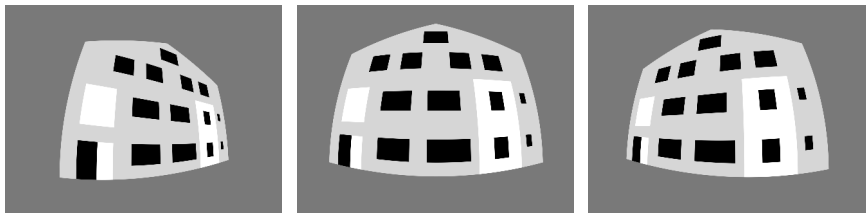


Figure 4.2. Examples of computer-generated images of synthetic building scenes.

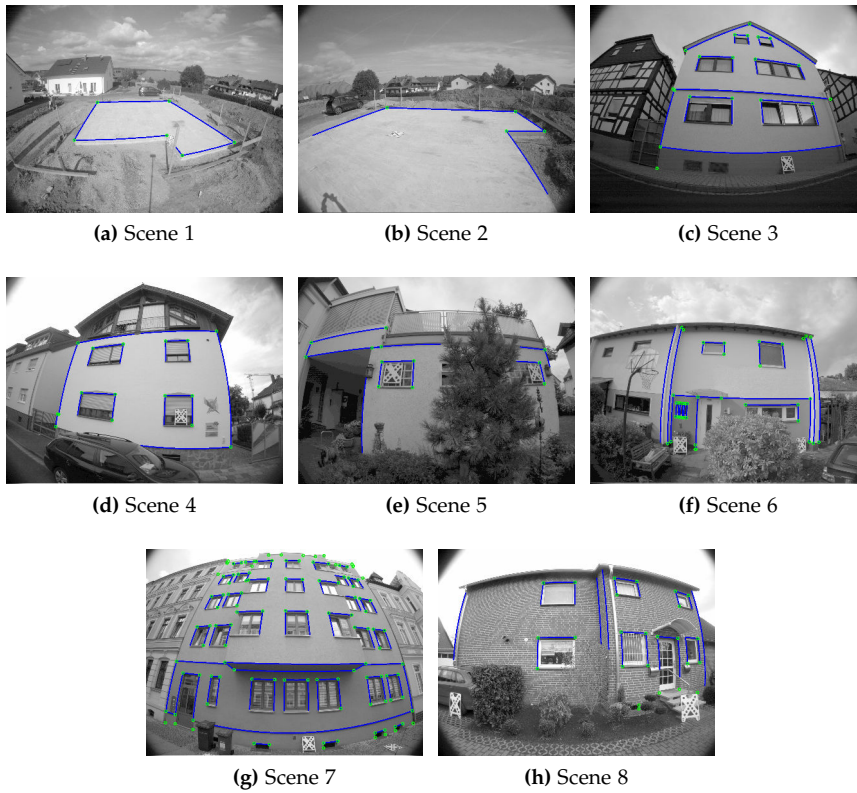
The images are disturbed with different levels of additive white Gaussian noise. In addition, images with different dynamic range are generated.

Real-World Scenes

The first dataset with real-world images consists of eight scenes showing real buildings. Each scene contains between 8 and 32 images, so that the dataset consists of a total of 165 images. The images were taken with a fisheye camera with an angle of view of 120°. The calibration of the camera is known and was also used to generate the synthetic scenes. The images have a resolution of 2064×1544 pixels.

Figure 4.3 on the facing page shows one image from each scene. The images cover several common challenges, such as complex geometric primitives, occlusions and background clutter.

In addition, the scenes contain reference objects of known size. We use special customized boards with dot patterns. The diagonal size of the boards is 60 cm. These boards can be detected in the images and enable



(a) Scene 1 **(b)** Scene 2 **(c)** Scene 3

(d) Scene 4 **(e)** Scene 5 **(f)** Scene 6

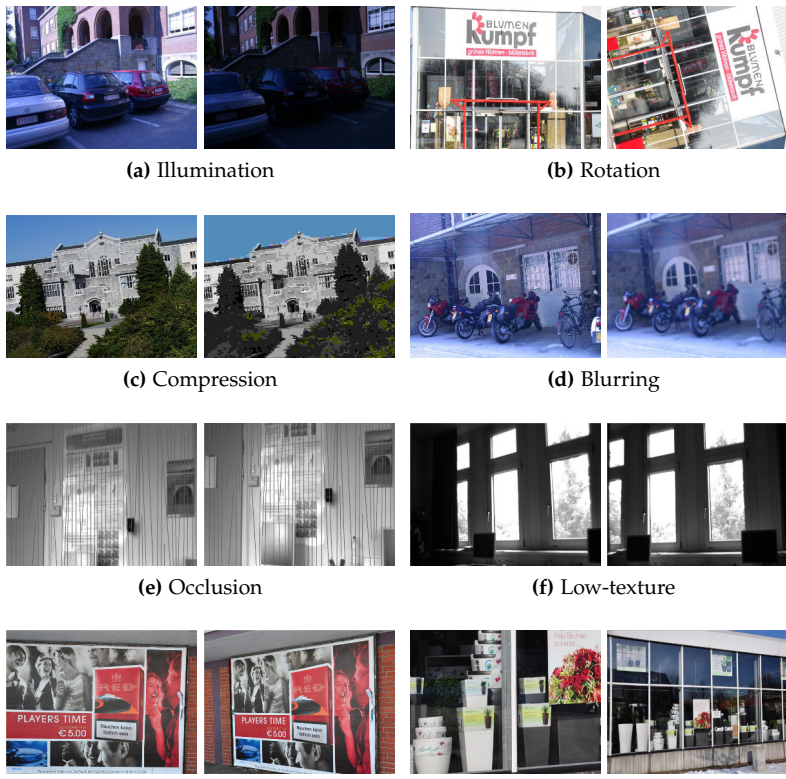
(g) Scene 7 **(h)** Scene 8

Figure 4.3. Example images from the real-world scenes with the relevant geometric structures labeled by humans.

to calculate an initial pose of the cameras and to determine the absolute scaling of a reconstruction of the scene.

In order to provide more than just a qualitative evaluation of the typical behavior of the detectors using examples, the essential geometric structures, e. g., the corners and edges of the buildings and windows in the facade, were labeled by humans in each image. These human labeled points and lines are used as reference for the evaluation of the

detectors. These reference features do not represent a complete ground truth for the detection results, but a selection of significant structures in the images. The features generally correspond to the geometric primitives that are relevant for a reconstruction in man-made environments. They are therefore a suitable measure to evaluate the quality of the detection for this application.



(a) Illumination **(b)** Rotation

(c) Compression **(d)** Blurring

(e) Occlusion **(f)** Low-texture

(g) Viewpoint **(h)** Scale

Figure 4.4. Example images from the dataset with eight groups of image transformations. Images from the groups (a), (c) and (d) are from [MS05] and the others from [ZK13].

Image Transformations

The second dataset with real-world images consists of eight groups of images with different image transformations. These are illumination changes, in-plane rotation, JPEG compression, images blurring, image occlusion, low-texture, viewpoint changes and scale variations. Each group contains six images with increasing image transformations. Figure 4.4 on the preceding page shows two example images from each group. The images from the groups (a), (c) and (d) are from [MS05] and the others from [ZK13].

The images either show planar scenes or have a fixed camera position. The transformation between the images can therefore be described via a homography. The ground truth homographies are known and included in the dataset.

4.1.5 Evaluation Measures

Localization Error

The accuracy of the detection results can be measured by the localization error. The localization error is calculated from the error distance in 2D space.

In the case of a corner, this corresponds to the Euclidean distance d between the detected corner $p_q = (x_q, y_q)^\mathrm{T}$ and the ground truth $p_r = (x_r, y_r)^\mathrm{T}$:

$$d(p_r, p_q) = |p_r - p_q| = \sqrt{(x_r - x_q)^2 + (y_r - y_q)^2} \tag{4.1.1}$$

In the case of a line, we calculate the localization error as the sum of the perpendicular distances $d_\perp$ from the start and end point of the ground truth line $l_r = (s_r, e_r)$ to the detected line $l_q = (s_q, e_q)$.

$$d(l_r, l_q) = d_\perp(l_q, s_r) + d_\perp(l_q, e_r) \tag{4.1.2}$$

$$= \frac{\left|n_q \cdot (s_r - s_q)\right|}{\left|n_q\right|} + \frac{\left|n_q \cdot (e_r - s_q)\right|}{\left|n_q\right|} \tag{4.1.3}$$

where n_q is the normal vector of the line l_q.

Precision and Recall

In order to compare the quality of the detection results of different detectors, precision and recall are used as measures.

The precision is also called positive predictive value (PPV) and is the proportion of positive results in statistics that are true positive results. Recall is also referred to as the true positive rate (TPR) or sensitivity and is defined as the proportion of positives that are correctly identified as such.

The values can be determined based on the number of true positives t_p, false positives f_p and false negatives f_n.

$$\text{PPV} = \frac{t_p}{t_p + f_p} \tag{4.1.4}$$

$$\text{TPR} = \frac{t_p}{t_p + f_n} \tag{4.1.5}$$

In the context of image feature detection, a detected image feature is considered to be true positive if it can be matched with a ground truth feature. For the synthetic datasets, the ground truth features are known. For real-world scenes, human labeled ground truth features are used. False positives are detected features that cannot be matched to the ground truth and false negatives are ground truth features that have not been detected.

In the evaluation, we consider a ground truth corner to be recognized correctly if the detector detects a corner with a maximum distance of 2.5 pixels from the ground truth corner. For line segments, on the one hand, we consider the perpendicular distances from the end points of the ground truth line to the detected line. These must each be below 2.5 pixels. On the other hand, the overlap ratio of the ground truth line and the detected line must be above 0.6.

Repeatability

For a pair of images with known homography, the repeatability of the image features can be determined. For this purpose, the homography is applied to all image features that have been detected in the first image. Then, for each transformed feature, the nearest neighbor is determined from the features detected in the second image. If the distance between the two

features is below a threshold value, they are considered as correspondence. The repeatability R is then defined as the number of correspondences N_{corr} divided by the minimum of the number of detected features, N_1 and N_2, in the two images.

$$R = \frac{N_{\text{corr}}}{\min(N_1, N_2)} \tag{4.1.6}$$

4.2 Corner Detection

In the following section, we compare our corner detector with two classic gradient-based corner detectors: Harris corner detector [HS88] and the Shi-Tomasi corner detector [ST94].

4.2.1 Localization Accuracy

The aim of the first experiment is to determine the accuracy of the intersection features generated by our detector compared to classical corner features.

To test the detection accuracy, the synthetic dataset with images of simple geometric shapes that are disturbed with different levels of Gaussian noise is used. We apply the algorithms to the images and determine the residuals from real to the determined corner points of the objects.

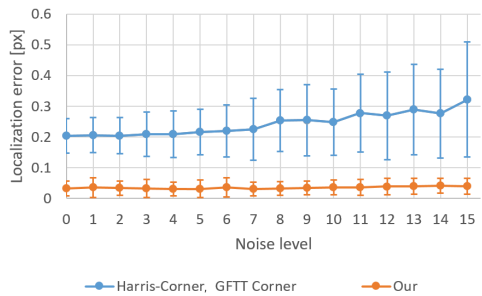


Figure 4.5. Comparison of the localization error. The results of Shi-Tomasi corner detector and Harris corner detector are equal.

The results are shown in Figure 4.5 on the preceding page. The detection accuracy of the Harris corner detector and the Shi-Tomasi corner detector is equal, as the same refinement method is used to determine the corners with subpixel accuracy.

Furthermore, it can be observed that the intersection features have a significantly higher localization accuracy than classical corner features. In particular, it is shown that the intersection features are more robust against the noise and can still be reliably localized even in the case of strong noise. The mean localization error of classical corner detectors increases noticeably with higher noise. In particular, the standard deviation of the localization error indicated by the error bars increases significantly.

4.2.2 Detection Accuracy

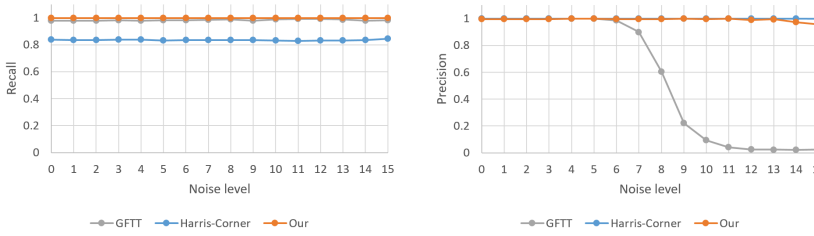
In addition to the localization accuracy, the detection accuracy is of particular importance. Here it is evaluated how reliably corners in images are detected. Synthetic computer-generated images as well as natural images are used for the comparisons.

Synthetic Scenes

In a first evaluation of the detection accuracy, we use the dataset with the synthetic building scene. The images are again disturbed with different levels of Gaussian noise. In the evaluation, precision and recall for the different methods are determined as a function of the noise.

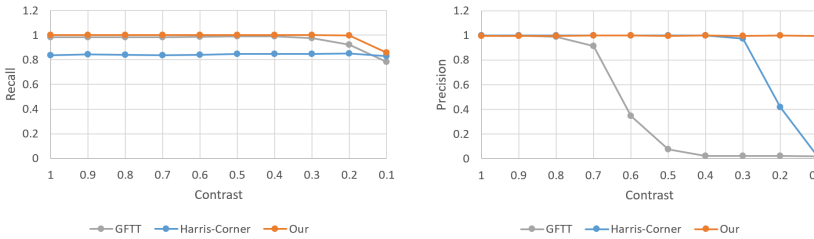
The results are shown in Figure 4.6 on the next page. The values for recall are stable for all methods independent of the noise level. Our approach achieves the highest recall. Almost all corners in the synthetic images are detected correctly. The Shi-Tomasi corner detector achieves only slightly lower results. The Harris corner detector cannot detect a number of corners and only achieves a recall of approx. 0.82.

A comparison of the precision shows different results. For the Harris corner detector and our method, the precision is independent of noise at a very high level. This means that these methods generate almost no false detections even with high noise. The Shi-Tomasi corner detector behaves quite differently. The precision decreases significantly from a noise level



(a) Recall **(b)** Precision

Figure 4.6. Detection accuracy for different levels of Gaussian noise.



(a) Recall **(b)** Precision

Figure 4.7. Detection accuracy for different levels of image contrast.

of 7. This means that many false detections are generated on the images that are disturbed by higher noise.

Figure 4.8 on the following page shows the results for each detector on an image without noise and an image disturbed by noise. The described behavior is easy to comprehend. In the results of the Harris corner detector, some relevant corners are missing and the Shi-Tomasi corner detector produces many false detections on the image, which are disturbed by noise.

In a second experiment, we use the same synthetic building scene with a fixed noise level of 5 and vary the dynamic range of the images. The aim is to assess how the detection quality behaves in areas with low contrast, as this is a typical property of scenes in man-made environments.

