FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Simplifying Indoor Scenes for Real-time Manipulation on Mobile Devices

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Patrick Wolf
Matrikelnummer 0726757

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: PD Dr. Martin Kampel
Mitwirkung: Dr.techn. Michael Hödlmoser

Wien, 19.04.2015

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Simplifying Indoor Scenes for Real-time Manipulation on Mobile Devices

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Media Informatics

by

Patrick Wolf
Registration Number 0726757

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: PD Dr. Martin Kampel
Assistance: Dr.techn. Michael Hödlmoser

Vienna, 19.04.2015

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Patrick Wolf
Dorfstraße 70, 6561 Ischgl

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

Zu Beginn möchte ich mich bei allen Menschen bedanken die mich beim Verfassen meiner Diplomarbeit begleitet haben. Ein außerordentlicher Dank gilt dabei meinem Betreuer Michael Hödlmoser. Ich möchte mich für seine außerordentliche Geduld, die ständige Verfügbarkeit und die vielen Diskussion bedanken. Seine wissenschaftliche Expertise, die bedingungslose Unterstützung sowie die vielen persönlichen Ratschläge und motivierenden Worte haben mir sehr geholfen. Außerdem möchte ich mich beim Team des Computer Vision Labs der Technischen Universität Wien bedanken, die es mir ermöglicht haben diese Arbeit zu verfassen. Insbesondere bedanke ich mich dabei bei Professor Robert Sablatnig für das Lehren der wissenschaftlichen Standards, sowie Martin Kampel für das Reviewen meiner Arbeit.

Der größte Dank gilt aber meiner Familie die es mir ermöglicht haben mein Studium zu absolvieren und mich auf meine Ausbildung zu konzentrieren. Eure großartige und uneingeschränkte Unterstützung, sowie die Wertschätzung die ihr mir entgegengebracht habt, werde ich niemals vergessen. Danke!

Ein besonderer Dank gilt Katharina Radisavljevič die mir während meiner Studienzeit, und besonders während dem Verfassen meiner Diplomarbeit, immer zur Seite Stand. Ich möchte mich für deine Geduld, die aufmunternden Worte und dein persönliches Engagement bedanken.

Zuletzt möchte mich auch bei meinen Freunden Matthias Dorfer, Marco Augustin, Michael Eisenkölbl und Katharina Meusburger bedanken, mit denen ich meine Studienzeit verbringen und absolvieren durfte.

Abstract

Having exact 3D reconstructions and measures of indoor scenes is useful for numerous applications *e.g.* augmented reality furniture placement. Recent 3D reconstruction approaches obtain complex and highly detailed 3D models, which are difficult to handle, since the computational cost of manipulating models is directly related to its complexity. Consequently, it is also challenging to display and manipulate such detailed models on a mobile device because of limited resources. In order to keep the processing time low, simplified approximations of highly detailed models are desirable for mobile applications. Therefore, in this thesis we present a framework for simplifying indoor scenes using multi-modal RGBD video sequences. The framework consists of two parts – a 3D layout estimation pipeline as well as an object detection and pose estimation approach. Layout segments (ground plane, walls, ceiling) are represented by 3D planes and merged over time. After determining the 2D floor plan of the fused point cloud obtained from registered shots, a compact representation of the scene is generated by extruding the floor plan. In order to create semantically meaningful 3D layouts, objects are detected and further replaced by synthetic CAD models using state-of-the-art 2D object detection methods and 3D point cloud descriptors. In each frame semantic types and poses are determined. A Markov Random Field (MRF) is introduced over time, which exploits temporal coherence between consecutive frames in order to refine the pose results. The framework is trained in an offline stage with synthetically rendered point clouds obtained from CAD models downloaded from a public database. Qualitative and quantitative experiments on various indoor video sequences show that the resulting spatial layout results outperform monocular state-of-the-art algorithms when comparing with a variety of semantically labeled ground truth scenes. The MRF optimization as well as the temporal fusion of multiple 3D layouts yield to improvements concerning the pose results and the accuracy of the scene dimensions. Moreover, in terms of the storage demand, we achieve a data reduction rate of over 99% compared to the raw point-based representations.

Kurzfassung

3D Rekonstruktionsanwendungen sind hilfreich um exakte Modelle von dreidimensionalen Szenen zu errechnen. Bezüglich Augemted Reality Lösungen, wie beispielsweise Smart Phone Applikationen mit denen Möbeltücke in einer Szene platziert werden können, ist es hilfreich die genauen Dimensionen einer Szene zu kennen. Die Berechnung von hochauflösenden Details ist dabei zweitrangig. Aufgrund limitierter Ressourcen sind besonders für mobile Applikationen vereinfachte 3D Modelle essentiell. Die vorliegende Arbeit beschäftigt sich mit der Erstellung von vereinfachten und semantischen 3D Modellen von Innenräumen. Als Datenbasis dienen multimodale RGBD Videosequenzen welche mit einem Microsoft Kinect Sensor aufgenommen werden. Die Modell-Simplifizierung besteht im wesentlichen aus zwei Teilen – einer Pipeline für die Erstellung von vereinfachten 3D Modellen sowie einer Methode um Objekte in der Szene zu detektieren und deren Pose zu erkennen. Architektonische Elemente wie Wände, der Boden und die Decke werden mit Hilfe von dreidimensionalen Ebenen modelliert. Um geometrisch erweiterte Modelle zu berechnen, werden 3D Punktwolken iterativ über die Zeit fusioniert. Basierend auf diesen Registrierungen wird eine erweiterte Grundfläche berechnet. Kompakte 3D Szenen werden erstellt indem Wandelemente und die Decke durch geometrische Extrusion simuliert werden. Um semantische 3D Modelle zu erstellen, werden Objekte in der Szene detektiert und durch CAD Modelle ersetzt. Für die multimodale Ojekterkennung wird ein State-of-the-Art 2D Bilddetektor sowie Geometrie-Deskriptoren für 3D Punktwolken verwendet. In einem initialen Detektionsschritt werden dabei in jedem Frame Objektkandidaten und deren Orientierungen ermittelt. Die berechneten Posen werden mit einem Markov Random Field (MRF) optimiert indem die Orientierungsänderung von aufeinander folgenden Objekten berücksichtigt wird. Die Klassifikationsmethode wird offline mittels synthetisch gerenderten 2.5D Punktwolken trainiert. Qualitative und quantitative Experimente basierend auf 10 Videosequenzen zeigen, dass verglichen mit photometrischen Ansätzen, die zeitlich Fusionierung zu exakteren Repräsentationen führt. Mit Hilfe der MRF-Optimierung lassen sich robustere Objektposen berechnen indem fehlerhafte Posen aussortiert werden. Bezüglich der Speicherbelastung lässt dich durch den Simplifizierungsansatz eine Datenreduktion von über 99% erreichen.

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 2
- 1.2 Contribution . . . 3
- 1.3 Thesis Organization . . . 4

2 Related Work **5**
- 2.1 3D Layout Estimation . . . 5
  - 2.1.1 3D Layout Estimation from Photometric Images . . . 6
  - 2.1.2 3D Layout Reconstruction from Range Images . . . 9
  - 2.1.3 3D Layout Estimation from RGBD Images . . . 12
- 2.2 2D/3D Object Detection, Recognition and Pose Estimation . . . 15
  - 2.2.1 2D Object Detection from Photometric Images . . . 17
  - 2.2.2 3D Object Detection from Range Images . . . 19
    - 3D Object Detection using ICP Registration . . . 19
    - 3D Object Detection using Point Cloud Descriptors . . . 20
  - 2.2.3 3D Object Detection from RGBD Images . . . 23
- 2.3 Discussion & Implications for the Proposed Methodology . . . 24
  - 2.3.1 3D Layout Estimation . . . 25
  - 2.3.2 2D/3D Object Detection and Pose Estimation . . . 25

3 Methodology **27**
- 3.1 Implementation Details . . . 27
- 3.2 3D Layout Estimation and Scene Simplification . . . 28
  - 3.2.1 Preprocessing . . . 30
  - 3.2.2 Point Cloud Registration . . . 31
  - 3.2.3 Wall Detection and Plane Fitting . . . 33
  - 3.2.4 Floor Plan Estimation . . . 35
  - 3.2.5 Floor Plan Triangulation and Wall Extrusion . . . 37
  - 3.2.6 Window Detection . . . 40
  - 3.2.7 Temporal Fusion . . . 43
    - 2D Floor Plan Estimation . . . 43
    - 3D Layout Estimation and Simplification . . . 45
- 3.3 3D Object Detection and Pose Estimation . . . 45