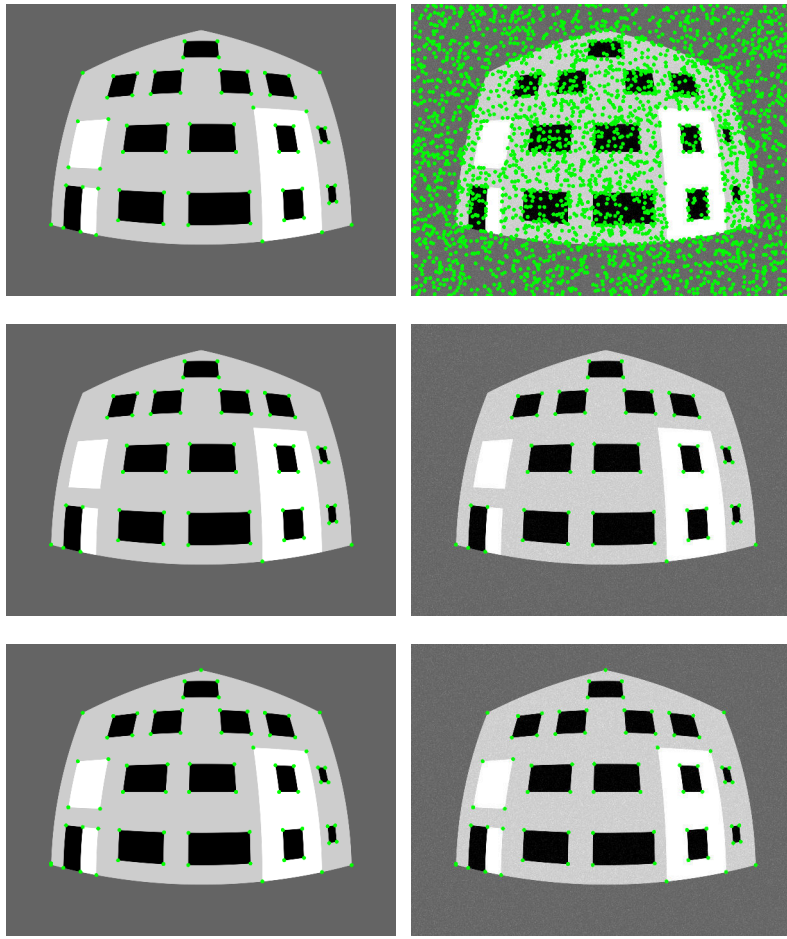


Figure 4.8. Detection results on images from the dataset of a synthetic building scene without noise in the left column and with noise in the right column. In the first row is Shi-Tomasi corner detector, in the second Harris corner detector and in the third our approach.

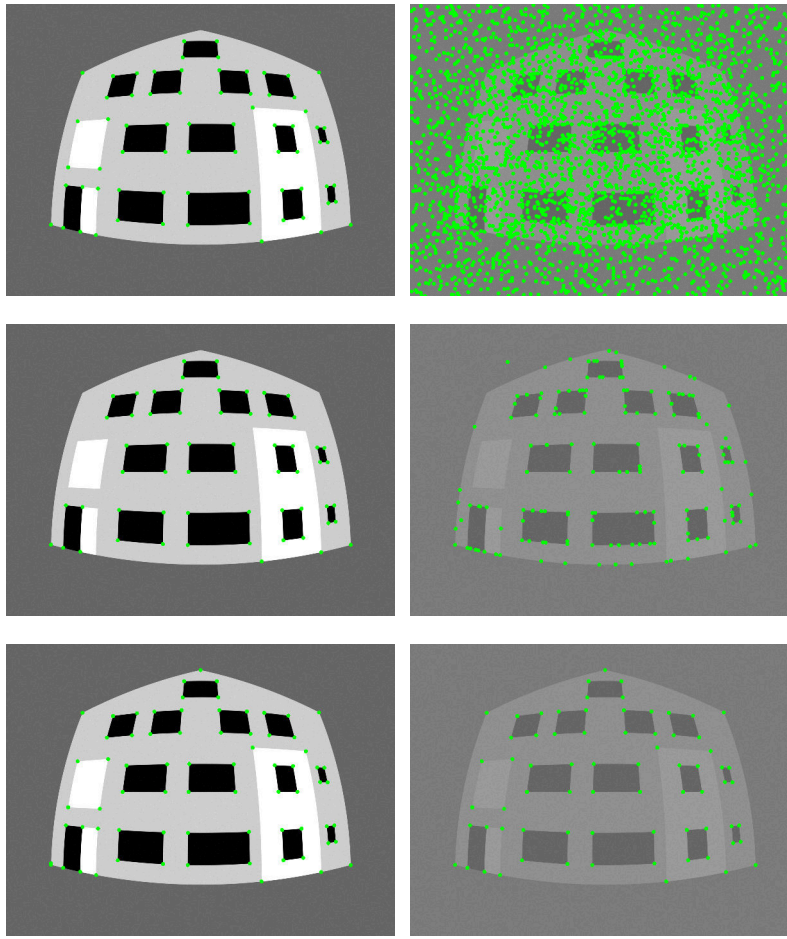


Figure 4.9. Detection results on images from the dataset of a synthetic building scene with high contrast in the left column and with low contrast in the right column. In the first row is Shi-Tomasi corner detector, in the second Harris corner detector and in the third our approach.

The results are shown in Figure 4.7 on page 71. We evaluate precision and recall as before. The Shi-Tomasi corner detector and our approach again achieve a very high recall. The results do not deteriorate until the contrast is very low. An evaluation of the precision shows that both the Shi-Tomasi corner detector and the Harris corner detector produce numerous false detections in images with low contrast. Our approach achieves high precision even at low contrast and thus does not produce false detections in regions with low contrast.

The behavior can be observed in the example images in Figure 4.9 on the preceding page. Here the detection results for the different methods are shown on high contrast and low contrast images.

Real-World Scenes

In addition to the synthetic test data, we use the dataset with the images of real buildings for a further test of detection accuracy. The corners determined by the detectors are compared with the corners labeled by humans.

The results of the evaluation are summarized in Table 4.2. For each detector, the recall achieved and the average number of corners per image determined for the eight real-world scenes (Figure 4.3 on page 65) are

Table 4.2. Detection accuracy on the eight real-world scenes (Figure 4.3 on page 65). The recall and the average number of corners per image for Shi-Tomasi corner detector, Harris corner detector and our approach are given.

	Shi-Tomasi		Harris corner		Our	
Scene	Recall	Avg. corners	Recall	Avg. corners	Recall	Avg. corners
No. 1	0.60	2726	0.10	178	0.90	1481
No. 2	0.50	1804	0.00	106	0.96	1231
No. 3	0.55	398	0.04	63	0.88	1523
No. 4	0.12	282	0.01	48	0.86	1364
No. 5	0.52	1124	0.13	301	0.83	1206
No. 6	0.20	1387	0.02	170	0.75	1367
No. 7	0.47	1421	0.07	178	0.90	2333
No. 8	0.24	1797	0.07	267	0.53	4228

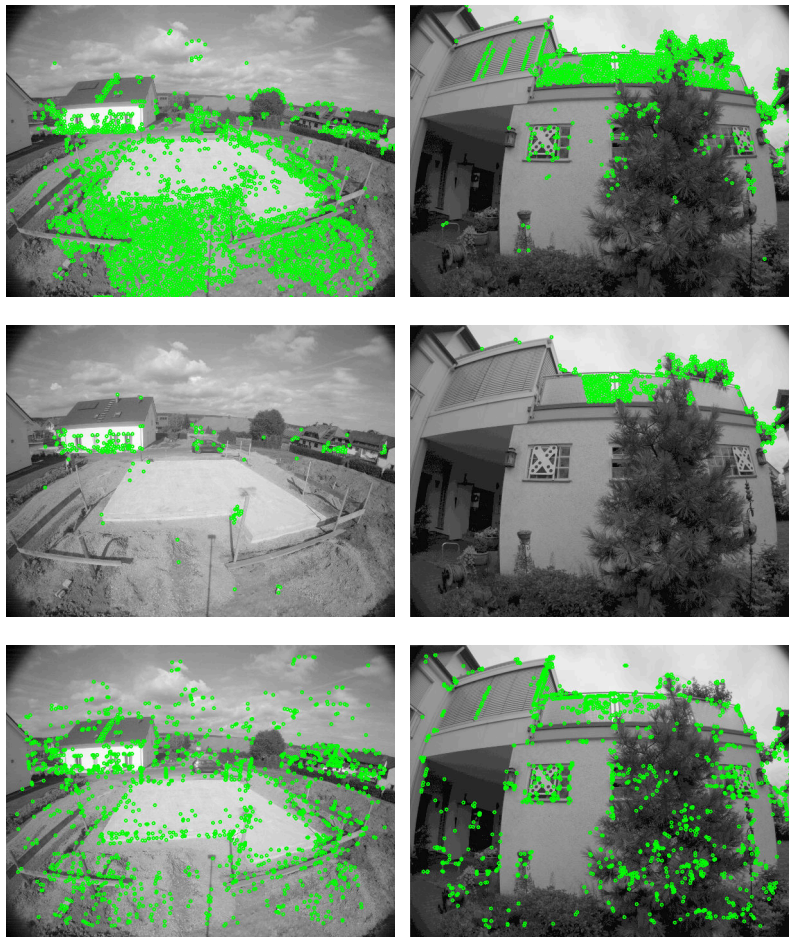


Figure 4.10. Detection results on images from the dataset with real-word building scene. In the first row Shi-Tomasi corner detector, in the second Harris corner detector and in the third our approach.

Figure 4.11. Detection results on images from the dataset with real-word building scene. In the first row Shi-Tomasi corner detector, in the second Harris corner detector and in the third our approach.

given. The precision is not considered for the real-world scenes, since the corners labeled by humans do not correspond to a complete ground truth detection. A meaningful assessment of the false negatives is therefore not possible. Instead, the average number of detected corners per image is given, as a high number of corners can be an indication of false detections.

The Harris corner detector achieves the least recall. Overall, the detector detects significantly fewer corners than the other two. The Harris corner detector is therefore not suitable for reliable detection of the relevant corners in man-made environments. This can also be clearly seen in the example images in Figure 4.10 on page 75 and Figure 4.11 on the facing page. The Shi-Tomasi corner detector and our approach detected significantly more corners. The number is on a similar level. A comparison of the recall shows that our approach achieves a significantly higher rate. It is between 0.8 and 0.9 on most scenes, so nearly all corners labeled by humans were recognized as corners by our approach.

The clearly lower recall of 0.53 in the last scene is noteworthy. This scene shows a brick building. Not all edges required to create the corners from intersections can be detected due to the strong irregularity of the outer edges. Such scenes are challenging for our approach. Nevertheless, the recall achieved is significantly higher than with the classic corner detectors taken into account in the comparison.

4.3 Line Detection

To evaluate the results of the line detection, we compare our results with those of the LSD [vGJM+10], EDLines [AT11] and ELSDc [PGG17], as this are the state-of-the-art methods for line and ellipse detection.

Since our method is the only one that can work directly on distorted images, we have created software-corrected images without distortion for all the input images. Our method is applied once to the distorted original images and for comparison also to the software-corrected version.

4.3.1 Localization Accuracy

In a first experiment, we look again at the synthetic dataset with geometric shapes used in the previous section. This time we measure the localization error of the lines by determining the perpendicular distance between the start and end point of the ground truth line to the lines detected by the different methods.

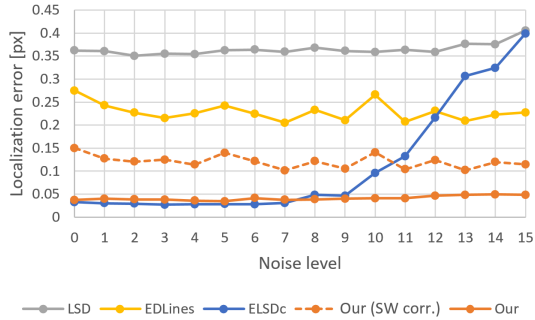


Figure 4.12. Localization accuracy of the detected lines.

Figure 4.12 shows the mean localization error of the individual methods as a function of the noise level. For the measurement of the localization error, the lines detected in the distorted original image are warped into the coordinate system of the software-corrected image to provide a meaningful comparison of the calculated residuals.

ELSDc achieves the highest accuracy of all methods that use the software-corrected images as input. A similar accuracy is achieved by our method when it works directly on the distorted original images. With increasing noise, the localization error of the ELSDc increases significantly and at a noise level of 15 it is similar to the localization error of LSD. The LSD has the largest localization error, which is between 0.35 and 0.4 pixels independent of the noise level. The localization error of EDLine is in the range between 0.2 and 0.3 pixels. The fact that EDLines has a larger localization error than our method on the software-corrected images, although it also uses the Edge Drawing method for edge detection, is due

to the fact that EDLines does not perform a refinement step that calculates positions with subpixel accuracy for the edge pixels.

If our approach is applied directly to the distorted original image, the localization error is below 0.05 pixels even with high noise. The method thus achieves significantly higher accuracy on images with a high noise level than the methods taken into account in the comparison.

4.3.2 Minimum Resolvable Distance

In a further experiment, the ability to recognize fine details is examined. For this purpose, the minimum distance between two parallel lines is determined, in which they can be detected separately. We use generated synthetic images which contain parallel lines with different distances between 1 and 10 pixels. In the evaluation, we determine how often theses lines are detected as two separate lines for each distance. The generated images have no distortion, so in this case there is no need to distinguish between software-corrected and original images.

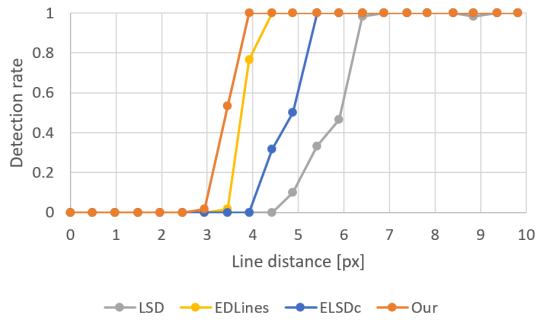


Figure 4.13. Proportion of lines that can be resolved by the detection methods as two separate parallel lines depending on the distance between the lines.

The results (Figure 4.13) show that with our method, two parallel lines can be detected separately with a distance of approximately 4 pixels. For EDLines the minimum distance is 4.5 pixels and for ELSDc it is 5.5 pixels. With the LSD, this is possible not until a distance of about 6.5 pixels. Our approach has the highest resolution and can distinguish finer structures.

4.3.3 Detection Accuracy

For the evaluation of the detection accuracy, we use the dataset with the real-world images. We determine the recall of the different detection methods using the lines labeled by humans.

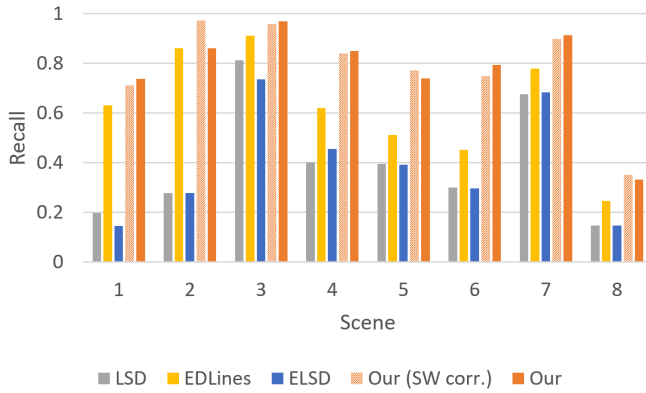


Figure 4.14. Detection accuracy on the eight real-world scenes. The recall for LSD, EDLines and ELSDc and our approach, with and without software correction, is given.

Figure 4.14 shows the recall achieved by the detection methods for the eight real-world scenes (Figure 4.3 on page 65). The results differ clearly across the different scenes. The first two scenes, which show a concrete slab, are a challenge for LSD and ELSDc. The other methods achieve suitable results here. The brick building (scene no. 8) is a challenge for all methods, while the comparatively simple facade (scene no. 3) can be reliably detected by all methods.

Our approach achieves the highest recall on all scenes, both on the distorted original images and when using the software-corrected images. An examination of the number of detected lines per image (Table 4.3) and the average length of the lines (Table 4.4) can give an explanation.

Our method usually detects fewer lines on the images, but these are significantly longer. This is an indication that lines, especially those detected by LSD and ELSD, often fragment into several sub-segments and

Table 4.3. Average number of lines per image detected by LSD, EDLines and ELSDc and our approach, with and without software correction, on the eight real-world scenes.

Scene	LSD	EDLines	ELSDc	Our (SW corr.)	Our
No. 1	1217	1018	1282	371	348
No. 2	640	738	691	290	342
No. 3	495	496	571	607	734
No. 4	342	332	413	506	620
No. 5	559	418	644	371	509
No. 6	929	644	984	416	485
No. 7	1232	1206	1475	800	975
No. 8	3130	2976	3916	1153	1250

Table 4.4. Average length of lines detected by LSD, EDLines and ELSDc and our approach, with and without software correction, on the eight real-world scenes.