3.3.1 Synthetic CAD Model Training . . . 46
Partial Point Cloud Generation . . . 47
Surface Normal Estimation . . . 48
Point Cloud Descriptor (VFH) . . . 49
3.3.2 From 2D to 3D - Initial RGB Object Detection . . . 50
3.3.3 3D Object Cluster Estimation and Euclidean Clustering . . . 52
3.3.4 Feature Description and Classification . . . 53
3.3.5 PCA Optimization . . . 54
3.3.6 Temporal Fusion and Pose Optimization . . . 56
Unary Potentials . . . 57
Binary Potentials and Model Type Estimation . . . 58
3.4 Discussion . . . 59
3.4.1 3D Layout Estimation . . . 59
3.4.2 3D Object Detection and Pose Estimation . . . 60

4 Results and Experiments **61**
4.1 Data . . . 62
4.1.1 RGBD Shots and Videos . . . 62
4.1.2 CAD Models . . . 63
4.2 3D Layout Evaluation . . . 63
4.2.1 Single Shot 3D Layout Evaluation . . . 64
Qualitative Experiments . . . 64
Qualitative Results . . . 65
Quantitative Experiments . . . 74
Quantitative Results . . . 75
4.2.2 Multiple RGBD Shots Evaluation . . . 76
Experimental Setup . . . 76
Qualitative Results . . . 76
Quantitative Results . . . 79
4.3 Memory Capacity, Data Reduction ans Use Cases . . . 81
Experimental Setup . . . 82
Quantitative Results . . . 82
Qualitative Results . . . 82
Use Case - Google SketchUp Scene Composer . . . 83
4.4 3D Object Classification Evaluation . . . 84
4.4.1 Single 3D Object Classification Evaluation . . . 84
Experimental Setup . . . 84
Quantitative Results . . . 85
4.4.2 MRF optimized 3D Object Classification Evaluation . . . 86
Experimental Setup . . . 86
Quantitative Results . . . 87
Qualitative Results . . . 87
4.5 Discussion . . . 88

5 Conclusion and Future Work **91**
5.1 Conclusion 91
5.2 Future Work 92

Bibliography **95**

Nomenclature **103**

CHAPTER 1

Introduction

Humans are able to immediately conceive the spatial layout of indoor scenes and to recognize all objects visible in the setting [31]. The human visual system is able to rapidly understand and classify even complex setups [20]. Understanding the scene and reconstructing the geometry of indoor scenes are well-studied areas of research and useful for a numerous of applications *e.g.* indoor navigation [32], Simultaneous Localization And Mapping (SLAM) [1], room organisation [10] or augmented reality [11]. In general, scene understanding in terms of indoor scenarios attempts to automatically determine the extents of the floor, the walls, and the ceiling. Objects visible in the scenes provide semantic cues which help to classify the scenes [10, 20]. Visual recognition is an active field of research, and numerous applications have been developed based on photometric images, range images or multimodal color/depth data [35].

In the past few years depth cameras have gained large attention from computer science researchers. One reason is the availability of low budget 3D cameras like the Microsoft Kinect sensor which is a cheap alternative to expensive structured light scanners. Recent developments like Google's project Tango[1] capture depth information and enable to obtain a human-scale understanding of indoor scenes even on a mobile device. In the field of computer vision the objective is to make the applications more robust by using the additional depth information. Examples can be found in the area of room layout estimation [60, 66], three-dimensional reconstruction [24, 39, 72] and object classification [2, 3, 57].

In contrast to applications like Microsoft's KinectFusion[2] [39] or Google's project Tango, where millions of 3D measurements are stored in order to create exact 3D reconstructions, for augmented reality applications a *detailed* reconstruction implies to provide precise scene dimensions [51]. Sample questions in terms of augmented reality approaches could be "Does this couch fit into the scene?" or "What is the maximum permissible dimension of the couch?" To answer such questions, it is important to have a rough but exact and metric 3D layout of the underlying environment. Besides exact 3D models, it is also necessary to include semantic information into the estimated 3D layouts. Possible use cases are scene compositions in which

[1] `https://www.google.com/atap/projecttango/` (last access: 17.04.2015)

[2] `research.microsoft.com/projects/surfacerecon/` (last access: 17.04.2015)

objects visible in the scene are removed, modified or exchanged by virtual objects. For an efficient manipulation of such 3D representations in real-time, *e.g.* on a mobile device, it is essential to provide simplified 3D models which consist of a minimum number of faces and vertices in order to keep the computational power and the the memory demand to a minimum. Simplification methods are used to create aesthetically pleasing 3D models which allow operators to capture and navigate through environments [66].

1.1 Problem Statement

The goal of this thesis is to obtain compact and semantically meaningful 3D layouts of indoor scenes captured using a Red Green Blue Depth (RGBD) sensor. Therefore we propose a simplification framework which consists of two parts – a 3D layout estimation method and an object detection and pose estimation approach. The simplified 3D layouts consist of 3D planes describing the ground plane, wall segments and the ceiling. Apart from using geometric information, semantic 3D maps are created by detecting objects in the scene and replacing them by using existing Computer-Aided Design (CAD) models gathered from a web-based 3D object database. For both, the 3D layout estimation as well as the object detection approach, the multimodal characteristics of RGBD data (color and intensity images) are used.

As an input we are faced with RGBD video sequences captured using a Kinect sensor (Kinect ifor Xbox 360). A typical depth sensor returns noisy point based data because of missing depth information. This results in point cloud representations with *holes*. Reasons for these artefacts are occlusions, specular surfaces or direct illumination. A further problem of 3D reconstructions obtained by using structured light techniques is that the 3D models consist of a high number of points, especially if multiple models are fused into a global model. Since the computational cost of processing a 3D model is directly related to its complexity [25], simplified models are useful for post-processing tasks or the manipulation of the models on mobile devices. To overcome the mentioned problems, we create compact mesh representations of indoor scenes with a minimum number of vertices and faces. In addition to the surface reconstruction, environmental objects are detected and replaced by complete CAD models. The simplified 3D layouts should be exported into a common 3D files. The objective is to load and manipulate these outcomes on a mobile device in real-time or to modify the 3D models using a third-party CAD software.

Image-based layout estimation methods describe the spatial layout of indoor scene by fitting a 3D box into the scene [10, 29, 71]. Hence, such frameworks are limited on convex geometries. We therefore present a method which is able to process concave ground plans as well. Our object classification method is trained offline, which is an improvement over approaches which are trained online with real-data captured by a sensor (*e.g.* [34, 56, 57]).

The aforementioned goals of this thesis are illustrated in Figure 1.1. Several RGBD input shots are fused in order to obtain a dense point-based representation of the input scene. Layout segments (ground plane, walls, ceiling) are used to describe the scene in a highly simplified manner. Objects are detected and further replaced by complete CAD models in order to obtain semantically meaningful scene layouts. The data reduction allows the manipulation of the simplified models on a mobile device in real-time.

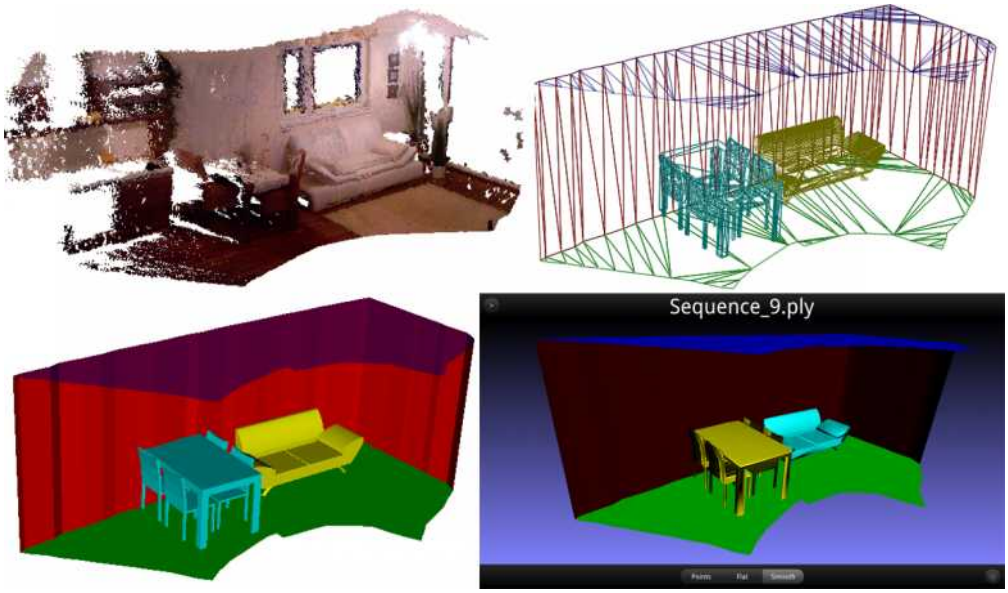


Figure 1.1: A simplicifaction framework is used to describe fused point clouds with 3D planes. Objects visible in the scene are detected (type and pose) and replaced by full 3D models from a synthetic CAD model dataset. From top left to bottom right: Fused point cloud coming from multiple shots, wire-frame representation of the simplified 3D layout and real-time manipulation on a mobile device.

1.2 Contribution

Previous work shows that monocular Red Green Blue (RGB) images allow the estimation of scene layouts using edges, color features and vanishing points [30, 36, 59]. Additional depth information helps to reconstruct the scenes in an accurate way [4, 39, 75]. Moreover, object recognition methods allow the estimation of additional semantic labels which help to classify the scenes [5, 10, 66]. Thus, state-of-the-art object classification algorithms are available for both – 3D point clouds [2, 3, 57] as well as 2D image data [21]. The contribution of our work concentrates on three areas:

1. The strengths of the aforementioned research areas are combined – (Depth, RGB, Object detection). Single 3D layouts are estimated and object candidates are determined in each frame. The results are expanded and refined using a temporal fusion method, while still keeping the complexity and the memory demand low, which allows the real-time manipulation on mobile devices.

2. The entire 3D model (ground plane, walls, ceiling) is described by multiple 3D planes. The corresponding 3D layouts are merged over the time using the transformations obtained from the camera pose problem between consecutive input frames.

3. Different to related work, objects are not represented using bounding boxes [10, 21, 27] or labeling-based inference [29, 30, 37]. Instead we use a pre-defined database of synthetic CAD models downloaded from the web to describe the type of the detected objects as well as the corresponding orientation within the 3D layout.

1.3 Thesis Organization

This thesis is divided into four chapters:

1. **Chapter 2** investigates existing approaches in the field of object recognition and scene understanding. In Section 2.1 we discuss methods for 3D reconstruction and layout estimation. Section 2.2 presents object detection and pose estimation approaches. The chapter concludes with a brief discussion and suggestions for the methodological approach in Section2.3.

2. **Chapter 3** introduces a simplification framework for 3D layout estimation and object classification and presents the underlying methodology. **First**, implementation details are described in Section 3.1. **Second**, the pipeline for the estimation of 3D layouts is presented in Section 3.2. **Third**, in Section 3.3 we outline the object detection and pose estimation pipeline. A brief summary is presented in 3.4.

3. **Chapter 4** presents qualitative and quantitative experiments for evaluating the presented methodology. **First**, Section 4.1 outlines the data sets used to train and evaluate the presented methods. **Second**, in Section 4.2 we present the evaluation of the 3D layout estimation framework. Therefore, in Subsection 4.2.1 we present 3D layout results by processing single RGBD shots. Optimized 3D layout results and improvements obtained from multiple RGBD frames are presented in Subsection 4.2.2. **Third**, In Section 4.3 we conduct with possible use cases and results of the data reduction experiments and show the ability to process the simplified models in real-time on mobile devices. **Fourth**, Section 4.4 discusses the object detection and pose estimation results based on single frames (see Subsection 4.4.1) as well as video sequences and highlights the improvements achieved by the presented MRF optimization approach (see Subsection 4.4.2). In Section 4.5 the overall results are summarized.

4. **Chapter 5** concludes with an overall summary of the thesis and gives suggestions for further research.

CHAPTER 2

Related Work

Since the objective of this thesis is the estimation of semantically meaningful 3D layouts, in this chapter we summarize relevant methods of the two research areas – 3D layout estimation and object classification. Thus, we discuss strengths and weaknesses of 3D reconstruction and layout estimation methods, which focus on indoor environments. In order to estimate semantic attributes, object recognition methods are useful. Therefore, we summarize object detection and pose estimation approaches. For both – 3D layout estimation approaches as well as object recognition methods – we present works which are based on photometric images, depth (range) images / point clouds as well as multimodal RGBD data.

2.1 3D Layout Estimation

Humans have a great ability to recognize and understand the structure of indoor scenes from monocular images [31]. Even clutter present in the scene (*e.g.* objects) does not restrict the human spatial understanding. Finding the underlying 3D structure of a scene means to identify the *ground plane*, the *walls*, the *ceiling* and the *objects* visible in the environment [29]. In terms of computer vision research, this is known as scene understanding, where the goal is to assign each pixel of an image with an additional semantic information. Finding the 3D structure from an image is important for developing autonomous systems where the goal is to navigate in the environment and interact with it [72]. Besides the spatial understanding of the geometric structure of indoor scenes, humans are also able to immediately recognize objects in the environment and understand the geometric configuration between the 3D layout and these objects [10]. While objects are described as clutter (*e.g.* [29, 30]), it is also common to apply separate object detection techniques [21, 61].

Finding the 3D layout of an indoor scene from monocular imagery is extremely difficult [72]. Therefore, depth sensors can be employed in order to overcome the ambiguities of photometric images. In the following we discuss 3D layout estimation methods based on photometric images [10, 29, 46], depth images [39, 66, 68] as well as multimodal RGBD data [24, 60, 72]. Moreover,

we briefly discuss the differences between layout estimation methods and 3D reconstruction approaches [39]. The latter ones (in general) do not focus on the semantic understanding of the scenes [1]. The goal of such applications is to estimate highly detailed 3D scene reconstructions.

2.1.1 3D Layout Estimation from Photometric Images

In Figure 2.1a an image of a typical indoor scene is shown. Numerous methods exist which estimate the spatial layout of such monocular images [30]. Hoiem *et al*. [36] present a learning based method which uses geometric classes in order to assign each pixel of an input image with an additional semantic label. The label information for each pixel is added by using local surface orientations. Different to labeling-based methods, in [10, 29, 36] the problem of scene understanding is solved using 3D-to-2D mapping techniques. Such approaches model the layout of a scene by fitting a parametrized 3D cuboid (also known as 3D box layout) into the scene (cf. Figure 2.1b). Since indoor environments are man-made, the *Manhattan world* assumption is used to estimate 3D box layouts [10, 29–31].

(a) (b)

Figure 2.1: Spatial layout understanding from single images. (a) RGB sample image in which objects (clutter) covers parts of the underlying 3D layout. The scene is described by a 3D box layout (black lines) and surface labels (pink areas) which represent objects. (Image taken from [29])

Manhattan World Assumption The Manhattan world assumption starts from the premise that three dominant vanishing points are available in the scene [12]. Therefore, under perspective projection typically parallel lines are extracted which intersect in the image plane at the vanishing points [29]. In terms of indoor environments, the Manhattan world assumption satisfies that visible surfaces are along one of the three orthogonal directions [60]. Thus, in a monocular setting the objective is to find a cuboid which aligns with these three orthogonal orientations [72] (cf. Figure 2.2). The bottleneck of the Manhattan world assumption is that the method assumes that the room is supported by a 3D box. Thus, it is difficult to model more complex geometries (*e.g*. a scenes with a convex floor plan or rooms in which more then 4 walls are visible).

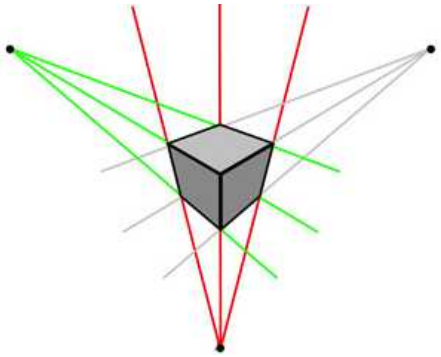


Figure 2.2: A 3D layout defined by a cuboid using a 3 point perspective. (Images taken from [24])

Hedau *et al*. [29, 30] present a framework which describes 3D layouts using a 3D box layout and surface labels (cf. Figure 2.1b). The 3D box layout models the empty room. The surface labels provide information about objects visible in the scene as well as more precise surface information (*walls*, *floor* and *ceiling*). The overall framework is shown in Figure 2.3.

(a) (b) (c) (d) (e)

Figure 2.3: Spatial layout estimation. (a) Edge segments are extracted from the RGB image. (b) Based on these edges, vanishing points are calculated. (c) Multiple layout hypothesis are estimated an sorted using a structure learning method. (d) Surface labels (*wall*, *floor*, *ceiling*, and *clutter*) are calculated based on confidence maps which are generated from edge, color and texture features. (d) <Surface labels allow a feature re-estimation which results in more robust 3D box layouts. (Image taken from [29]

The approach first estimates edges and groups these edges into three mutually orthogonal vanishing points. Second, multiple 3D box candidates are created by sampling rays from the vanishing points. The box layouts are ranked using edge features and learned models. Therefore, it is examined how well the layouts matches with ground truth layouts. In general, it is difficult to align a box layout with a general setting because of clutter (*e.g*. furnitures) which leads to inaccurate alignments. Thus, in a third step surface labels (segments) are estimated using a modified version of the superpixel algorithm presented by Hoiem *et al*. [36]. A confidence map is calculated over the segments and based on this map, the features which have been used to rank the raw box layouts are re-estimated. Finally, the box layouts are re-ranked based on these optimized features. The basic idea of the approach is that a more representative 3D layout can be estimated if the clutter in the scene is known in advance (*e.g*. a wall can not intersect with an

object). The workflow is shown in Figure 2.3.