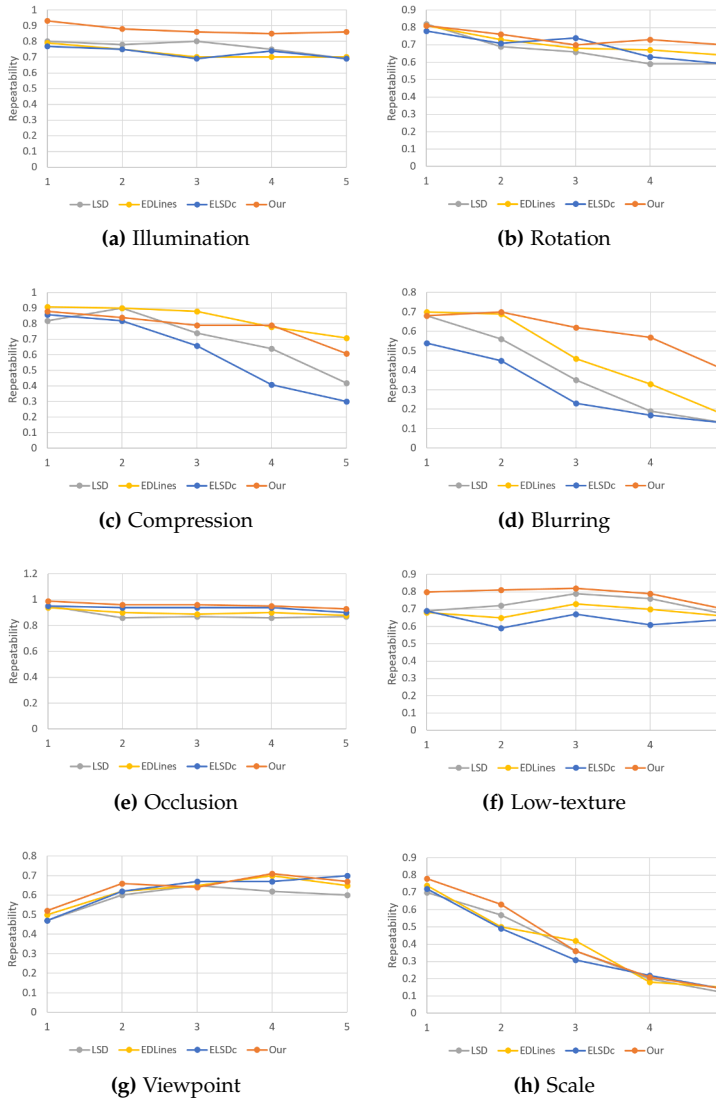
Scene	LSD	EDLines	ELSDc	Our (SW corr.)	Our
No. 1	33	53	32	105	110
No. 2	39	56	39	116	116
No. 3	80	104	76	139	146
No. 4	63	93	65	150	153
No. 5	44	73	42	132	149
No. 6	43	74	43	155	158
No. 7	62	82	60	142	147
No. 8	41	55	39	106	112

are not recognized as a contiguous line segment. Noise interference and background clutter cause the line segments to break off.

4.3.4 Robustness to Image Transformations

In the following section, we evaluate the detection methods with respect to the robustness against image transformations. We use the dataset with eight groups of different transformations and known homographies from [MS05] and [ZK13] (see Figure 4.4 on page 66).

For the evaluation, we determine the repeatability between the image



(a) Illumination **(b)** Rotation

(c) Compression **(d)** Blurring

(e) Occlusion **(f)** Low-texture

(g) Viewpoint **(h)** Scale

Figure 4.15. Results of the repeatability for each group of image transformations. The top 100 lines are used to measure repeatability.

pairs of all groups. Since this measurement method is biased towards detection methods that detect many lines, we employ the measure proposed by [BWG15], which uses only the best k lines. To sort the lines, we use the length as a criterion for all methods. For the evaluation, we consider the longest 100 lines detected by the different methods.

The results for the repeatability for all groups of image transformation are shown in Figure 4.15 on the facing page. For most transformations, such as rotation, occlusion, viewpoint changes and scaling, there are no significant differences between the different detection methods.

In the case of illumination changes, blurring and low textured scenes, it can be observed that our approach outperforms the other methods. Particularly for illumination changes and low textured scenes, this is not surprising, as the results of the other evaluations have also shown that our approach can handle difficult exposure situations better.

4.3.5 Computational Efficiency

Another important measurement for an empirical evaluation of detection methods is the computational efficiency. It is measured by the time costs spent on detection.

Table 4.5 shows the average runtimes per image for the eight real-world scenes. LSD and EDLines have the shortest runtimes. These are significantly lower than those for ELSDc and our approach. In particular,

Table 4.5. Average runtime per image of the detectors LSD, EDLines, ELSDc and our approach on the eight real-world scenes.

Scene	LSD	EDLines	ELSDc	Our (SW corr.)	Our
No. 1	149 ms	155 ms	4080 ms	906 ms	821 ms
No. 2	138 ms	144 ms	3956 ms	883 ms	811 ms
No. 3	138 ms	145 ms	3958 ms	882 ms	811 ms
No. 4	139 ms	146 ms	3919 ms	885 ms	819 ms
No. 5	138 ms	143 ms	3841 ms	886 ms	824 ms
No. 6	138 ms	144 ms	3770 ms	894 ms	831 ms
No. 7	134 ms	141 ms	3606 ms	898 ms	834 ms
No. 8	129 ms	136 ms	3378 ms	897 ms	828 ms

the runtime of ELSDc is more than one order of magnitude higher. In the evaluation it must be taken into account that LSD and EDLines only detect lines, while ELSDc and our approach enables the detection of further primitives. Our method additionally detects parabolic arcs and corners, and ELSDc detects circles and elliptical arcs.

Furthermore, it must be noted that the calculation of the software-corrected input images, which are required for all methods except ours, takes approximately 40 ms per image. These time costs are included in the specified runtimes. For a fair comparison, the same resolution was used for the software-corrected images as for the distorted original input images.

4.4 Arc Detection

Most of the structures that occur in man-made environments can be described by line segments. Architectural elements that occur occasionally and cannot be adequately approximated by line segments are arches. They are generally used to span openings in masonry. They are therefore primarily found on the upper edges of windows and doors.

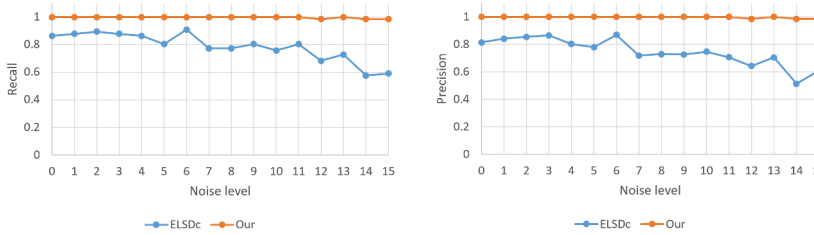
In the following section, we evaluate the ability of our approach to detect and describe arcs. We compare our detector with ELSDc, which is also able to detect arcs.

4.4.1 Detection Accuracy

In a first experiment, we evaluate the detection accuracy using synthetic images showing round and segmental arched windows.

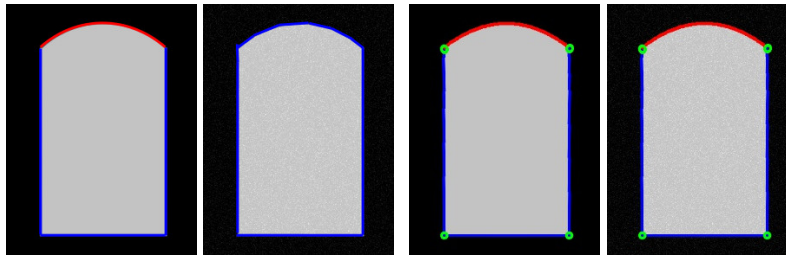
A total of 60 arched windows were generated. The images were disturbed with different levels of Gaussian noise. Depending on the noise level, the number of arches that were correctly detected was determined.

The results are shown in Figure 4.16 on the facing page. Our method achieves a recall and a precision of almost 1. Even with strong noise, the results do not deteriorate. ELSDc only achieves a recall of 0.87. In addition, recall and precision decrease with increasing noise level.



(a) Recall **(b)** Precision

Figure 4.16. Detection accuracy for different levels of Gaussian noise.



(a) ELSDc **(b)** Our

Figure 4.17. Detection results on synthetic images with arched windows. Left image without noise and right images disturbed by Gaussian noise with a standard deviation of 15 (red: arcs, blue: line segments, green: intersections).

Figure 4.17 shows the detection results of ELSDc and our approach for one of the arc windows. Once without noise and once disturbed by strong noise with a standard deviation of 15.

In the image disturbed by noise, ELSDc incorrectly approximates the arc by several short line segments. This behavior can often be observed on the test data and explains the deterioration of recall and precision with increasing noise level. In both cases, our method recognizes the arch of the window as an arc.

4.4.2 Qualitative Results

For a quality evaluation of the arc detection methods in real-world applications, we have tested it on numerous natural images that cover common challenges like complex geometric primitives and background clutter.

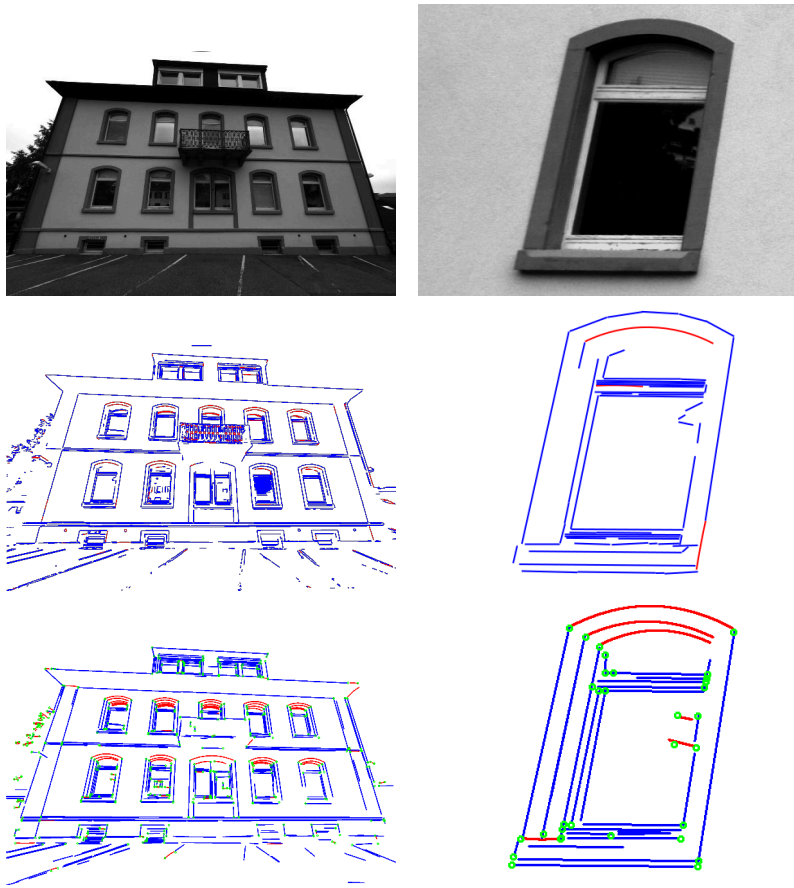


Figure 4.18. Detected image features on a natural image. In the upper row are the input images, in the middle the results of ELSDc and in the bottom row our results (red: arcs, blue: line segments, green: intersections).

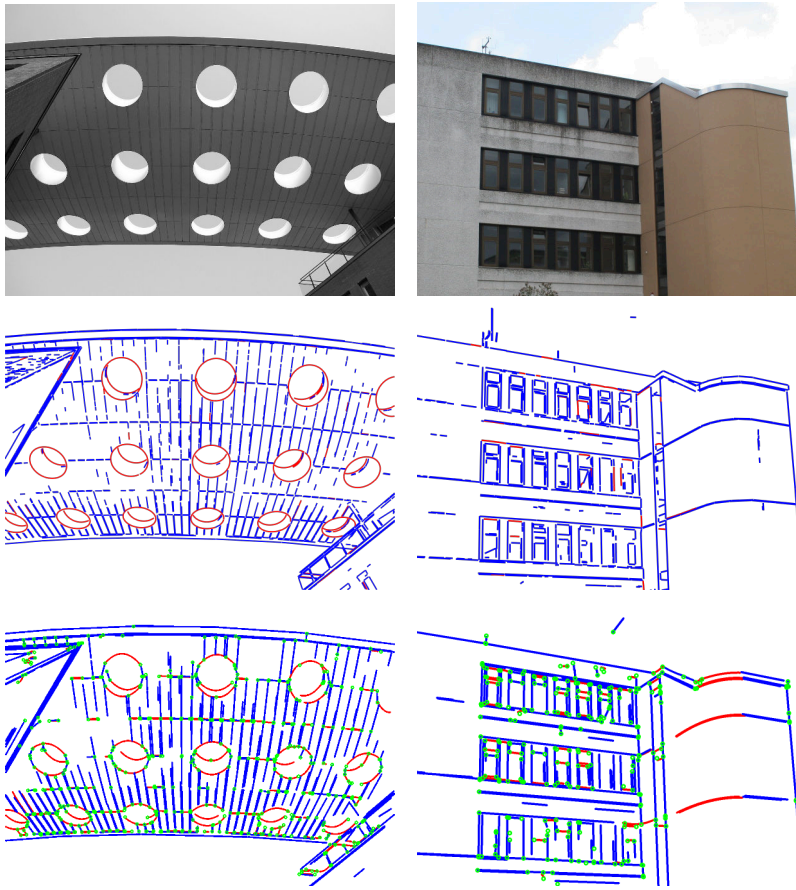


Figure 4.19. Detected image features on natural images. In the upper row are the input images, in the middle the results of ELSDc and in the bottom row our results (red: arcs, blue: line segments, green: intersections). Left image is from the dataset from [PGG17].

For this purpose, we used the datasets from [PGG17] as well as own images. We describe the typical behavior of the detectors by some examples.

Figure 4.18 on page 86 shows the results for detection on a building with arched windows. Both detection algorithms show similar results, but in the detail view of a window, typical differences are recognizable. With ELSDc, the upper edge of the arched window is approximated by several line segments and is not detected as an arc. With our approach, all arches of the windows are correctly recognized.

Furthermore, with our approach finer structures can be detected, for example on the window frame. This confirms the quantitative results that have been shown in the analysis with the synthetic images. In addition, the straight lines break down less often into several sub-segments.

The intersections, which are additionally calculated with our approach, are each quite well located at the physical corners of the objects. At the same time, there are hardly any intersection points in non-relevant areas such as the wall surface or the background clutter.

Further examples of detection results are given in Figure 4.19 on the previous page. Despite the limitation of our detector on parabolas, the circular structures in the left image can be recognized. However, these are approximated by multiple parabolas. The ELSDc correctly recognizes these structures as ellipses.

In the right image, the rounding of the staircase is correctly recognized as an arc with our approach. With ELSDc, the arc is again approximated by several line segments.

4.5 Topology Graph

A novel feature of our approach is that not only image features are detected, but also the topological relationships between them are extracted and stored in a graph.

The aim of the following section is to evaluate whether the extracted topological information is suitable for supporting subsequent steps of a reconstruction process, such as matching the image features. For this, the extracted topological relationships must meet several requirements. On

the one hand, the topology extracted from the images should correspond as closely as possible to the actual topology of the three-dimensional scene. On the other hand, the topological information extracted from different images of the same scene should be consistent.