The disadvantages of the method are that the 3D box layouts are not suitable to represent more complex scenes (*e.g.* concave floor plans) and that scenes which show only a small portion of a room are nor represented accurately. Moreover, the approach calculates clutter but does not incorporate semantic information about the clutter.

Choi *et al.* [10] introduce an improved version of the layout estimator presented by Hedau *et al.* [29]. Therefore, a scene-object interaction method provides information about how the scene type influences the object detection results. Furthermore, an object-layout interaction method examines how objects are placed in the 3D layouts, and vice versa. The scene understanding framework is threefold: A scene classifier [45] is used to estimate the scene type (*e.g.* dining room). An object detector [21] is applied in order to detect objects visible in the scene and a layout estimator [29] generates 3D box layouts which represent the underlying room geometry. The object detector is trained with data from the PASCAL dataset [18]. The semantic 3D layouts are estimated by an image parsing method in which a parse graph is used to specify relations between the components. The root node defines the 3D layout as well as the scene type. Leave nodes represent the objects detected in the scene. The core of the framework consists of *3D Geometric Phrases* which encode relations between several objects and object groups. The graph which best fits with the scene configuration is calculate by applying an energy maximization method (see [10] Section 3 for details). Figure 2.4 shows the overall worklow of the approach.

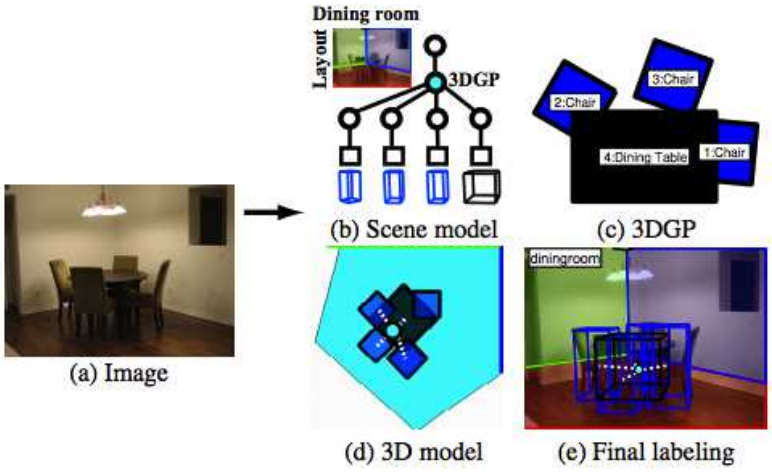


Figure 2.4: The entire scene is defined by a scene classification, detected objects and a 3D box layout. (a) A single input scene is represented by creating (b) a parse scene graph. (c) 3D geometric phrases are used to describe the relations among several objects. (d)-(e) The final 3D model consists of object groups and a 3D box layout which interacts in a physically valid way. The box layout in (d) is represented by the colored lines. (Image taken from [10])

Compared to Hedau *et al.* [29], the 3D layouts provide additional semantic information by including object detection results as well as information obtained from a scene classification method. Moreover, the 3D box layout hypotheses are modified because the method satisfies that objects and layouts interact in a physically correct way. A further advantage is that objects

are represented in the 3D space. Thus, relations between objects as well as constraints between objects and the box layouts are formulated (*e.g.* a wall can not intersect with an object's 3D bounding box). Since the scene geometries are defined by box layouts, it is still a problem that the representation of more complex layouts (*e.g.* concave floor plan) is inaccurate.

Lee *et al.* [46] present a layout estimation approach which is similar to Choi *et al.* [10]. The method formulates relations between 3D layouts and objects using volumetric constraints. Different to [29], the objects are described by parameterized 3D volumes instead of projections onto the 2D image. The volumetric representations are used to formulate configuration hypothesis. The idea is to create configurations in which all objects are defined by exclusive 3D volumes which are correctly placed (*e.g.* do not intersect with walls) inside the volume of the corresponding 3D layout. Therefore, volumetric concepts *i.e.* spatial exclusion or containment are applied. Figure 2.5 illustrates the idea of examining physical validity. Beside the limitation that 3D boxes are used to describe the layouts, the approach does not incorporate semantic information [10].

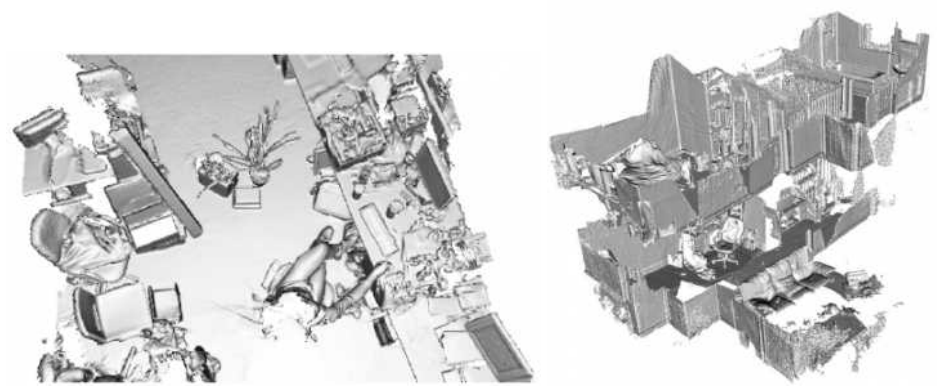
Figure 2.5: The usage of 3D volumetric reasoning results in more accurate 3D layouts. (left) Sample images. (middle) Spatial layouts estimated using geometric cues. (Right) Improved 3D layout results using physical valid configurations. (Image taken from [46])

2.1.2 3D Layout Reconstruction from Range Images

Different to approaches which are based on photometric images, methods which rely on range images are helpful to create more detailed layout models and exact 3D reconstructions of indoor scenes [66]. In this subsection we want to present methods ([39,66,68], Google's *Project Tango*) which focus on the usage of pure depth information.

The *KinectFusion* approach presented by Izadi *et al.* [39] generates detailed 3D reconstructions of indoor scenes in real-time. The approach uses 3D data coming from a Microsoft Kinect camera. (Please note that only the depth information is used from the RGBD input images.) The dense reconstructions are generated by moving a Kincet camera within a room. In the following we briefly summarize the reconstruction pipeline. Since the approach is based on point clouds, in an initial step the range images provided by the Kinect are converted into 3D point-based representations. The approach continually tracks the 6 Degrees Of Freedom (DOF) pose of the Kinect sensor and fuses the point clouds into a global model. Therefore, a fast Graphics

Processing Unit (GPU) implementation of the Iterative Closest Point (ICP) algorithm is used to estimate the rigid transformations which aligns multiple point clouds. A volumetric surface representation which is based on a Truncated Signed Distance Function (TSDF) volume [39] is used to represent the global model. New point clouds are integrated into the global model by updating the voxel grid of the corresponding volume. For the volume visualization from a specific camera pose, a raycasting algorithm is used. Moreover, it is possible to generate a point cloud or a mesh representation of the entire volume. Figure 2.6a shows an rendered scene obtained by the KinectFusion software.



(a) (b)

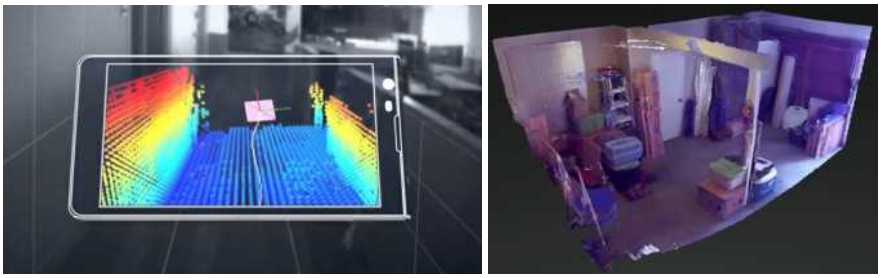
Figure 2.6: Highly detailed 3D reconstructions. (a) Rendered volume generated with the KinectFusion approach. (b) Extended 3D reconstruction based on a spatially extended KinectFusion algorithm. (Images taken from [64] and [68])

The KinectFusion approach has a couple of advantages compared to image-based layout estimation methods. Since the method relies on dense depth information, highly detailed 3D reconstructions are generated [64]. Furthermore, the framework is not limited on static images and no assumptions about the structure (*e.g.* Manhattan world assumption) are required. The bottleneck of the approach is that the models get large in terms of storage demand and that the scenes are limited to a maximum room size [68]. With the lowest resolution, the maximum volume size that can be scanned is up to around $8m^3$. Compared to scene understanding approaches like [10,29,36], the KinectFusion approach does not incorporate semantic information about the scene layout or the objects visible in the environment.