4.5.1 Synthetic Scene

In a first experiment, we use the computer-generated synthetic test scene. The advantage of this scene is that the topology of the image features is clearly defined. The image features that are extracted at the individual elements of the scene, e. g., windows and doors, should each form a common connected component of the graph. At the same time, there should be no connection in the graph between the image features that are extracted at different scene elements.

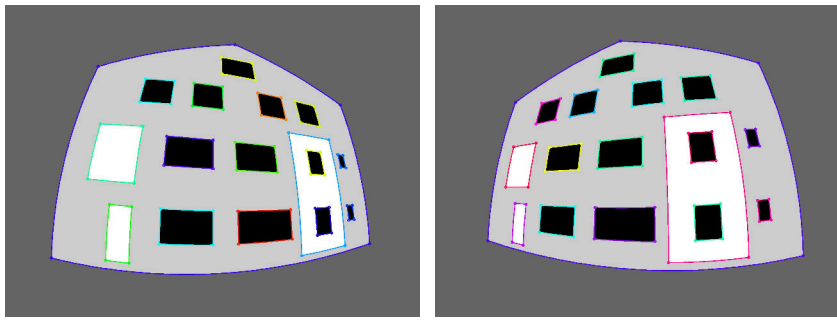
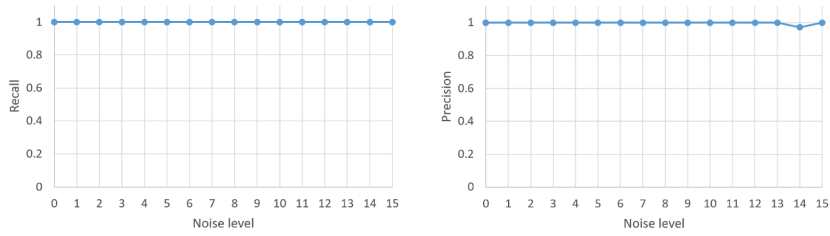


Figure 4.20. Example images of the synthetic scene with the image features detected therein. The image features that form a connected component in the topology graph are displayed in the same color.

Figure 4.20 shows two images of the synthetic scene and the image features extracted from it. All image features that form a connected component in the graph are displayed in the same color. The corners and lines of the individual scene elements are marked with the same color as expected, i. e., the topological relationships of the image features were correctly extracted in the scene.

For a quantitative evaluation, we disturb the images of the synthetic scene with different levels of Gaussian noise and determine the number of scene elements whose topology was correctly represented in the extracted graph.

Figure 4.21 shows recall and precision of the extraction of the topological relationships for the synthetic scene. Both recall and precision are almost constant at 1, so in the case of a simple synthetic scene, the topological relationships are extracted completely correctly even at high noise levels.



(a) Recall **(b)** Precision

Figure 4.21. Accuracy of the extracted topological relationships for different levels of Gaussian noise.

4.5.2 Real-World Scenes

In the following experiments we analyze to what extent the results on the simple synthetic scenes can be transferred to natural images from real-world scenes. The real-world scenes contain various challenges such as more complex geometric primitives and background clutter.

Creating reliable ground truth for the topological relationships of image features in arbitrary natural images is far from trivial. Therefore, we use a different approach for the evaluation than for the synthetic scenes.

Consistency of the Topology between Images

First, we look again at the eight real-world scenes with the human labeled ground truth features (see Figure 4.3 on page 65). In addition, the corresponding features in successive images of the scenes were labeled. Thus, for each human labeled ground truth feature in one image, its position in the other images of the scene is known.

The idea of the first experiment is to determine the consistency of the extracted topology on natural images. We determine the detected features and the topological relationships between them on all images of a scene. For all detected features for which a human labeled ground truth feature exist, we determine whether a path exists between them in the topology graph, that is, whether they belong to the same connected component of the graph. Then we look at two successive images of the scene and determine how many of the feature pairs that are connected in the first image are also connected in the second image.

The results of the experiment are given in Table 4.6. The evaluation is made separately for corners and line segments. For most scenes, the proportion of topologically consistent feature pairs in successive images is between 70% and 90%. The extracted topological relationships are therefore largely consistent across several images of a scene. The topology

Table 4.6. Consistency of the extracted topology between successive images of real-world scenes for corners and line segments. The connected pairs and the consistent matched pairs as well as their ratio are given.

	Corners			Line segments		
Scene	# pairs	# matched	Prop.	# pairs	# matched	Prop.
No. 1	72	39	0.54	44	32	0.73
No. 2	25	22	0.88	12	9	0.75
No. 3	129	106	0.82	187	165	0.88
No. 4	195	164	0.84	200	162	0.81
No. 5	100	81	0.81	28	25	0.89
No. 6	380	283	0.74	92	68	0.74
No. 7	1821	1198	0.66	956	724	0.76
No. 8	52	15	0.29	26	13	0.50

graph can thus provide valuable information to support matching and correspondence search between images.

The results of scene no. 8, which shows a brick building, are clearly worse. However, the reduced detection rate (see Table 4.2 on page 74) must also be taken into account. This explains why the topological relationships of the features from different images of the scene deviate from each other.

Histogram of the Size of the Connected Components

Figure 4.22 shows a histogram of the size of the connected components of the topology graphs for all images of the eight real-world scenes. Connected components with a size of three elements have the largest proportion. These have a frequency of about 30%.

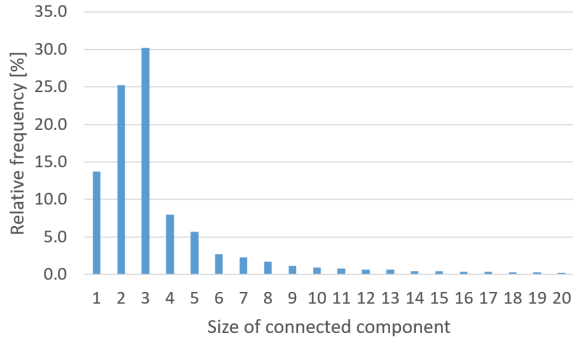


Figure 4.22. Histogram of the size of the connected components of the topology graphs for all images of the eight real-world scenes.

Connected components with a size between one and five elements together represent a proportion of over 80%. This means that the individual connected components represent only a small local structure of the scene. If image features are in a connected component across multiple images of a scene, this is therefore a strong indication that the image features also represent a coherent structure in the three-dimensional scene.

Qualitative Results

At the end of the evaluation of the topology graphs, we give some qualitative results using example images representing typical results of the extracted topological relationships.

Figure 4.23. Detected image features in sample images from the real-world scenes (no. 3, no. 4 and no. 7). The features that are part of a connected component of the topology graph are displayed in the same color.

Figure 4.23 on the preceding page shows for three of the real-world scenes the detected features that are part of a connected component of the topology graph.

Two images from each scene are shown. The features of a connected component have the same color.

The extracted topologies often map the correct relationships of the objects in the scene. Each of the individual windows are part of a connected component of the graph. It is also clearly visible that the topology of the two different images of the same scene is very similar.

4.6 Recognition of Repetitive Lines

The evaluation of the detection of repetitive lines is carried out qualitatively using sample images that show typical results.

The images (Figure 4.24 on the next page) show repetitive structures that often occur in man-made and urban scenes. The left column shows the input images and the right column the detected line segments. Line segments that are colored magenta are recognized as part of a repetitive structure.

In the first image, the lines detected on the closed roller shutter are reliably detected as a repetitive structure. This also applies to the roller shutters of the second house. In addition, the lines detected on the roof of the neighboring house are recognized as repetitive structures. In the third image, the vertical lines on the two garage doors are correctly detected as a repetitive structure and in the last image, the roller shutters are detected again. In addition, the vertical struts of a gate are detected here.

Such repetitive structures represent a great challenge for subsequent processing steps. This is especially the case for line-based wide-baseline reconstruction. Repetitive structures do not fulfill the assumption that they are unique in their local environment and often lead to matching errors and thus to incorrect reconstruction results.

The method we propose enables reliable detection of repetitive structures and can thus prevent such problems in subsequent processing steps. The repetitive structures are each recognized as a grouped cluster and can therefore be handled separately in subsequent processing.

Figure 4.24. Recognition of repetitive lines. Left is the input image and right the detected line segments. The lines colored in magenta are recognized as part of a repetitive structure.

4.7 Summary

In this chapter, we have evaluated our proposed method in detail by means of numerous experiments with synthetic and real-world data and compared it with the current state-of-the-art methods.

We could show that our approach achieves a high localization accuracy comparable to the state-of-the-art methods and at the same time is more robust against disturbances caused by noise. In addition, our approach allows extracting more fine details in the images.

The detection accuracy achieved on the real-world scenes is constantly above that achieved by the other methods. Furthermore, our process can reliably distinguish between line and arc segments.

The additional topological information extracted by our method is largely consistent over several images of a scene and can therefore be a support for subsequent processing steps, such as matching and correspondence search.

Chapter 5

Application: 3D Reconstruction of Mid-level Primitives

In this chapter, we demonstrate the integration of our proposed approach for the detection of points, lines and arcs in a complete image-based reconstruction process.

In the first part, we combine the image features detected by our approach with current state-of-the-art methods for feature-based 3D scene reconstruction. We present the achieved reconstruction results using several real-world scenes.

In the second part, we present a novel method that uses the topological relationships between the image features extracted by our detection method to reconstruct connected three-dimensional structures and to extract rectangular shapes in the scene.

5.1 Feature-based 3D Reconstruction

In the following section, we show how our detection approach can be used for feature-based 3D reconstruction. To demonstrate this, we integrate our detection approach into a classic point-based structure from motion process. We then extend this by 3D reconstruction of the detected line segments.

5.1.1 Point-based 3D Reconstruction

We presented the common structure of a point-based structure from motion pipeline in Section 2.2.2. The pipeline usually consists of five steps: feature extraction, feature matching, geometric verification, reconstruction

and bundle adjustment. The structure from motion pipeline we use is structured as follows:

SfM Pipeline

1. **Feature Extraction:** For the detection of the image features, we use the combined detection method for points, lines and arcs developed by us.

   A prerequisite for the subsequent matching process is that the image features have a descriptor that describes the local appearance of the features and enables recognition in other images. We therefore extract descriptors associated with all detected points.

   We use FREAK [AOV12] as descriptor. This is a binary descriptor inspired by the human visual system. The aim of the development was to create a descriptor that is both faster and more robust than SIFT [Low04], SURF [BET+08] or BRISK [LCS11].

2. **Point Matching:** The next step is matching. Here visually similar pairs of features are determined in two different images.

   For matching the detected features, we use the Brute-Force Matcher from the OpenCV library. For the descriptors of the features from the first image, the Hamming distance to all descriptors of the features in the second image are determined. The pairs with the smallest distances are the matches.

   Additionally we use a cross check of the matches. This means that only those matches (i, j) are considered for which the i-th descriptor in set A has the j-th descriptor in set B as the best match and vice versa.

3. **Geometric Verification:** In the verification step, the relative pose between the matched camera pairs is determined. The determined feature correspondence and the RANSAC method are used here. Correspondences are then checked against epipolar constraints and outliers are removed.

4. **Reconstruction:** Starting from an initial camera pair, a 3D model is incrementally built up. All matched image features are triangulated so that a sparse three-dimensional point cloud is generated.

5. **Bundle Adjustment:** The aim of bundle adjustment is to determine 3D point positions and camera parameters that minimize the reprojection error. This is a non-linear least squares problem, where the error is the squared Euclidean norm of the distance between the detected feature and the projection of the associated 3D point onto the image plane of the camera.

   For the final bundle adjustment, we use Ceres [AM+]. This is an open source solver for solving large and complex optimization problems. Ceres offers extensive support to solve bundle adjustment problems.

5.1.2 Line-based 3D Reconstruction

In this section, we describe how the reconstruction process can be extended so that the detected lines can also be reconstructed. We assume that the camera poses are already known. In our case, the camera poses are known through point-based reconstruction. Therefore, a pose estimation and bundle adjustment are not necessary when reconstructing the lines. The following steps allow an additional reconstitution of the lines.

Reconstruction Pipeline

1. **Feature Extraction:** The combined method detects the lines at the same time as the points. To characterize the local appearance of line segments we use the Line Band Descriptor (LBD) [ZK13]. The descriptor uses band-like regions parallel to the line segment in which the gradients are accumulated.

2. **Line Matching** For line matching, we use the method presented by Zhang et al. [ZK13] accordingly. This line matching algorithm utilizes both the local appearance of line segments and their geometric attributes.

3. **Reconstruction:** Starting with an initial camera pair, the matched line segments are triangulated. Iteratively, observations from further cameras are added. The triangulated 3D lines are then verified by their reprojection error. Only lines with reprojection errors below a fixed

threshold are used in the final 3D line model. We calculate the reprojection error of a 3D line as the sum of the perpendicular distances $d_\perp$ from the start and end point of the detected 2D observation $l_r = (s_r, e_r)$ in the image to the projected 3D line $l_p = (s_p, e_p)$.

$$d(l_r, l_p) = d_\perp(l_p, s_r) + d_\perp(l_p, e_r) \tag{5.1.1}$$

$$= \frac{\left|n_p \cdot (s_r - s_p)\right|}{\left|n_p\right|} + \frac{\left|n_p \cdot (e_r - s_p)\right|}{\left|n_p\right|} \tag{5.1.2}$$

where n_p is the normal vector of the line l_p. We use a fixed threshold of 3.5 pixels for the reprojection error in the evaluation.

5.1.3 Reconstruction Results

Figure 5.1 and Figure 5.2 show for six real-world scenes one of the input images used for reconstruction and the results of the three-dimensional reconstruction of the points and lines. The reconstructed 3D lines are shown in blue and the reconstructed 3D points in red. The scenes have between 11 and 32 images with a resolution of 2064×1544 pixels.

The scenes show various common challenges that occur on buildings. The size of the buildings ranges from smaller single-family houses to multi-storey buildings in the urban area. Typical difficulties such as occlusions, e. g., by vegetation or balconies, different depth levels, e. g., by oriels, as well as repetitive structures are present in the scenes.

Especially the reconstructed lines enable a clear understanding of the structure of the scenes. The essential structures such as windows, doors and outer edges of the buildings are reconstructed.

Single false triangulation can be seen in the marginal areas, which are only contained in a few images. These are mainly due to errors in matching and the low detection accuracy in the marginal areas of the images. Since the lines are only detected in a few cameras with a small baseline to each other, they are not identified as outliers by the reprojection error.

The reconstructed points are mostly located at the corners of the relevant structures such as windows and doors and thus support the understanding of the scene. The points determined by intersecting line

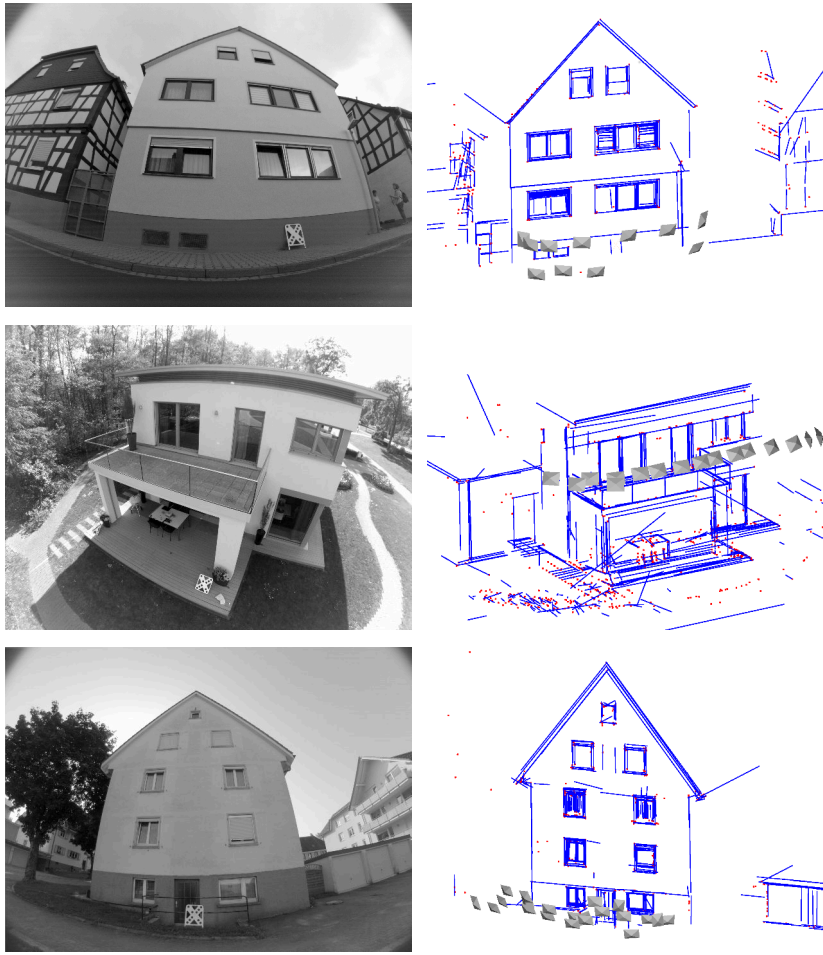


Figure 5.1. Results of the three-dimensional reconstruction on several real-world scenes. The left column shows one of the input images of the scene used for the reconstruction and the right column shows an image of the reconstruction. Reconstructed lines are displayed in blue and reconstructed points in red.