Whelan *et al.* [68] propose an extension of the KinectFusion approach by using a mesh-based mapping technique. The algorithm incrementally adds new point clouds into a global mesh model. Thus, the method creates 3D models over extended areas which is not possible with the KinectFusion application [68]. The usage of multi-threaded components enables the generation of the spatial extended 3D models in real-time. The high number of vertices/points (*e.g.* 2.3×10^6 faces for a sample model) as well as the lack of missing semantic information are still challenging problems. Figure 2.6b shows an sample output created with the spatially extend KinectFusion application.

Recent projects like Google's *Project Tango* create detailed and colored 3D reconstructions

of indoor environment in real-time even on a mobile device. Similar to the Microsoft Kinect sensor, an Android smartphone device uses a RGBD sensor to capture multimodal data. Different to the KinectFusion approach and [68], the 3D point cloud representations hold additional color information. (Please note that although the application uses RGB images for color mapping, the focus is on estimating point-based models.) Figure 2.7a shows a demo project which shows the 3D-sensing abilities of Project Tango.

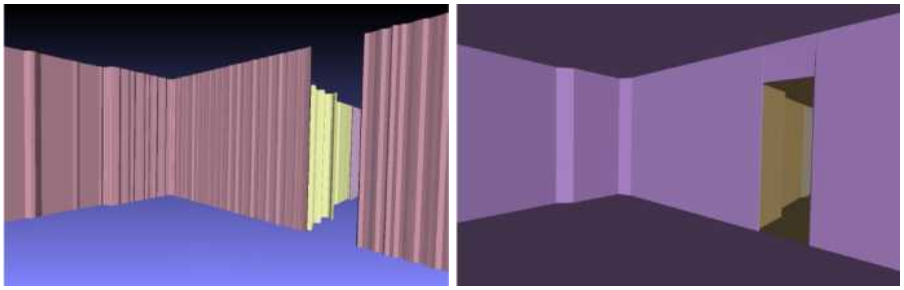


(a) (b)

Figure 2.7: Project Tango. (a) 3D Reconstructions are obtained in real-time by a mobile device. (b) Colored 3D reconstruction of an indoor environment created with Project Tango. (Images taken from the web)

Turner *et al*. [66] introduce a real-time approach which generates simplified and aesthetically pleasing 3D reconstructions from dense 3D point clouds which are captured by a mobile mapping systems. The algorithm completely ignores objects visible in the scene. The focuses lies on the estimation of exact 3D layouts which can be used in simulation approaches or for navigation in SLAM applications. First, the approach generates a set of 2D wall samples. Therefore, the 3D points of a scene are projected onto the floor. A Delaunay triangulation method [62] is used to create a 2D mesh of the corresponding floor plan. The scene is divided into interior and exterior triangles and redundant triangles are removed. Finally, 3D models are estimated by extruding the 2D floor plan using a heigh information. In order to reduce the number of faces, the entire 3D model is simplified using the simplification algorithm proposed in [25]. Since the approach creates smooth 3D meshes, it is possible to add texture information. Figure 2.8 shows a sample output of a reconstructed indoor model. The simplified models created by Turner *et al*. [66] are more usable (in terms of storage demand) compared to the detailed 3D reconstructions proposed in [64, 68]. The bottleneck of the simplification approach is that the approach does not create semantic maps.

Compared to image-based methods which use a cuboid to describe the underlying geometry of a scene, an advantage of the point-based methods presented is that it is possible to process complex scenes. Since the point-based methods we have discussed are not based on the Manhattan world assumption, concave layouts are no limitation for the frameworks (cf. floor plan of the scene in Figure 2.8). Moreover, the depth-based methods we have presented create extended 3D layouts over time using multiple frames/video data. This is an further advantage compared to image-based layout estimation approaches which are generally limited on single images (*e.g*. Hedau *et al*. [29]).

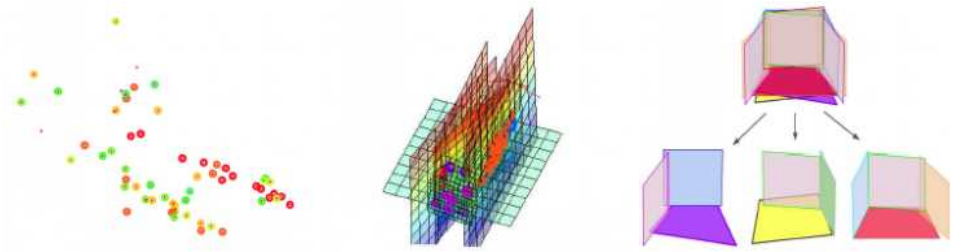


(a) (b)

Figure 2.8: (a) Watertight 3D models are created by extruding the 2D floor plan into a 3D model. (b) A simplification method creates smooth 3D layouts with a reduced number of faces. (Images taken from [66])

2.1.3 3D Layout Estimation from RGBD Images

Depth-based 3D reconstruction approaches (*e.g.* [39, 68]) generate highly detailed 3D reconstruction. Different to those, Furlan *et al.* [24] introduce a probabilistic framework for 3D layout estimation which estimates simplified 3D models based on a combination of structural reasoning and motion estimation. Figure 2.9 shows a pictorial representation of the workflow. First,



(a) (b) (c)

Figure 2.9: 3D reconstruction using multiple 3D planes. (a) Based on a sparse 3D point cloud model, (b) multiple layout candidates are estimated using a plane-fitting method. (c) The final layout consists of randomly combined layout candidates. (Images taken from [24])

the framework estimates sparse 3D point clouds and the camera motion from an input image sequence. Second, these 3D point are used to generate a higher level representation based on 3D planes. Therefore, several planes are fitted into the 3D points in order to obtain a large number of (potentially inaccurate) layout components *i.e. walls* and *floor*. The 3D planes are estimated by a RANdom SAmple Consensus (RANSAC) plane fitting approach. Third, a 3D layout is generated by randomly combining the layout components. An inference is calculated by measuring (1) the compatibility of each frame using visual observations in the images (*e.g.* edges, lines and regions) and (2) the geometrical constraints over adjacent frames. Thus, the layout components at each time step are locally adjusted, merged or stitched and finally the resulting 3D layout is evaluated using a probabilistic scoring method. The objective is to find a configuration which

accurately describes the underlying scene. Figure 2.10 shows sample 3D layouts created with the probabilistic layout estimation framework. Compared to detailed 3D reconstructions, the layouts consist of multiple 3D planes instead of dense point clouds.

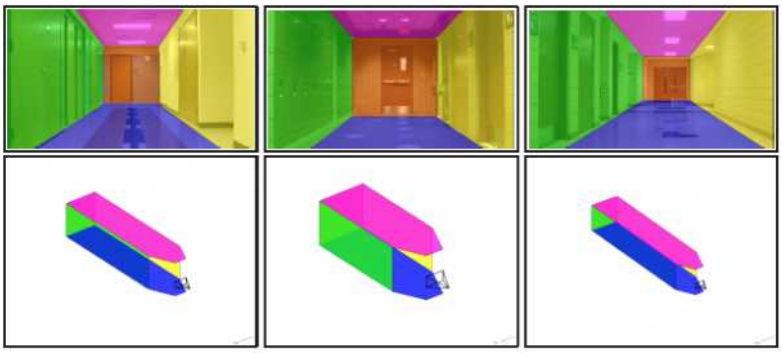
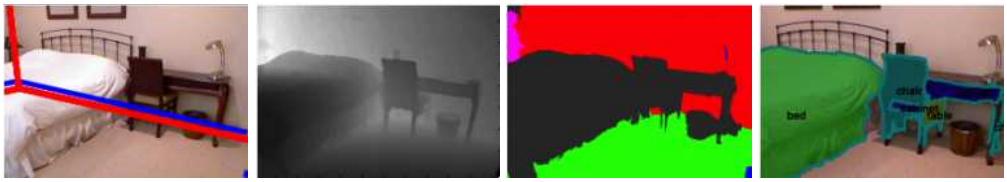


Figure 2.10: The simplified 3D reconstructions consist of a set of 3D planes which are arranged over time. (Image taken from [24])

A major advantage of the probabilistic simplification method proposes in [24] is that no assumptions about the scene geometry (*e.g.* Manhattan world assumption, known camera height) are claimed. Moreover, the multiple plane representation consists of a reduced number of points and is generated in real-time using a multi-threading technique. A bottleneck of the approach is that semantic information (*e.g.* object detection or scene classification) is completely missing.

Zhang *et al.* [72] propose a semantic parsing approach to jointly estimate (1) the 3D layout of indoor scenes and (2) clutter present in the environments using appearance and depth features. Different to the previous presented method [24], the layout estimation method satisfies the Manhattan World assumption concerning color and depth cues. Figure 2.11 shows an 3D layout created with the framework. In the following we briefly summarize the approach.



(a) (b) (c) (d)

Figure 2.11: Jointly 3D layout estimation. (a) RGB input image with the proposed d 3D layout (blue) and the ground truth layout (red). (b) Corresponding depth image. (c) Pixel-wise labeling inference and (d) scene classification. (Images taken from [72])

First, the superpixel algorithm of [36] is used in order to partition the color/depth image into planar surfaces. The problem is solved my minimizing a term obtained from shape, appearance and depth energies. The idea is to estimate superpixels which (1) are piecewise planer, (2) consist of pixels with a similar appearance and (3) provide regular shapes. Second, a Conditional Random Field (CRF) is applied to formulate an energy which encodes both – layout and label

(clutter) information. In particular, the energy used consists of a labling term, a layout term and a terms which describes the relation between the two energies. (For more detailed information about the formulation of the energies, please see Section 3.2 in [72]). Third, a pixel-wise inference is calculated using an iterative minimization algorithm.

One benefit of the semantic parsing approach is that different to [24, 39, 72], the method estimates clutter. Moreover, an method is introduced which segments and classifies this clutter. Therefore, the pixel-wise classification algorithm proposed in [70] is used. An disadvantage of the approach is that it is not possible to exploit RGBD video data. Furthermore, objects are classified using labeling-based inference which is a limitation compared to methods which provide detected objects in the form of 3D cuboids.

Similar to [72], Silberman *et al*. [60] propose a approach which estimates semantic 3D layouts from RGBD images by using surfaces and object labels. The approach introduces an additional technique to create relations between layout segments and objects. The overall worflow is illustrated in Figure 2.12.

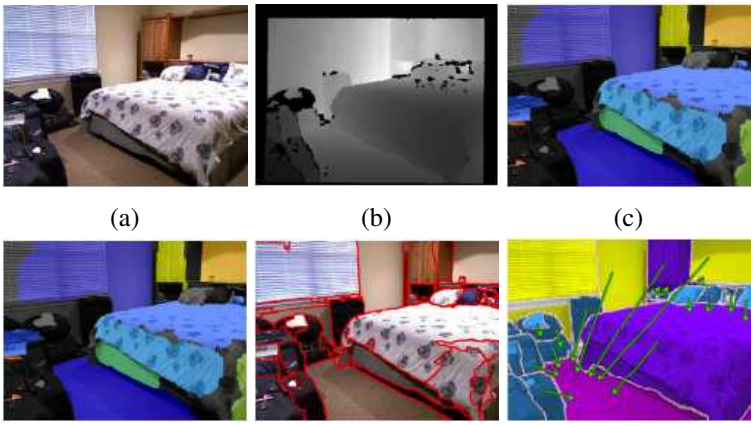


Figure 2.12: From top left to bottom right. (a)-(b) Multimodal RGBD input images are used to estimated surface normals. (c) These normals are aligned with the tree orthogonal scene directions. (d) 3D planes are estimated using a RANSAC plane-fitting method. (e) The scene is segmented into multiple regions based on color and depth cues. (f) Physical support relations are defined on these regions. (Image taken from [60])

In a first step, surface normals are calculated from the depth images. Taking into account the Manhattan world assumption, the 3D measures are aligned with the orthogonal scene coordinates. The goal is to obtain 3D floor points/vectors which point upwards. Second, potential *floor*, *wall*, *ceiling* and *support* planes are estimated using a RANSAC plane-fitting algorithm. Each pixel of the color image is assigned to a 3D plane using a graph cat optimization algorithm combined with an alpha expansion method. Therefore, information coming from the RGB image, the normals and the 3D points is used. Third, the image is segmented into surface and object instances. A watershed algorithm which is consist within the depth planes is used to create an

oversegmentation. These superpixels are then iteratively merged based on learned similarities. Therefore, the method proposed in [36] is adapted. In a finale stage, and different to all previous presented frameworks, the approach models relationships between the estimated regions (layout and objects). These relations represent physical interactions between several regions. Such relations are important in more complex robotic applications in which objects interact with each other (*e.g.* if a *book* is placed on a *cup*, first the *book* has to be lifted in order to grab the *cup*). The support relations are estimated using an energy minimization method over the support regions, the support types and the structure classes (see [60] Section 5 for more details). The major disadvantage of the approach is that only static imaged are processed.

While semantic information is missing in depth-based frameworks like *KinectFusion* [39], in a number of monocular applications (*e.g.* [29]) objects are represented as clutter. More advanced approaches (*e.g.* [10]) use separate object detectors to detect and classify objects visible in the scene.

2.2 2D/3D Object Detection, Recognition and Pose Estimation

The automatic and semantic understanding of a scene as well as the detection and recognition of all visible objects is a challenging task. Objects detected in indoor environments make it possible to estimate semantic meaningful 3D layout [1, 60, 72]. Humans are able to detect objects from different viewpoints, despite large variation in shape, changing illuminations and hight intra-class variabilities [34]. Hence, object class recognition and pose estimation are fundamental challenges and well-studied in computer vision [64]. The recognition problem can be divided along several axes (as proposed in [64]):

- **Object detection:** Describes the problem of detecting areas in an image where matches may occur. An example is the build-in function of a digital camera to detect faces or persons in photometric images (*e.g.* [67]).
- **Instance Recognition:** Describes the challenge of re-recognizing a known object in a rigid form. Typical applications deal with the detection of objects which are viewed from different viewpoints in highly cluttered scenes with the additional problem of partial occlusion. Such problems are solved by matching extracted contours, lines or surfaces against an pre-defined object database (*e.g.* [48]).
- **Category Recognition:** Deals with the recognition of instances of a certain class (*e.g.* furnitures) and is therefore the most challenging recognition task because of the high number of possible variations within a certain class. Feature-based algorithms are used to solve the category recognition problem. Possible approaches rely on pure features (*e.g.* *Bag of Words approach* [13]) or the relative position between features belonging to a detected object (*e.g.* *part-based models* [22]).

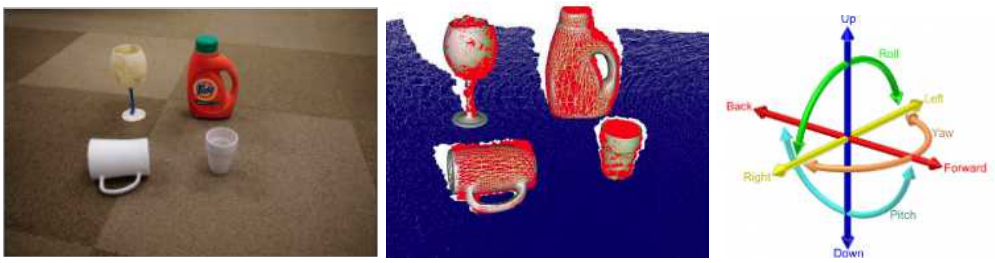
Figure 2.13 illustrates the difference between the three mentioned recognition tasks.

In this thesis we focus on the problem of category recognition (in the following also called object detection/classification). Typical object recognition applications can be found in the field

(a) (b) (c)

Figure 2.13: Different object recognition tasks. (a) Object detection methods are used to detect faces. (b) Instance (known object) recognition based on contour matching techniques. (c) Feature-based recognition is used to search for a specific class within a scene. (Image taken from [64])

of robotics [3], where the problem can be described as a manipulation task: In order to manipulate a segmented object, first a robot has to identify object candidates. Secondly, the 6 DOF pose has to be estimated, in order to manipulate (*e.g.* to grab) an object. The 6 DOF describe the possible movement of a rigid object in the three-dimensional space. These movements are defined by three translations along the axis of a coordinate system (up/down, left/right, forward/back) and the rotations around the axis. The rotations are also known as yaw, roll and pitch. Figure 2.14 shows a typical scenario from the area of robotics.



(a) (b) (c)

Figure 2.14: Robotic Object Detection. (a) Objects placed on a table should be grabbed. (b) Segmented object candidates are classified and the 6 DOF poses are determined. (c) The 6 DOF pose is defined by translations along and rotations around the coordinate system. (Images taken from [3])

Object recognition methods have been introduced based on photometric images, depth (range) images and recently on RGBD images [35]. In the following subsections we briefly summarize relevant works and methods.