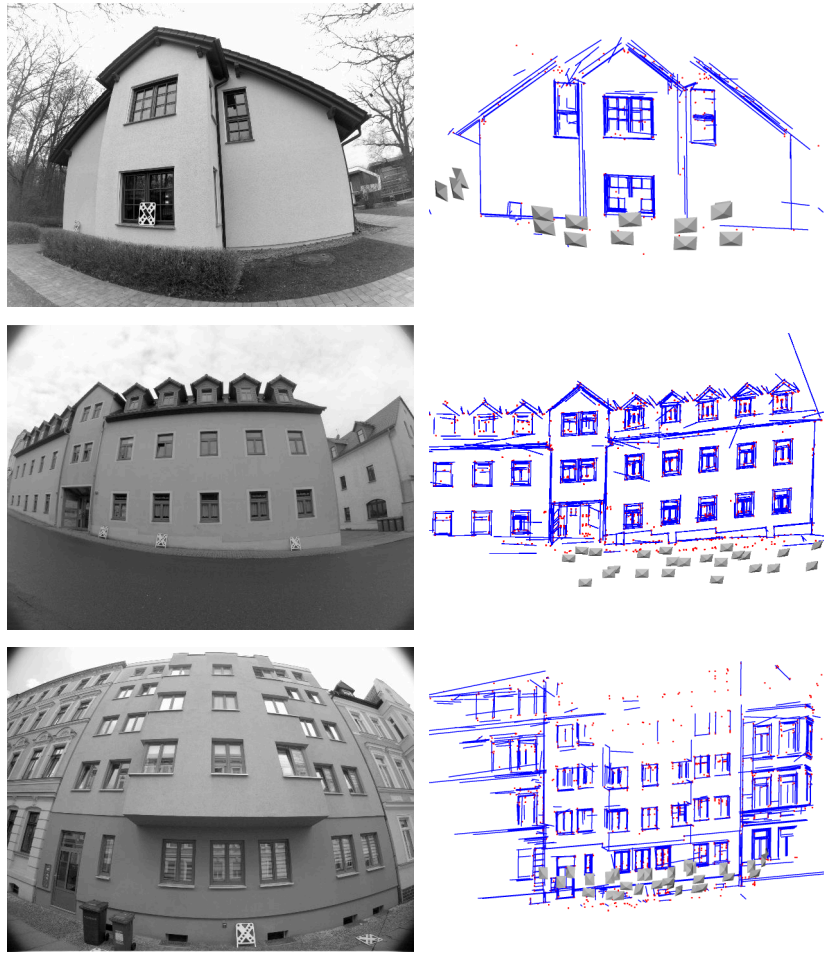


Figure 5.2. Results of the three-dimensional reconstruction on several real-world scenes. The left column shows one of the input images of the scene used for the reconstruction and the right column shows an image of the reconstruction. Reconstructed lines are displayed in blue and reconstructed points in red.

segments are therefore suitable image features for the reconstruction. They can be matched and triangulated reliably over multiple images.

The real-world scenes contain additional reference objects of known size. If these are detected in the images, a scaling factor can be automatically calculated to determine the absolute scaling of the scene with correct dimensions. In this way, the reconstructed scenes can be used for distance measurements. In addition, the reference objects are used to support the initial pose estimation.

Table 5.1 shows the total runtime as well as the average runtime per image for the feature-based reconstruction of points and lines. The runtime depends mainly on the number of images in the scene and the number of features that are detected in the images. The runtime per image is between 7.5 s and 17.9 s. The reconstruction of a complete scene is thus completed after a few minutes.

Table 5.1. Evaluation of the runtime of the feature-based 3D reconstruction using six real-world scenes.

Scene	Runtime [s]	# images	Runtime per image [s]
No. 1	82.1	11	7.5
No. 2	239.6	17	14.1
No. 3	259.5	21	12.4
No. 4	215.6	14	15.4
No. 5	430.2	24	17.9
No. 6	522.7	32	16.3

5.2 Topology-based Reconstruction of Geometric Structures

By reconstructing the line segments, the structure of the scene is easier to understand. However, the reconstruction of each line and point is independent of each other, even if they represent contiguous structures of objects in the scene. In particular, line segments that describe a contiguous planar object in the scene are generally not in the same plane, i. e., they are

not coplanar, in the reconstructed model. Instead, they are slightly shifted in depth relative to each other. The individual lines are therefore skew to each other, which makes it impossible to directly determine contiguous 3D structures.

We therefore propose a method that uses the extracted topological relationships to reconstruct contiguous structures that are on a common plane.

For the extraction of the main planes, we use the Manhattan world assumption [CY99], which states that the scene contains three orthogonal, dominant directions. The architecture of buildings generally corresponds to this assumption to a large extent. The Manhattan world assumption implies that the walls are aligned along two directions orthogonal to each other. In addition, they are orthogonal ground plane.

5.2.1 Detection of the Main Planes

The first step is the detection of the main planes of the scene. We use an approach based on a method that we have presented for the automatic indoor reconstruction based on dense point clouds (see [Wol14]).

Determining the Manhattan World Directions

First the vertical or the gravity direction is determined. This can either be given by an inertial navigation system (INS) or estimated from the vanishing points. In our case, the corresponding gravity vector is given by an INS for each image. The gravity direction corresponds to the normal of the ground plane.

Using the camera poses known from the reconstruction and the associated gravity directions of each image, we calculate an average gravity direction. We use this direction as normal for the ground plane.

To determine the normals of the facade planes, we use the assumption that they are perpendicular to the gravity direction. In addition, we use the assumption that the facade planes are orthogonal to each other. Both facade normals can therefore be determined by only one rotation around the gravity direction.

We use an entropy-based method to determine the rotation. The basic idea is to analyze the distribution of point coordinates in the point cloud. The analysis of the entropy of histograms is a common approach in literature to determine the main directions of a point cloud. Gallup et al. [GFM+07] use the entropy of the histograms along the coordinate axes of a two-dimensional point cloud to optimize alignment. Similar approaches are also used by [OV11] and [NMT13].

For this, the points of the point cloud are projected orthogonally along the gravity direction to determine 2D points. The point cloud determined in this way is now rotated in discrete steps between 0° and 90°. For each rotation, two histograms, H_x and H_y, are calculated based on the x and y coordinates of the 2D points.

The entropy E of a discrete random variable X with possible values $x_1, ..., x_m$ and the corresponding probabilities $p(x_1), ..., p(x_m)$ is defined as:

$$E = -\sum_{i=1}^{m} p_i \log_2 p_i, \tag{5.2.1}$$

In the case of $p(x_i) = 0$ for some i, the value of the corresponding summand is assumed to be 0, which corresponds to the limit:

$$\lim_{p \to 0+} p \log(p) = 0 \tag{5.2.2}$$

The entropy of a histogram is greater if the frequency distribution of

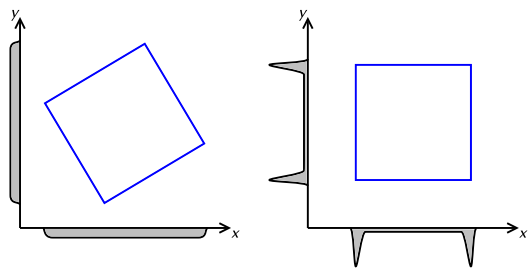
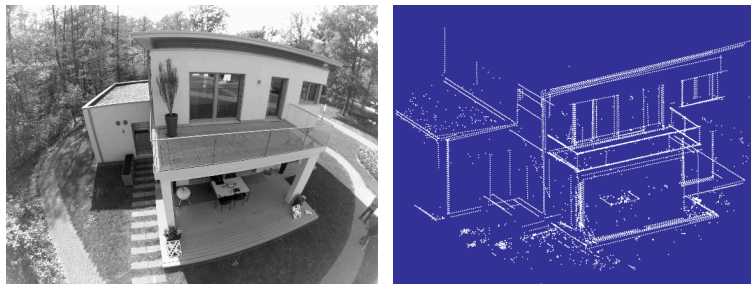


Figure 5.3. Concept of entropy minimization. Two different rotations of the same point cloud (blue) with histograms in x and y direction. The histograms of the right drawing have clearly a low entropy.



(a) Input image **(b)** Point cloud

Figure 5.4. Point cloud generated from the feature-based reconstruction of a building. (a) One of the input images and (b) generated point cloud.

the histogram is closer to a uniform distribution. An ideal alignment to the Manhattan world directions of the point cloud results in a narrow peak in the histogram for each wall section. The entropy of the histogram is minimal in this case (see Figure 5.3 on the previous page).

The rotation α of the point cloud can be determined by Equation 5.2.3. Here $E(H_x(X_\alpha))$ and $E(H_y(X_\alpha))$ are the entropies of the histograms of the x and y coordinates of the point cloud with a rotation of α.

$$\operatorname{argmin} \left(E(H_x(X_\alpha)) + E(H_y(X_\alpha)) \right) \tag{5.2.3}$$

To illustrate the computation, we consider the determination of the Manhattan world directions using the example of a real-world scene. Figure 5.4 (a) shows one of the input images and Figure 5.4 (b) shows the point cloud generated from the reconstructed points and lines. The reconstructed lines are sampled by points to simplify processing.

The point cloud is rotated in discrete steps around the gravity direction. In each step, histograms are generated from the distribution of the 2D coordinates of the points projected onto the ground plane and the entropy of the histograms is calculated. Figure 5.5 on the facing page shows the calculated entropy as a function of the rotation angle for the real-world scene. A clear minimum at 38° can be seen here. This is the rotation α.

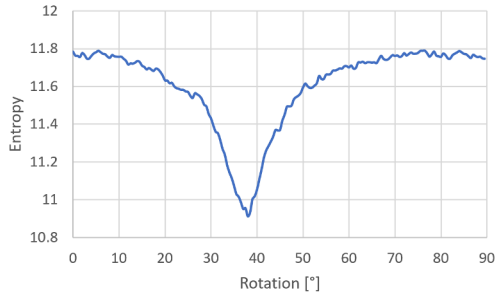


Figure 5.5. Entropy of the histograms of the point coordinates of the real-world scene as a function of the rotation around the gravity direction.

Plane-Sweep

In point clouds, planes can be detected by a plane sweep [Col96; BB10]. This is achieved by moving a plane through the three-dimensional space at discrete intervals along a given direction. The direction is chosen so that it corresponds to the structures to be recognized. In our case, the Manhattan world directions are used as search directions for the plane sweep.

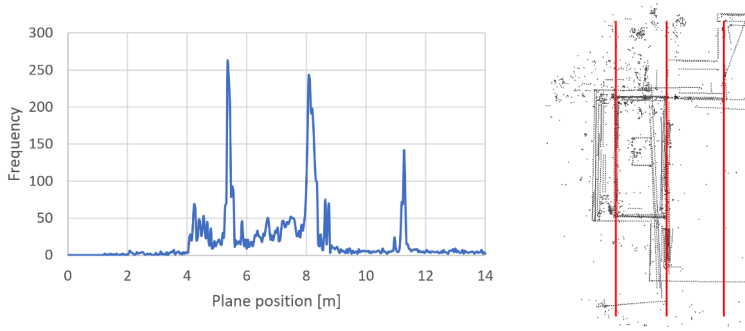
For each step, the number of points is determined that lie within a predefined range around the plane. This creates a histogram of the sweep area. For each sweep step, a class is created in the histogram with the number of points as value.

The peaks in the histogram correspond to areas with a high point density and represent areas of potential planes. The peaks are determined using non-maxima suppression. In this way, the histogram classes are determined whose values are really larger than all values of the classes in a local environment. In addition, a threshold value is used to suppress local maxima in areas with few points.

The Hesse normal form is used to describe the plane:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \cdot \vec{n} - d = 0 \tag{5.2.4}$$

Here $\vec{n}$ is the normal of the plane, d the distance of the plane from the



(a) Histogram **(b)** 2D point cloud

Figure 5.6. Plane sweep through a point cloud along a vertical Manhattan world direction. (a) Histogram of the plane sweep and (b) point cloud projected onto the ground plane with the detected main planes (red).

origin of the coordinate system. In the plane sweep, the plane is moved through the three-dimensional space by gradually changing d.

The histogram of a plane sweep in one of the vertical Manhattan world directions through the point cloud of the real-world scene is shown in Figure 5.6 (a). The plane is moved in discrete distances of 5 cm and the number of points is determined with a distance of less than 2.5 cm from this plane. The histogram shows three clear maxima. An initial plane is generated for each maximum of the histogram with a frequency above a minimum threshold value. We use a fixed minimum threshold of 100 for the frequency.

Figure 5.6 (b) shows a projection of the point cloud onto the ground plane and the vertical planes determined from the peaks in the histogram. These correspond to the main structures of the scene.

Finally, the minimum bounding rectangle (MBR) is determined on the plane. This is the smallest possible rectangle parallel to the axis that encloses all points of the point cloud that lie on the plane or have a distance of less than 2.5 cm to the plane.

Figure 5.7 on the next page shows the main planes of the real-world scene determined in this way. The planes represent a suitable approxi-

mation of the scene and represent the main surfaces of the building. The same way, planes for the second vertical Manhattan world direction are determined.

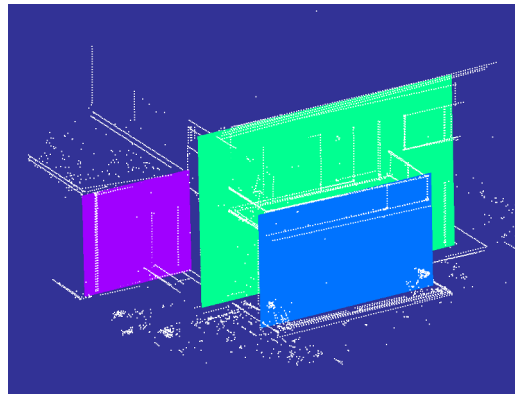


Figure 5.7. Three-dimensional point cloud with the detected main planes for one of the vertical Manhattan world directions.

5.2.2 Topology-based 3D Reconstruction

The main planes are the basis for the topology-based reconstruction procedure we propose. The method uses the detected three-dimensional planes and the extracted topological relationships between the image features to reconstruct connected three-dimensional structures.

In order to reconstruct 3D features and establish the topological relationships between the 3D features, correspondence between the detected 2D features must be determined from all images.

The aim of the procedure proposed by us is the reconstruction of planar objects. Therefore, we use a special matching procedure that works directly on the detected main planes to determine correspondence and none of the known matching procedures (see Section 2.3).

Overview of the Reconstruction Process

The reconstruction process is performed successively for all detected main planes and consists of three steps:

1. Pairwise matching
2. Generation of feature hypotheses
3. Establishment and verification of the topology

The complete algorithm is summarized in Listing 5.1. The individual steps are described in detail in the following sections.