2.2.1 2D Object Detection from Photometric Images

Image-based object detection methods can be divided into two categories — learning-based and template-based approaches [34]. Template-based methods try to describe a model (*e.g.* a face) as a function. The advantage of template-based methods is that they do not require an expensive training stage. Moreover, *new* models can be trained online. Learning-based methods describe a model based on a huge amount of pre-defined data. Such methods are also known as feature-based recognition. Models are describes using a set of features which are used to train a classifier. Object are detected by extracting features and matching those features against a pre-defined database. Such methods work well for objects of a particular class (*e.g.* cars or pedestrians) [21, 34]. A disadvantage is that learning-based methods lead to large training datasets and therefore high computation times. A simple and well-known feature-based recognition method is the Bag Of Words (BOW) approach [13].

Bag of Words Model The bag of words approach [13] (bag of key points or bag of features) is a basic method to solve the class recognition problem [64]. The processing steps of the BOW approach can be summarized as follows:

- **Keypoint detection and feature extraction:** In a first step features (*e.g.* SIFT [49]) are extracted at keypoints in the image. Such features should be invariant against transformations like translation, scale, rotation and changing illuminations. The extracted features are also called *visual words*.

- **Visual vocabulary construction:** The extracted features are quantized in order to get a distribution (histogram) over the visual words. Therefore, clusters of the visual words (*codewords*) are calculated which are representative for several features. The resulting clusters (codewords) are used to describe the entire image with visual words.

- **Classification:** Based on the feature histogram distribution, a classification algorithm (*e.g.* a support vector machine) is used to create a decision surface. Images are recognized by classifying the feature distribution of the query image.

The idea of the BOW approach is shown in Figure 2.15. A more detailed study can be found in [73], where the BOW model is compared based on different feature descriptors and classification algorithms.

The BOW approach does not incorporate the spatial relation between the estimated features. Hence, *part-based models* are introduced which take spatial relations into account.

Part-based Models The idea of describing objects by their geometric parts, and the relations between those parts, goes back to the idea of the pictorial structures presented in [22]. Pictorial structures consider an object as an deformable version of a template [64]. The major idea of the pictorial structure approach is to represent a model as a collection of parts with connections between pairs of parts. Each part encodes local visual properties of the object. The deformable configuration is characterized by so called *springs*. Figure 2.16a illustrates the idea of pictorial structures. A typical way to describe such a model is the usage of an undirected graph $G =$

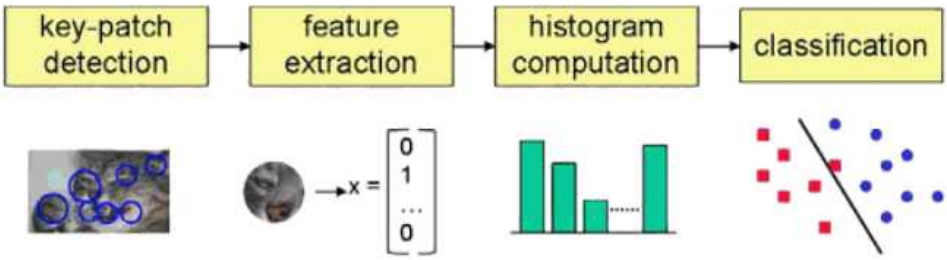


Figure 2.15: Bag of Words approach. Features are extracted at keypoints and the quantized features are used to create a histogram of visual words. Classification algorithms are used to train and classify feature histograms. (Image taken from [64])

(V, E). The vertices $V = \{v_1, ..., v_n\}$ represent the n parts of the query object. Each edge $(v_i, v_j) \in E$ defines a connected part of the model. The problem of matching a pictorial structure to an image can be described using energy functions and optimization methods [22].

(a) (b)

Figure 2.16: Object recognition using part-based models. (a) Pictorial structures decompose objects into parts and the spatial relations between those parts. (b) Deformable *filters* define relevant parts of a model using gradient features and the position of the features inside a bounding box. (Images taken from [22] and [21])

Recent state-of-the-art object recognition algorithms are based on the previous mentioned ideas of feature-based recognition and part-based model. The Deformable Part Model (DPM) object detector presented by Felzenszwalb *et al.* [21] is based on Histogram of Oriented Gradients (HOG) features and spatial models between these HOG features (cf. Figure 2.16a). The DPM algorithm provides the bounding box of the object in question. The algorithm achieves state-of-the-art result in the PASCAL Visual Object Challenge (VOC)[1]. For more details on the DPM detector compare Section 3.3 in the methodology chapter.

[1] `http://pascallin.ecs.soton.ac.uk/challenges/VOC` (last access: 17.04.2015)

2.2.2 3D Object Detection from Range Images

The detection of 3D models using depth/range images is a well-studied area in computer vision. An introduction and an extensive overview can be found in [50]. In this thesis we introduce a detection approach which (1) recognizes the type/class of an object in question and (2) determines the pose (orientation) of the corresponding object. Hence, in the following we briefly discuss 3D object detection methods based on point clouds which are used for both – object classification as well as pose estimation.

The challenge of 3D object detection and recognition in point clouds (range images) can be described as aligning single point cloud views into a global point cloud [50]. This task is also known as (3D-) registration, where the goal is to find the relative position and orientation of a partial point cloud in a global model. 3D object detection is applied in fields like robotics, medical imaging, computer graphics or virtual augmented reality [50]. In the following subsections we describe approaches based on (1) points cloud registration using the ICP algorithm and (2) shape based retrieval using point cloud descriptors. It is also possible to combine these methods. In [3] a point cloud descriptor is used to recognize the type and the pose of object candidates and in a post-processing step the ICP algorithm is applied in order to refine the 6 DOF pose results.

3D Object Detection using ICP Registration

The ICP algorithm [74] looks for the transformation (rotation and translation) between corresponding 3D points of two point clouds by minimizing the mean squared error between there points.

Objects are typically detected by matching a source point cloud against a trained database using the ICP algorithm. The source model which delivers the lowest registration error is then returned. Methods in which variants of the ICP algorithm are used to detect objects in points clouds are presented in [26] and [1]. The latter deals with the generation of semantic maps for mobile navigation. In [26] cars are detected in outdoor scenes using a high resolution laser scanner. Both methods are based on synthetically created point clouds which are generated in an offline training stage. Therefore, all faces of the trained CAD models are uniformly filled with points. Since point clouds in real worlds scenarios are seen from a single viewpoint, partial 2.5D point clouds are more suitable [56].

A bottleneck of ICP registration methods is that similar target and source point clouds (shape and size) are required [26,43]. Consequently, large training databases are a challenging problem. The method of [26] requires approximately 90 model to detect one specific object lass (*e.g. BMW Z3*).

ICP approaches theoretically solve the full 6 DOF problem [2]. Nevertheless, the most ICP approaches require an initial pose estimation [35], or have the restriction that objects need to be placed perpendicular on the ground plane. In order to detect objects of unknown size, in [26] an additional scale factor is introduced. This problem can also be solved by using different scaled versions of the 3D models during the training, which results in time-consuming training stages.

3D Object Detection using Point Cloud Descriptors

In contrast to ICP registration methods, it is also possible to use shape analysis methods to recognize objects and determine their poses. The general idea behind shape-based retrieval is to compare objects by measuring the similarity between their shape signatures (also known as *point clouds descriptors*) [56]. Signatures are computational representations of objects, which should be fast to compute and easy to compare, while still discriminating between similar and dissimilar point clouds. Shape-based retrieval is introduces in [53].

Widely used methods for describing full 3D models are spin-images [40] or spherical harmonics [41]. Spin-images and spherical harmonics are useful for full 3D models and densely sampled point-based representations. Therefore, such descriptors are not designed to handle partial and cluttered point clouds [69]. Between those two extrema (2D image / 3D point clouds), a numerous of methods (*e.g.* [3, 57, 69]) which are based on partial 2.5D point clouds have been introduced.

Partial 2.5D Point Clouds Descriptors Point cloud descriptors are widely used in computer vision applications for tasks like point clouds registration, object recognition, categorization and shape retrieval. An extensive review and an comparison of point clouds descriptors can be found in [2].

In order to use global 2.5D point cloud descriptors, point cloud candidates (pre-detected objects) are required. Therefore, clustering approaches and plane detection methods are used in several works (*e.g.* [3, 26, 56]). Plane-fitting methods are required in order to detect and remove points which belong to the ground plane. This helps to extract object candidates. In [2, 3, 56] objects are supposed to be placed on a table. Consequently, in those works a simple plane-fitting methods is used. In more complex scenarios, an additional layout estimation approach is desirable. After object candidates have been extracted from the entire point cloud, the recognition task is solved by determining a global descriptors for each object candidate.