Listing 5.1. Pseudocode for topology-based 3D reconstruction.

```
Input:  P_1,..,P_m - detected planes
        F_1,..,F_n - detected features of images 1,..,n
        C_1,..,C_n - camera poses of images 1,..,n

Output: M - reconstructed 3D model
        G - 3D topology graph

for each P in {P_1,..,P_m}

  // pairwise matching
  for i = 1 to n-1
    backproject features F_i and F_(i+1) to plane P
    for each feature in F_i find most similar in F_(i+1)
    for each feature in F_(i+1) find most similar in F_i
    cross check
  end

  // generate feature hypotheses
  build feature trails based on the pairwise matches
  for each trail
    if trail has at least 4 observations
      create 3D feature hypothesis
    end
  end

  // establishment and verification topology
  for each 3D feature pair
    if at least 3 associated 2D observations are connected
      connect 3D features in the topology graph
    end
  end
  remove unconnected 3D feature hypotheses
end
return 3D model M and topology graph G
```

Pairwise Matching

In the matching step, corresponding image features are determined between successive images of the image sequence. We assume that the image sequence is sorted, i. e., that successive images are visual neighbors and share a large common field of view.

If this assumption is not fulfilled, the cameras that are visual neighbors must be determined in a preprocessing step. An easy way to find visual neighbors is to determine the Euclidean distance between the camera centers and the angle between their optical axes. Alternatively, the results of the SfM reconstruction can also be used to determine visual neighbors based on the correspondences [HMB14].

The matching process now proceeds as follows. First, all image features that have been detected by our detection method are backprojected to the current 3D plane. The matching process takes into account all image features that are within the minimum bounding rectangle of the plane. For a camera pair, for each image feature from the first image, the feature from the second image that is most similar to it is determined.

To determine the similarity of point features, we consider the Euclidean distance between the features backprojected to the planes. In addition, two thresholds are used to define the maximum Euclidean distance and the maximum descriptor distance that is allowed to occur between two features that are considered correspondence. The features backprojected to the plane are used for the distance calculation, as in this way distortion of the images and perspective transformations do not have to be taken into account. Since in our case the correct scaling of the scene is known, the distances can be specified in metric units. The maximum descriptor distance is used because additional checking of the appearance can reduce misfits caused by occlusions and features that are not actually on the plane.

For the lines, the sum of the perpendicular distances from the start and end point of the line feature from the first image to the line feature from the second image is used to determine the similarity (see Equation 5.1.1). Threshold values for the maximum distance of two lines features and for the maximum descriptor distance are also used when determining line matches. Additionally we also use a cross check of the matches here. This

means that only those matches (i, j) are considered for which the i-th feature from the first image has the j-th feature from the second images as the best match and vice versa.

Generation of Feature Hypotheses

The aim of the next step is to determine hypotheses for 3D features that are supported by multiple image observations.

Starting from a pair of images, supporting observations from other images are searched on the basis of the matching results. In this way, trails of corresponding image features are created across multiple images.

For all features supported by at least four image observations, a 3D feature hypothesis is determined. The position of the feature on the 3D plane is determined by fitting it to the image features backprojected onto the plane.

Establishment and Verification of the Topology

In the third step, the topological relationships between the 3D features are established. At the same time, this step is used to verify the feature hypotheses.

To establish the topological relationships between the 3D feature hypotheses, it is examined for each feature pair whether a topological relationship exists between at least three of the associated 2D observations. If this is the case, the nodes associated with the 3D feature hypotheses are connected in the corresponding topology graph.

All feature hypotheses that are not part of a connected component of the graph with at least three nodes are removed after this process. These are usually caused by mismatches. Furthermore, the aim of the method is to identify more complex planar structures.

Reconstruction Results

To evaluate the procedure we use the six real-world scenes, which were also used for the feature-based reconstruction (see Section 5.1). The parameters of the topology-based reconstruction and the values used are summarized in Table 5.2 on the facing page.

Table 5.2. The default values for the parameters of our topology-based reconstruction method.

Parameter	Value	Description
h_{sweep}	5 cm	Bin width of the histogram of point distribution
t_{sweep}	100	Minimum threshold for the frequency of an initial plane
d_{MBR}	2.5 cm	Maximum distance of points for MBR determination
$t_{p,\text{dist}}$	3 cm	Maximum Euclidean distance for point matching
$t_{p,\delta}$	100	Maximum descriptor distance for point matching
$t_{l,\text{dist}}$	3 cm	Maximum Euclidean distance for line matching
$t_{l,\delta}$	0.6	Maximum descriptor distance for line matching
n_{obs}	4	Number of observations for a feature hypothesis
n_{connect}	3	Number of supporters for connecting 3D features

Figure 5.8 on the next page and Figure 5.9 on page 115 show the results of the topology-based reconstruction and one of the input images of each scene. The reconstruction was performed for all vertical main planes detected in the scene. Then the partial reconstructions are combined to one three-dimensional model. All image features associated to a connected component of the topology graph are drawn in the same color.

It is clearly visible that the related structures, such as the windows, are also present in the reconstructed 3D model as a connected structure. The topological relationships of the features can therefore also be preserved by the method during the reconstruction. The contiguous planar objects of a scene are thus also connected in the reconstructed model and on a common plane.

The advantages of the topology-based reconstruction approach become particularly clear in the first scene from Figure 5.8 on the next page. The facades of the neighboring half-timbered houses are reliably reconstructed. The panels between the timbers are each identified and reconstructed as a connected planar structure. In classic feature-based reconstruction (see Figure 5.1 on page 101), only a small part of the area is reconstructed and there is no information about the connection to each other.

The runtimes for detection of the main planes and topology-based reconstruction are given in Table 5.3 on page 116. For topology-based

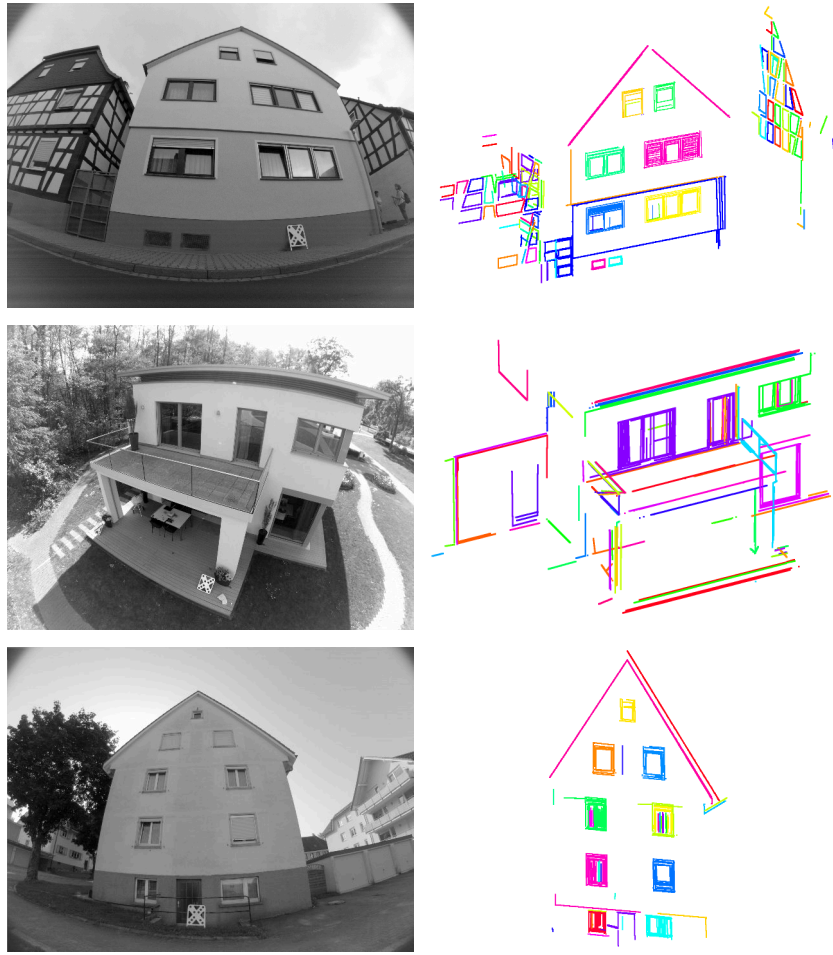


Figure 5.8. Results of topology-based reconstruction. The left column shows one of the input images for each scene and the right column the reconstruction result. Features that belong to the same connected components of the topology graph are drawn in the same color.

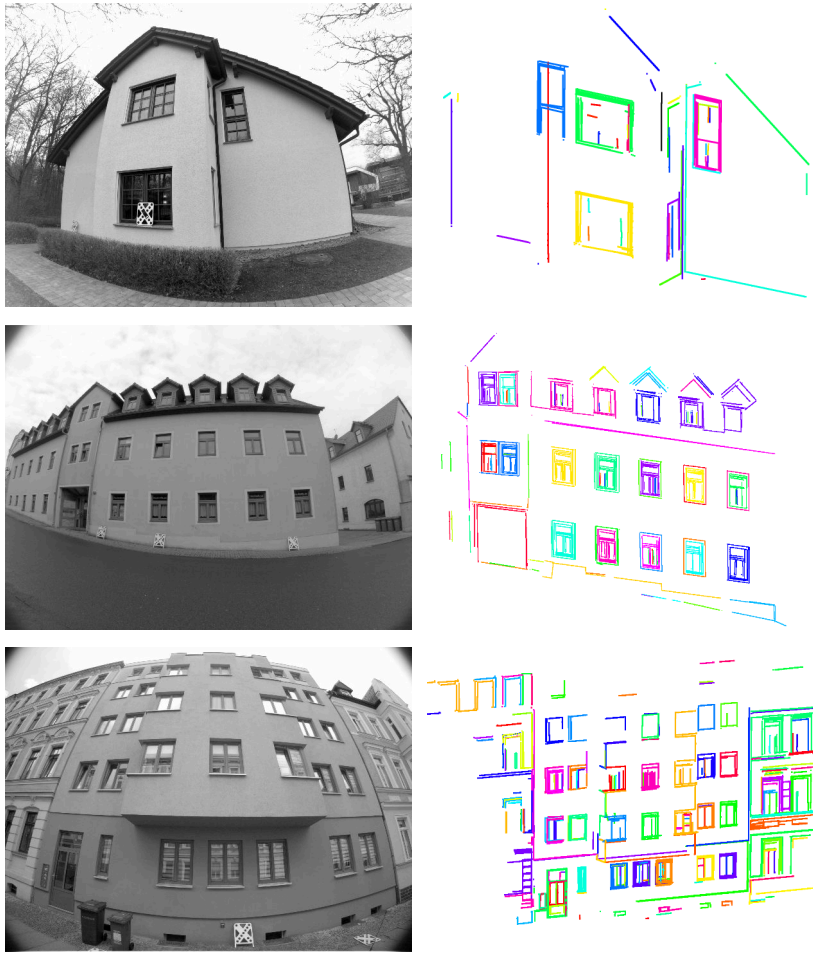


Figure 5.9. Results of topology-based reconstruction. The left column shows one of the input images for each scene and the right column the reconstruction result. Features that belong to the same connected components of the topology graph are drawn in the same color.

Table 5.3. Evaluation of the runtime of the topology-based 3D reconstruction using six real-world scenes.

	Runtime [s]		
Scene	Plane detection	Reconstruction	Avg. per image [s]
No. 1	4.4	141.0	12.8
No. 2	7.3	239.7	14.1
No. 3	2.0	86.1	4.1
No. 4	3.1	155.8	11.1
No. 5	3.5	249.6	10.4
No. 6	4.1	400.2	12.5

reconstruction, the average value per image is also specified.

The detection of the main planes in the point cloud of the SfM reconstruction takes only a few seconds. The proposed method thus enables an efficient detection of the main structures of a scene based on a sparse SfM reconstruction. The runtime depends only on the size of the point cloud and the bin width of the histograms.

The subsequent topology-based reconstruction method has a runtime similar to feature-based reconstruction. The average running time per image is between 4.1 s and 14.1 s. The runtime depends mainly on the number of planes and the number of image features found on them. Therefore, scene no. 3 has the shortest runtime, since it is composed of only one plane and has a simple structure with few image features. For scenes with several planes, e. g., no. 2, or a lot of image features, e. g., no. 6, the runtime is longer.

The overall runtime of the proposed topology-based reconstruction method is thus at a level that allows practical application in reconstruction frameworks.

5.2.3 Topology-based Detection of the Main Rectangles

In the following section, we present a method that uses the topological relationships of three-dimensional reconstruction to extract specific planar structures. We use the topological information to automatically detect rectangles. In man-made environments, this method is a simple way to

detect candidates for windows in facades, as these can usually be assumed to be rectangular.

Detection Process

In addition to the three-dimensional reconstruction of the scene, the topology-based reconstruction also provides a graph describing the topological relationships of the reconstructed features. This graph can be used to search for specific structures in the scene.

For the detection of rectangles, we use the assumption that a rectangle in the reconstructed 3D model consists of four line segments and four corner points that are part of a common connected component of the topology graph. First, simple cycles are searched in the graph.

In graph theory, a simple cycle of a graph $G = (V, E)$ is a path $(v_1, \ldots, v_n)$ with $v_i \in V$ for $i = 1, \ldots, n$ to which applies:

$$v_1 = v_n \tag{5.2.5}$$

$$v_i \neq v_j \text{ for } i, j \in 1, \ldots, n-1 \text{ and } i \neq j \tag{5.2.6}$$

The length of a simple cycle $(v_1, \ldots, v_n)$ is defined as $n - 1$.

For each simple cycle of length eight of the topology graph, it is checked whether it consists of an alternating sequence of point nodes and line nodes. If this is the case, it is validated whether the associated reconstructed features fulfill the properties of a rectangle, i.e. that all line segments connected by a point node have an angle of 90° to each other and that the point node corresponds to the intersection of the two line segments. In this way, planar rectangular structures can be identified in the scene.

Detection Results

For evaluation, we apply the presented method to the results of topology-based reconstruction.

Figure 5.10 on the following page shows the detected rectangles for each scene. For better visualization, the reconstructed rectangles are projected into one of the input images. This enables an easier assessment of

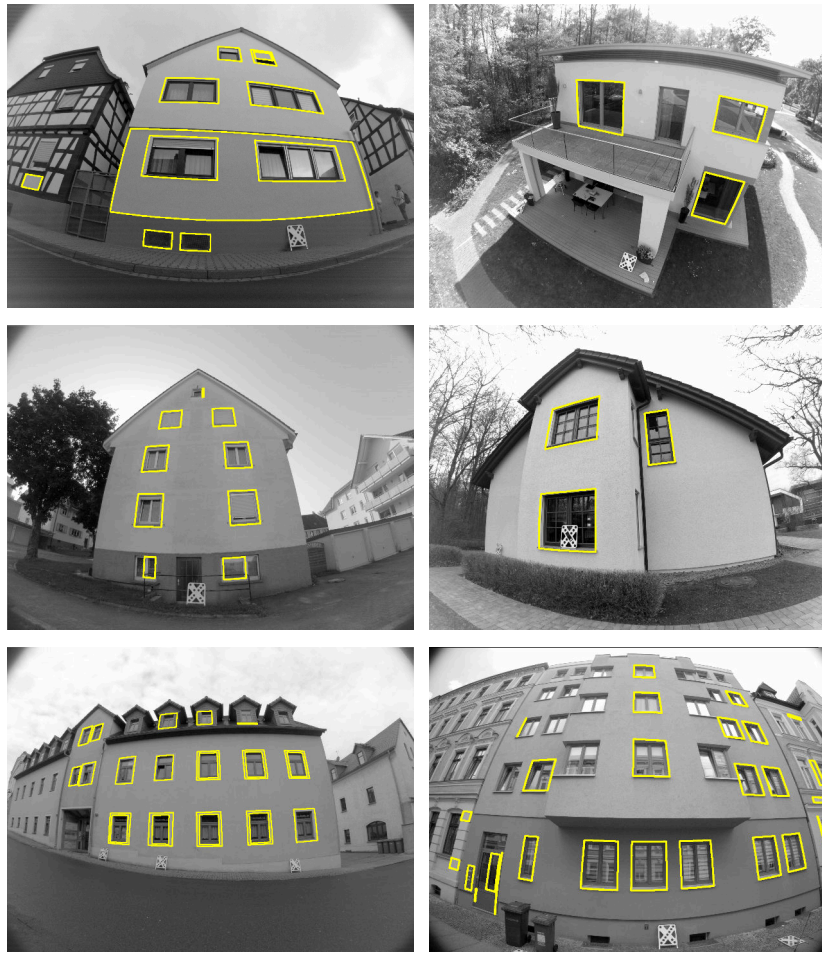


Figure 5.10. Results of automatic detection of rectangles based on topology-based reconstruction. For better visualization, the rectangles are projected into one of the input images.

Table 5.4. Evaluation of the runtime of automatic detection of rectangles based on topology-based reconstruction using six real-world scenes.

Scene	Runtime [s]
No. 1	1.98
No. 2	1.10
No. 3	1.79
No. 4	1.45
No. 5	3.36
No. 6	3.17

the quality of the detection. Only the rectangle with the longest perimeter is displayed for each connected component of the topology graph.

It is clearly visible that most rectangular structures in the scenes are already extracted by this relatively simple approach. In addition, the assumption that the rectangular structures in man-made environments often correspond to the windows is confirmed by the results.

The detection of rectangles in topology-based reconstruction is very efficient because the information about the connections between the characteristics is available in the topology graph, so that the desired structures can be determined with a simple search algorithm. The runtime for the detection of the rectangles in the six real-world scenes is between 1.45 s and 3.36 s (Table 5.4).

The presented method can make a meaningful contribution to the understanding and analysis of scenes. The level of semantic information in the reconstructed scenes can be significantly increased by the additional topological information. The topology-based reconstruction thus enables a highly abstract but at the same time semantically rich 3D model of a scene.

In contrast to the conventional methods used for facade interpretation (see Section 2.5 on page 24), no rectified images of the facade are required. Furthermore, no architectural constraints are enforced, such as the windows being arranged in regular grids. The fact that the structures are reconstructed from several images using detected line segments ensures high localization accuracy and accurate borders of the objects.

5.3 Summary

In this chapter, we have shown how the combined detection method we have developed can be used for feature-based 3D reconstruction. The reconstruction quality is analyzed and evaluated on the basis of several real-world scenes.

Furthermore, we presented a novel reconstruction method that uses the topological relationship of the features to create a highly abstract but semantically rich 3D model of the reconstructed scenes, in which certain geometric structures can easily be detected.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we propose a method that enables robust and accurate detection of geometric primitives in man-made environments.

The detection of geometric primitives is a fundamental step in computer vision techniques such as image processing, image analysis and pattern recognition. Reliable and accurate detection considerably simplifies subsequent processing steps.

Previous approaches usually focus on only one image feature and do not exploit the relationships between the definitions of the features. For example, a corner is defined as the intersection of two connected straight edges. Most methods for detecting corners or junctions operate locally and search for strong curvature in the gradient domain. These approaches produce many misdetections and often detect corners that are less relevant. Our approach uses the intersections of detected line and arc segments to determine corners. This enables robust detection of corners in scenes of architecture and man-made objects. Since the point features generated in this way match the corners of the real objects, they are very useful for further steps and applications such as image analysis and 3D scene reconstruction.

Besides the detection of geometric primitives, our method also extracts the topological relationships between the individual geometric primitives. In this way, more complex geometric primitives can be described over several individual elements. The geometric objects are represented in the topology graph as a connected component of the graph.

Our method enables direct detection of geometric primitives in distorted perspective and fisheye images without the need to correct dis-

tortions by warping with inverse distortion. This can increase both the accuracy and the performance of the detection.

Unlike most common edge detection methods, we do not use fixed threshold values for extracting the edges. Our method considers the noise behavior of the camera and can therefore extract even faint structures in dark areas without causing more misdetections.

In addition, our method provides the recognition of repetitive structures and thus ensures the unambiguity of the detected features. This is an important requirement for reliable matching of features in a wide-baseline reconstruction process.

On the basis of numerous evaluations with synthetic and real-world data we can show that our approach achieves a high localization accuracy comparable to the state-of-the-art methods and at the same time is more robust against disturbances caused by noise. In addition, our approach allows extracting more fine details in the images.

The detection accuracy achieved on the real-world scenes is constantly above that achieved by the other methods. Furthermore, our process can reliably distinguish between line and arc segments.

The additional topological information extracted by our method is largely consistent over several images of a scene and can therefore be a support for subsequent processing steps, such as matching and correspondence search.

Finally, we show how the detection method can be integrated into a complete feature-based 3D reconstruction pipeline and present a novel reconstruction method that uses the topological relationship of the features to create a highly abstract but semantically rich 3D model of the reconstructed scenes, in which certain geometric structures can easily be detected.

6.2 Future Work

Although the results obtained are promising, the method still has opportunities for improvement and expansion. In the following section, we discuss ideas for future research topics.

A first aspect is an extension that enables the detection of additional geometric primitives. Currently points, line segments and arcs in the form of parabolas are available.

In our numerous experiments in urban indoor and outdoor environments, we have found that these primitives are sufficient to handle the majority of cases. Nevertheless, it can be useful to think about extracting further primitives. These could be ellipses or general curves, e. g., in the form of splines. As a result, the area of application of the method could be extended so that reliable detection of natural non-urban environments is also achieved.

The proposed method is well suited for such an extension. By extracting any edges in the form of pixel chains in the first step, it is easy to fit and evaluate models for further primitives in the subsequent fitting step. The correct model can be selected based on the fitting error of the different models.

Another aspect we want to address is the relevance of the detected geometric primitives. Strong geometric structures that are not relevant are a challenge for our detection method. An example for this are the facades of brick buildings, for which the lowest detection rates were observed in the evaluation. The main problems are that not all relevant image features are detected due to the strong irregularities of the outer edges and at the same time, many non-relevant image features are detected on the individual bricks. Figure 6.1 (a) shows the detected image features on such a scene.

One possible solution is machine learning. In an experiment, we developed a filter that identifies non-relevant areas based on the results of edge detection with a learning-based detector [DZ15]. These areas are then skipped by our edge detector. This makes it possible to combine the high localization accuracy of our method with the learning-based detection of the relevant edges.

Figure 6.1 (b) shows the result of the experiment. It is clearly visible that most of the non-relevant structures on the facade of the brick building are suppressed and at the same time, the relevant geometric primitives in the region of the windows are still detected.

The combination with learning-based approaches provides an interesting and promising way to increase the relevance and meaningfulness of

(a) Without filtering **(b)** Learning-based filtering

Figure 6.1. Comparison of detection results without additional filtering and with learning-based filtering.

the detected features and can be an aspect of future research.

We also believe there is great potential for further research on the topological relationships of the different image features and possible applications.

In the application chapter, we could show that the topological relationships can be used to increase the semantic meaning of reconstructed 3D models. However, the possible applications are far from exhausted. An interesting research topic could be the direct integration into a SfM pipeline.

In particular, the search for image feature correspondences between images could benefit considerably. The evaluations have shown that the topology of the image features is largely consistent across several images. Therefore, this information can be used in a matching process.

For example, the consistency of the neighboring image features can be integrated into the matching process as an additional evaluation criterion. Especially the complimentary information of the different image features offers promising opportunities. In this way, the matching of the lines can be supported by the neighboring points and point matching can be supported by the lines in low-textured man-made environments.

At the same time, the intersection points of the lines can also be used to support the triangulation process of the lines. Especially when reliable

line-based triangulation is not possible due to the triangulation angles and camera poses.

As a conclusion, we see the main tasks of future research in this area in extracting the relevant image features and considering the topological relationships more strongly in order to generate highly abstract but semantically rich 3D models that are very useful for many applications.

Bibliography

[AAG+14] M. Alemán-Flores, L. Alvarez, L. Gomez, and D. Santana-Cedrés. "Line detection in images showing significant lens distortion and application to distortion correction". In: *Pattern Recognition Letters* 36 (Jan. 2014), pp. 261–271.

[ABD12] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. "KAZE features". In: *Computer Vision - ECCV 2012*. European Conference on Computer Vision. Springer, 2012, pp. 214–227.

[ALF12] M. Awrangjeb, G. Lu, and C. S. Fraser. "Performance Comparisons of Contour-Based Corner Detectors". In: *IEEE Transactions on Image Processing* 21.9 (Sept. 2012), pp. 4167–4179.

[AM+] S. Agarwal, K. Mierle, et al. *Ceres Solver*. URL: http://ceres-solver.org.

[AOV12] A. Alahi, R. Ortiz, and P. Vandergheynst. "FREAK: Fast Retina Keypoint". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference On*. IEEE, June 2012, pp. 510–517.

[ASJ+07] H. Ali, C. Seifert, N. Jindal, L. Paletta, and G. Paar. "Window Detection in Facades". In: *Image Analysis and Processing (ICIAP 2007), 14th International Conference On*. IEEE, Sept. 2007, pp. 837–842.

[ASS+09] S. Agarwal, N. Snavely, I. Simon, S. M. Sietz, and R. Szeliski. "Building Rome in a Day". In: *Computer Vision (ICCV), 2009 IEEE 12th International Conference On*. Kyoto, Japan: IEEE, Sept. 2009.

[AT11] C. Akinlar and C. Topal. "EDLines: A real-time line segment detector with a false detection control". In: *Pattern Recognition Letters* 32.13 (Oct. 2011), pp. 1633–1642.

[Bal81] D. H. Ballard. “Generalizing the Hough transform to detect arbitrary shapes”. In: *Pattern Recognition* 13.2 (1981), pp. 111–122.

[BB10] A. Budroni and J. Böhm. “Automatic 3D modelling of indoor manhattan-world scenes from laser data”. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2010), pp. 115–120.

[BET+08] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding*. Similarity Matching in Computer Vision and Multimedia 110.3 (June 2008), pp. 346–359.

[Bra00] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb's Journal of Software Tools* (2000).

[Bro66] D. C. Brown. “Decentering distortion of lenses”. In: *Photogrammetric Engineering and Remote Sensing* (1966).

[BSB+13] R. Boutteau, X. Savatier, F. Bonardi, and J. Y. Ertaud. “Road-line detection and 3D reconstruction using fisheye cameras”. In: *Intelligent Transportation Systems (ITSC), 16th International IEEE Conference On*. IEEE, 2013, pp. 1083–1088.

[BWG15] M. Brown, D. Windridge, and J.-Y. Guillemaut. “A generalisable framework for saliency-based line segment detection”. In: *Pattern Recognition* 48.12 (Dec. 2015), pp. 3993–4011.

[Can86] J. Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986), pp. 679–698.

[Col96] R. T. Collins. “A Space-Sweep Approach to True Multi-Image Matching”. In: *Computer Vision and Pattern Recognition (CVPR), 1996 IEEE Conference On*. 1996, pp. 358–363.

[Con19] A. E. Conrady. “Decentred Lens-Systems”. In: *Monthly Notices of the Royal Astronomical Society* 79.5 (Mar. 1919), pp. 384–390.

[COS+11] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. "Discrete-continuous optimization for large-scale structure from motion". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On*. IEEE, 2011, pp. 3001–3008.

[CS08] J. Cech and R. Sara. "Windowpane Detection Based on Maximum Aposteriori Probability Labeling". In: *IWCIA Special Track on Applications*. 2008, pp. 3–11.

[CSP14] A. Cohen, A. G. Schwing, and M. Pollefeys. "Efficient Structured Parsing of Facades Using Dynamic Programming". In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference On*. IEEE, June 2014, pp. 3206–3213.

[CY99] J. M. Coughlan and A. L. Yuille. "Manhattan World: Compass direction from a single image by Bayesian inference". In: *Computer Vision (ICCV), 1999 Seventh IEEE International Conference On*. IEEE, 1999, 941–947 vol.2.

[DMM08] A. Desolneux, L. Moisan, and J.-M. Morel. *From Gestalt Theory to Image Analysis*. Red. by S. S. Antman, L. Sirovich, J. E. Marsden, and S. Wiggins. Vol. 34. Interdisciplinary Applied Mathematics. New York, NY: Springer New York, 2008.

[DZ10] F. Da and H. Zhang. "Sub-pixel edge detection based on an improved moment". In: *Image and Vision Computing* 28.12 (Dec. 2010), pp. 1645–1658.

[DZ15] P. Dollár and C. L. Zitnick. "Fast Edge Detection Using Structured Forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.8 (Aug. 1, 2015), pp. 1558–1570.

[EE11] A. Elqursh and A. Elgammal. "Line-based relative pose estimation". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On*. IEEE, 2011, pp. 3049–3056.

[Ete92] A. Etemadi. "Robust segmentation of edge data". In: *Image Processing and Its Applications, International Conference On*. IET, 1992, pp. 311–314.

[Eur16] European Machine Vision Association (EMVA). *EMVA 1288: Standard for Characterization of Image Sensors and Cameras*. Standard. 2016.

[FB81] M. A. Fischler and R. C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (June 1981), pp. 381–395.

[FFG+10] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, and S. Lazebnik. “Building rome on a cloudless day”. In: *Computer Vision - ECCV 2010*. European Conference on Computer Vision. Springer, 2010, pp. 368–381.

[FG87] W. Förstner and E. Gülch. “A Fast Operator for Detection and Precise Location of Distict Point, Corners and Centres of Circular Features”. In: *Proceedings of the ISPRS Conference on Fast Processing of Photogrammetric Data*. Interlaken, 1987, pp. 281–305.

[FK10] R. Fabbri and B. Kimia. “3D curve sketch: Flexible curve-based stereo reconstruction and calibration”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference On*. IEEE, 2010, pp. 1538–1545.

[För94] W. Förstner. “A framework for low level feature extraction”. In: *Computer Vision - ECCV ’94*. European Conference on Computer Vision. Ed. by J.-O. Eklundh. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 383–394.

[GFF10] R. Gherardi, M. Farenzena, and A. Fusiello. “Improving the efficiency of hierarchical structure-and-motion”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference On*. June 2010, pp. 1594–1600.

[GFM+07] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. “Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions”. In: *Computer Vision and Pattern Recognition (CVPR), 2007 IEEE Conference On*. IEEE, June 2007, pp. 1–8.

[GJM+18] R. Gadde, V. Jampani, R. Marlet, and P. V. Gehler. “Efficient 2D and 3D Facade Segmentation Using Auto-Context”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.5 (May 1, 2018), pp. 1273–1280.

[GMP16] R. Gadde, R. Marlet, and N. Paragios. “Learning Grammars for Architecture-Specific Facade Parsing”. In: *International Journal of Computer Vision* 117.3 (May 2016), pp. 290–316.

[HAA16] M. Hassaballah, A. A. Abdelmgeid, and H. A. Alshazly. “Image Features Detection, Description and Matching”. In: *Image Feature Detectors and Descriptors*. Ed. by A. I. Awad and M. Hassaballah. Vol. 630. Cham: Springer International Publishing, 2016, pp. 11–45.

[HBB+08] T. Hermosilla, E. Bermejo, A. Balaguer, and L. Ruiz. “Non-linear fourth-order image interpolation for subpixel edge detection and localization”. In: *Image and Vision Computing* 26.9 (Sept. 2008), pp. 1240–1248.

[HDF12] J. Heinly, E. Dunn, and J.-M. Frahm. “Comparative Evaluation of Binary Features”. In: *Computer Vision – ECCV 2012*. Ed. by A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid. Vol. 7573. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 759–773.

[HF01] S. Heuel and W. Förstner. “Matching, reconstructing and grouping 3d lines from multiple views using uncertain projective geometry”. In: *Computer Vision and Pattern Recognition (CVPR), 2001 IEEE Conference On*. Vol. 2. IEEE, 2001, pp. II–II.

[HMB14] M. Hofer, M. Maurer, and H. Bischof. “Improving Sparse 3D Models for Man-Made Environments Using Line-Based 3D Reconstruction”. In: *3D Vision, 2014 2nd International Conference On*. Vol. 1. Dec. 2014, pp. 535–542.

[Hof16] M. Hofer. “Building with Lines: Efficient 3D Scene Abstraction for the Built Environment”. Technische Universität Graz, 2016.

[HS88] C. Harris and M. Stephens. “A combined corner and edge detector”. In: *Alvey Vision Conference*. Vol. 15. Citeseer, 1988, pp. 10–5244.

[HS97] R. I. Hartley and P. Sturm. “Triangulation”. In: *Computer Vision and Image Understanding* 68.2 (1997), pp. 146–157.

[HSD+15] J. Heinly, J. L. Schonberger, E. Dunn, and J.-M. Frahm. "Reconstructing the world* in six days". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference On*. IEEE, June 2015, pp. 3287–3295.

[JKT+10] A. Jain, C. Kurz, T. Thormählen, and H.-P. Seidel. "Exploiting global connectivity constraints for reconstruction of 3D line segments from images". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference On*. IEEE, 2010, pp. 1586–1593.

[KB06] J. Kannala and S. S. Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.8 (2006), pp. 1335–1340.

[KGZ+15] M. Kozinski, R. Gadde, S. Zagoruyko, G. Obozinski, and R. Marlet. "A MRF shape prior for facade parsing with occlusions". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference On*. IEEE, June 2015, pp. 2820–2828.

[KWY14] L. Kang, L. Wu, and Y.-H. Yang. "Robust multi-view L2 triangulation via optimal inlier selection and 3D structure refinement". In: *Pattern Recognition* 47 (Sept. 2014), pp. 2974–2992.

[LCL+17] L. Liu, D. Ceylan, C. Lin, W. Wang, and N. J. Mitra. "Image-based reconstruction of wire art". In: *ACM Transactions on Graphics* 36.4 (July 20, 2017), pp. 1–11.

[LCS11] S. Leutenegger, M. Chli, and R. Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints". In: *Computer Vision (ICCV), 2011 IEEE International Conference On*. IEEE, Nov. 2011, pp. 2548–2555.

[LDB+13] C. Lopez-Molina, B. De Baets, H. Bustince, J. Sanz, and E. Barrenechea. "Multiscale edge detection based on Gaussian smoothing and edge tracking". In: *Knowledge-Based Systems* 44 (May 1, 2013), pp. 101–111.

[LM96] T. Leung and J. Malik. “Detecting, localizing and grouping repeated scene elements from an image”. In: *Computer Vision - ECCV '96*. European Conference on Computer Vision. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 14, 1996, pp. 546–555.

[LMF09] V. Lepetit, F. Moreno-Noguer, and P. Fua. “EPnP: An Accurate O(n) Solution to the PnP Problem”. In: *International Journal of Computer Vision* 81.2 (Feb. 2009), pp. 155–166.

[Low04] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110.

[Low99] D. G. Lowe. “Object recognition from local scale-invariant features”. In: *Computer Vision (ICCV), 1999 Seventh IEEE International Conference On*. Vol. 2. IEEE, 1999, pp. 1150–1157.

[LR04] S. C. Lee and R. Nevatia. “Extraction and integration of window in a 3D building model from ground view images”. In: *Computer Vision and Pattern Recognition (CVPR), 2004 IEEE Conference On*. Vol. 2. IEEE, 2004, pp. 113–120.

[LWT+15] Y. Li, S. Wang, Q. Tian, and X. Ding. “A survey of recent advances in visual feature detection”. In: *Neurocomputing* 149 (Feb. 2015), pp. 736–751.

[MAF+08] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik. “Using contours to detect and localize junctions in natural images”. In: *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference On*. IEEE, 2008, pp. 1–8.

[MFM04] D. R. Martin, C. C. Fowlkes, and J. Malik. “Learning to detect natural image boundaries using local brightness, color, and texture cues”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.5 (2004), pp. 530–549.

[MGK00] J. Matas, C. Galambos, and J. Kittler. “Robust Detection of Lines Using the Progressive Probabilistic Hough Transform”. In: *Computer Vision and Image Understanding* 78.1 (Apr. 2000), pp. 119–137.

[MHB+10] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test". In: *Computer Vision – ECCV 2010*. European Conference on Computer Vision. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 5, 2010, pp. 183–196.

[MMV16] M. Mathias, A. Martinović, and L. Van Gool. "ATLAS: A Three-Layered Approach to Facade Parsing". In: *International Journal of Computer Vision* 118.1 (May 2016), pp. 22–48.

[MMW+12] A. Martinović, M. Mathias, J. Weissenberg, and L. Van Gool. "A three-layered approach to facade parsing". In: *Computer Vision - ECCV 2012*. European Conference on Computer Vision. Springer, 2012, pp. 416–429.

[MR05] H. Mayer and S. Reznik. "Building Façade Interpretation from Image Sequences". In: *Proceedings of the ISPRS Workshop CMRT* (2005), pp. 55–60.

[MS05] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10 (2005), pp. 1615–1630.

[MS98] F. Mokhtarian and R. Suomela. "Robust image corner detection through curvature scale space". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.12 (Dec. 1998), pp. 1376–1381.

[MW14] B. Micusik and H. Wildenauer. "Structure from Motion with Line Segments under Relaxed Endpoint Constraints". In: *3D Vision, 2014 2nd International Conference On*. IEEE, Dec. 2014, pp. 13–19.

[MWA+13] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer. "A Survey of Urban Reconstruction: A Survey of Urban Reconstruction". In: *Computer Graphics Forum* 32.6 (Sept. 2013), pp. 146–177.

[MYL+11] P. Mainali, Q. Yang, G. Lafruit, L. Van Gool, and R. Lauwereins. "Robust Low Complexity Corner Detector". In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.4 (Apr. 2011), pp. 435–445.

[NDM14] W. Nguatem, M. Drauschke, and H. Mayer. "Localization of Windows and Doors in 3d Point Clouds of Facades". In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* II-3 (Aug. 7, 2014), pp. 87–94.

[NF15] I. Nurutdinova and A. Fitzgibbon. "Towards Pointless Structure from Motion: 3D Reconstruction and Camera Parameters from General 3D Curves". In: *Computer Vision (ICCV), 2015 IEEE International Conference On*. Dec. 2015, pp. 2363–2371.

[NMT13] N. Neverova, D. Muselet, and A. Trémeau. "21/2 D Scene Reconstruction of Indoor Scenes from Single RGB-D Images". In: *Computational Color Imaging*. Ed. by S. Tominaga, R. Schettini, and A. Trémeau. Vol. 7786. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 281–295.

[OV11] S. Olufs and M. Vincze. "Room-structure estimation in Manhattan like environments from dense 2 1/2D range data using minumum entropy and histograms". In: *Applications of Computer Vision (WACV), 2011 IEEE Workshop On*. IEEE, Jan. 2011, pp. 118–124.

[ÖVB+17] O. Özyeşil, V. Voroninski, R. Basri, and A. Singer. "A survey of structure from motion". In: *Acta Numerica* 26 (May 2017), pp. 305–364.

[PB10] G. V. Pedrosa and C. A. Z. Barcelos. "Anisotropic diffusion for effective shape corner point detection". In: *Pattern Recognition Letters*. Pattern Recognition of Non-Speech Audio 31.12 (Sept. 1, 2010), pp. 1658–1664.

[PG10] A. M. G. Pinheiro and M. Ghanbari. "Piecewise Approximation of Contours Through Scale-Space Selection of Dominant Points". In: *IEEE Transactions on Image Processing* 19.6 (June 2010), pp. 1442–1450.

[PGG17] V. Pătrăucean, P. Gurdjos, and R. G. von Gioi. "Joint A Contrario Ellipse and Line Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (Apr. 2017), pp. 788–802.

[RD06] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection". In: *Computer Vision - ECCV 2006*. European Conference on Computer Vision. Springer, 2006, pp. 430–443.

[Ren08] X. Ren. "Multi-scale improves boundary detection in natural images". In: *Computer Vision - ECCV 2008*. European Conference on Computer Vision. Springer, 2008, pp. 533–545.

[RL10] M. Recky and F. Leberl. "Windows Detection Using K-means in CIE-Lab Color Space". In: *Pattern Recognition (ICPR), 2010 20th International Conference On*. IEEE, Aug. 2010, pp. 356–359.

[RM08] S. Reznik and H. Mayer. "Implicit Shape Models, Self Diagnosis, and Model Selection for 3D Facade Interpretation". In: *Photogrammetrie, Fernerkundung, Geoinformation* 3 (2008), pp. 187–196.

[RM18] K. Rahmani and H. Mayer. "High quality facade segmentation based on structured random forest, region proposal network and rectangular fitting". In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-2 (2018), pp. 223–230.

[RRK+11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *Computer Vision (ICCV), 2011 IEEE International Conference On*. IEEE, 2011, pp. 2564–2571.

[SB97] S. M. Smith and J. M. Brady. "SUSAN - A New Approach to Low Level Image Processing". In: *International Journal of Computer Vision* 23.1 (May 1, 1997), pp. 45–78.

[SF16] J. L. Schonberger and J.-M. Frahm. “Structure-from-motion revisited”. In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference On*. 2016, pp. 4104–4113.

[SM16] M. Schmitz and H. Mayer. “A convolutional network for semantic facade segmentation and interpretation”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B3 (June 10, 2016), pp. 709–715.

[SMM16] Y. Salaün, R. Marlet, and P. Monasse. “Multiscale line segment detector for robust and accurate SfM”. In: *Pattern Recognition (ICPR), 2016 23rd International Conference On*. IEEE, 2016, pp. 2000–2005.

[SP10] D. Sen and S. K. Pal. “Gradient histogram: Thresholding in a region of interest for edge detection”. In: *Image and Vision Computing* 28.4 (Apr. 2010), pp. 677–695.

[SSH+15] C. Sweeney, T. Sattler, T. Hollerer, M. Turk, and M. Pollefeys. “Optimizing the viewing graph for structure-from-motion”. In: *Computer Vision (ICCV), 2015 IEEE International Conference On*. 2015, pp. 801–809.

[SSS06] N. Snavely, S. M. Seitz, and R. Szeliski. “Photo tourism: Exploring photo collections in 3D”. In: *ACM Transactions on Graphics (TOG)*. Vol. 25. ACM, 2006, pp. 835–846.

[SSS09] S. N. Sinha, D. Steedly, and R. Szeliski. “Piecewise planar stereo for image-based rendering”. In: *Computer Vision (ICCV), 2009 IEEE 12th International Conference On*. Citeseer, 2009, pp. 1881–1888.

[ST94] J. Shi and C. Tomasi. “Good features to track”. In: *Computer Vision and Pattern Recognition (CVPR), 1994 IEEE Conference On*. June 1994, pp. 593–600.

[TA12] C. Topal and C. Akinlar. “Edge Drawing: A combined real-time edge and segment detector”. In: *Journal of Visual Communication and Image Representation* 23.6 (Aug. 2012), pp. 862–872.

[TK91] C. Tomasi and T. Kanade. *Detection and Tracking of Point Features*. International Journal of Computer Vision, 1991.

[TMH+00] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. "Bundle Adjustment — A Modern Synthesis". In: *Vision Algorithms: Theory and Practice*. Ed. by B. Triggs, A. Zisserman, and R. Szeliski. Red. by G. Goos, J. Hartmanis, and J. van Leeuwen. Vol. 1883. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372.

[TŠ11] R. Tyleček and R. Šára. "A Weak Structure Model for Regular Pattern Recognition Applied to Facade Images". In: *Computer Vision – ACCV 2010*. Ed. by R. Kimmel, R. Klette, and A. Sugimoto. Vol. 6492. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 450–463.

[vGJM+10] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. "LSD: A Fast Line Segment Detector with a False Detection Control". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.4 (Apr. 2010), pp. 722–732.

[VM04] P. Vasseur and E. M. Mouaddib. "Central Catadioptric Line Detection". In: *Fifteenth British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2004, pp. 8.1–8.10.

[WDF08] S. Wenzel, M. Drauschke, and W. Förstner. "Detection of repeated structures in facade images". In: *Pattern Recognition and Image Analysis* 18.3 (Sept. 1, 2008), pp. 406–411.

[Wen16] S. Wenzel. "High-Level Facade Image Interpretation using Marked Point Processes". Bonn: Rheinischen Friedrich-Wilhelms-Universität Bonn, 2016.

[WF08] S. Wenzel and W. Förstner. "Semi-supervised Incremental Learning of Hierarchical Appearance Models". In: *Proceeding of the 21st Congress of the International Society for Photogrammetry and Remote Sensing (ISPRS)* (2008), pp. 399–405.

[WF13] S. Wenzel and W. Förstner. "Finding Poly-Curves of Straight Line and Ellipse Segments in Images". In: *Photogrammetrie, Fernerkundung, Geoinformation* 2013.4 (2013), pp. 297–308.

[WK16] D. Wolters and R. Koch. “Precise and Robust Line Detection for Highly Distorted and Noisy Images”. In: *Pattern Recognition*. Ed. by B. Rosenhahn and B. Andres. Springer International Publishing, 2016, pp. 3–13.

[WK17] D. Wolters and R. Koch. “Combined Precise Extraction and Topology of Points, Lines and Curves in Man-Made Environments”. In: *Pattern Recognition*. Ed. by V. Roth and T. Vetter. Springer International Publishing, 2017, pp. 115–125.

[WK18] D. Wolters and R. Koch. “Topology-Based 3D Reconstruction of Mid-level Primitives in Man-Made Environments”. In: *Pattern Recognition*. Ed. by T. Brox, A. Bruhn, and M. Fritz. Springer International Publishing, 2018.

[Wol14] D. Wolters. “Automatic 3D Reconstruction of Indoor Manhattan World Scenes Using Kinect Depth Data”. In: *Pattern Recognition*. Ed. by X. Jiang, J. Hornegger, and R. Koch. Springer International Publishing, 2014, pp. 715–721.

[WZ02] T. Werner and A. Zisserman. “New techniques for automated architectural reconstruction from photographs”. In: *Proceedings of the 7th European Conference on Computer Vision - Part II*. ECCV ’02. Springer, 2002, pp. 541–555.

[XO93] L. Xu and E. Oja. “Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms, and Computational Complexities”. In: *CVGIP: Image Understanding* 57.2 (Mar. 1993), pp. 131–154.

[YFP05] J. Ye, G. Fu, and U. P. Poudel. “High-accuracy edge detection with Blurred Edge Model”. In: *Image and Vision Computing* 23.5 (May 2005), pp. 453–467.

[YH83] B. Yen and T. Huang. “Determining 3-D motion and structure of a rigid body using straight line correspondences”. In: *Acoustics, Speech, and Signal Processing (ICASSP), IEEE International Conference On*. Vol. 8. Apr. 1983, pp. 118–121.

[ZK13] L. Zhang and R. Koch. "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency". In: *Journal of Visual Communication and Image Representation* 24.7 (Oct. 2013), pp. 794–805.

[ZK14] L. Zhang and R. Koch. "Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment". In: *Journal of Visual Communication and Image Representation* 25.5 (July 2014), pp. 904–915.

[ZT98] D. Ziou and S. Tabbone. "Edge Detection Techniques - An Overview". In: *International Journal of Pattern Recognition and Image Analysis* 8 (1998), pp. 537–559.

[ZXL+12] L. Zhang, C. Xu, K.-M. Lee, and R. Koch. "Robust and Efficient Pose Estimation from Line Correspondences". In: *Computer Vision – ACCV 2012*. Ed. by K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu. Lecture Notes in Computer Science 7726. Springer Berlin Heidelberg, Nov. 5, 2012, pp. 217–230.