The Point Feature Histogram (PFH) descriptor presented by Rusu *et al.* [55] is one of the most used point cloud descriptors [2]. The idea of the PFH descriptor is to analyse the geometric relations between a point and all neighbouring points within a sphere. The PFH descriptor works as follows. First, normal vectors are estimated at each surface point. Second, based on this normal vectors, a Cartesian coordinate system is created at each surface point (cf. Figure 2.17). Third, the descriptor calculates all point pairs and estimates angular features between the coordinate systems. These values and the Euclidean distance between the query points are binned into a multidimensional histogram in order to create a comparable shape signature.

The bottleneck of the method is the computational complexity. For a point cloud with n points, the computational complexity is $\mathcal{O}(nk^2)$ where k defines the neighbouring points off a query point. The influence diagram of the PFH descriptor for one query point is shown in Figure 2.18a. Because of the computational complexity of the FPH descriptor, the Fast Point Feature Histograms (FPFH) is introduced in [56]. By removing redundant links between point pairs, the computational complexity is reduced to $\mathcal{O}(nk^2)$. The influence diagram of the FPFH descriptor is illustrated in 2.18b.

The PFH and the FPFH descriptor are successfully used for classification tasks, but both descriptors are invariant against rotations and thus invariant to the object's pose [57]. Therefore,

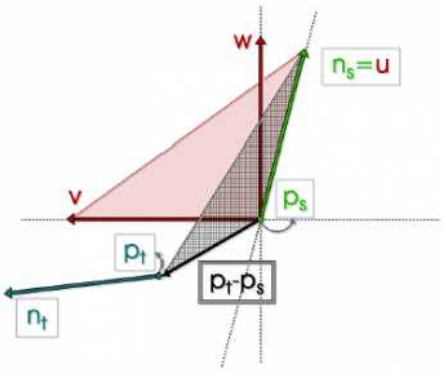
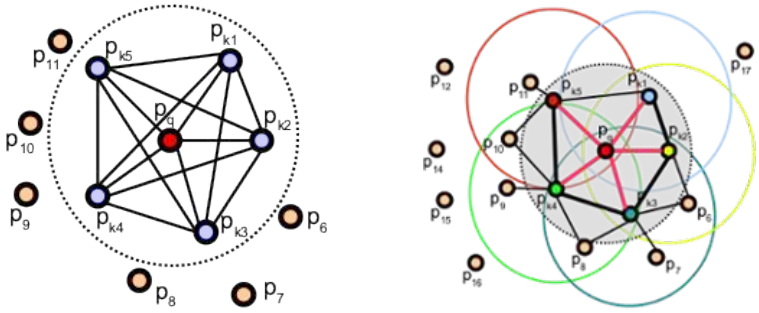


Figure 2.17: A Darboux coordinate frame $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ located at a query point $\mathbf{p}_s$. (Image taken from [56])



(a) (b)

Figure 2.18: Influence diagram. (a) The FPH descriptor calculates angular features between all possible points pairs referenced by a query point $\mathbf{p}_q$. (b) The FPFH descriptor first estimates angular features for a query point $\mathbf{p}_q$ and its direct neighbours (red lines). Secondly, the direct neighbours (points inside the grey circle) are linked to their own neighbours (black lines) and a re-weighting schema is used to determine the final signature (Images taken from [2] and [56])

based on the real-time ability of the FPFH descriptor, the Viewpoint Feature Histogram (VFH) descriptor is introduced by Rusu *et al.* [57]. This descriptor provides an additional viewpoint-dependent component using statistics between the viewpoint of the sensor and the surface normals. Moreover, the descriptor is invariant to scale transformations. Since the descriptor is invariant to rotations around the camera axis, the approach does not solve the full 6 DOF problem [3]. An further limitation of the original presented VFH approach is that the descriptors are trained online. Therefore, training objects are spun on a turntable, and partial views are captured with a stereo sensor. Such an online training is time-consuming and costly, especially if all possible poses are captured. A more detail introduction to the VFH descriptor can be found in the methodology of this thesis (see Section 3.3).

The VFH descriptor has problems dealing with missing parts, caused by specular surfaces, segmentation artefacts or occlusion [3]. Because of missing parts, the estimated centroid of a

cluster differs from the original centroid. Thus, the descriptor does not match correctly (*i.e.* a wrong model is found) with the corresponding descriptor of a trained model. To overcome this limitation, Aldoma *et al.* [3] introduce the Clustered Viewpoint Feature Histogram (CVFH) descriptor. The idea is to estimate N VFH descriptors for all stable regions of a point cloud instead of one global descriptor for the entire cloud. Figure 2.19 shows stable regions of two sample point clouds. After calculating the VFH descriptors for all regions, the entire signature is given by concatenating all histograms. Stable regions of a point cloud are estimated using a region growing algorithm which clusters neighbouring normal vectors by examining their positions and directions. (see [3] for details).

(a) (b)

Figure 2.19: Clustered Viewpoint Feature Histogram descriptor. (a) Three stable regions are estimated from a sample point cloud. (b) Similar point cloud model found in a trained database. (Images taken from [57])

Furthermore, the CVFH descriptor introduces a pose estimation method which solves the full 6 DOF pose problem by calculating an additional histogram – the *camera roll histogram*. The camera roll histogram projects all normal vectors on a plane which is normal to the camera [2]. Next, the angles between the up-vector of the camera and the projected normal vectors are calculated and binned into a *new* histogram with a resolution of four degrees. Compared to the VFH descriptor, the additional roll histogram solves the last degree of freedom (rotation around the camera axis). Since the full 6 DOF problem is solved, the CVFH method can be used in robotic applications to detect and grab objects (cf. Figure 2.14). The CVFH descriptor is not scale-invariant, which makes it possible to distinguish between objects of different size but identical shape. Moreover, the CVFH approach is successfully tested with an offline training approach which is based on the usage of synthetically CAD models downloaded from the a web database.

While in [2, 3, 57] point cloud descriptors are used to classify objects which are placed on a table (*e.g.* mugs, bottles, toys), the method proposed by Wohlkinger *et al.* [69] introduces a point clouds descriptor – the Spherical Harmonics (SH) descriptor – to classify objects which are found in a home environment (*e.g.* chairs). The SH descriptors is invariant to affine transformations and is calculated using a voxel representation of the input point cloud. Since shape histograms are used to represent the descriptor, it is possible to apply standard classification methods.

The major problem of all presented point cloud descriptors is that the methods have problems

to handle missing parts in the point clouds [2]. The VFH and the CVFH descriptor are robust to noise and clutter. Nevertheless, a considerable amount of missing points and holes in the clouds (*e.g.* caused by occlusion) are still a challenging problem.

We have presented two types of 3D object detection techniques – ICP-based registration methods and detection approaches which rely on point cloud descriptors. In [3] the strengths of both approaches is combined. First, a descriptor is used to recognize objects (shape and type) placed on a table. Second, a ICP algorithm is applied in a post-processing step in order to refine the 6 DOF pose. Another common method is to re-rank the results obtained from the nearest neighbour search (classification of the point clouds descriptor), using the error metric provided by the ICP algorithm [2].

2.2.3 3D Object Detection from RGBD Images

In the last years several works have been presented which use multimodal RGBD data to solve the object recognition task [35, 44, 63]. Feature-based methods (*e.g.* DPM detector) are based on extracted keypoints and therefore have problems to handle texture-less objects where the appearance is mainly given by the object contour. Hence, methods which combine depth and images cues are introduced.

The template-based recognition method proposed by Hinterstoißer *et al.* [34, 35] uses combined image and depth cues to handle texture-less objects. First, discriminative image gradients are calculated using the contours found in the color image. Second, a depth sensor is used to create surface normals. Normals are mainly found on the object interior. Third, the 2D image gradients and the 3D surface normals are quantized and binned into a histogram in order to create robust model representations. Finally, models are detected and classified using a similarity measure in combination with a classification algorithm. Figure 2.20 illustrates the idea of describing an object by combining 2D image gradient and 3D surface normals.

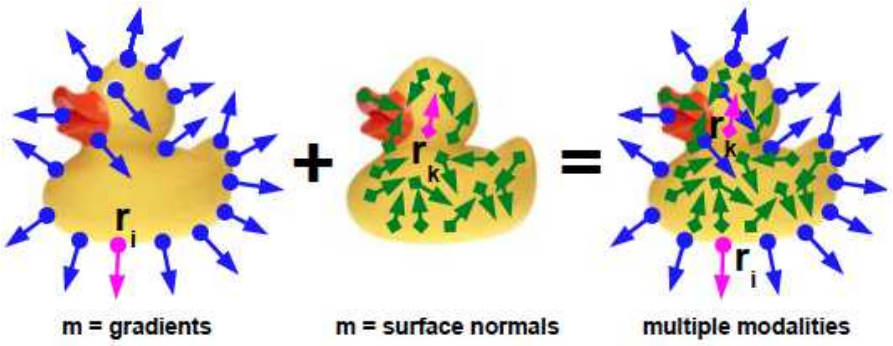
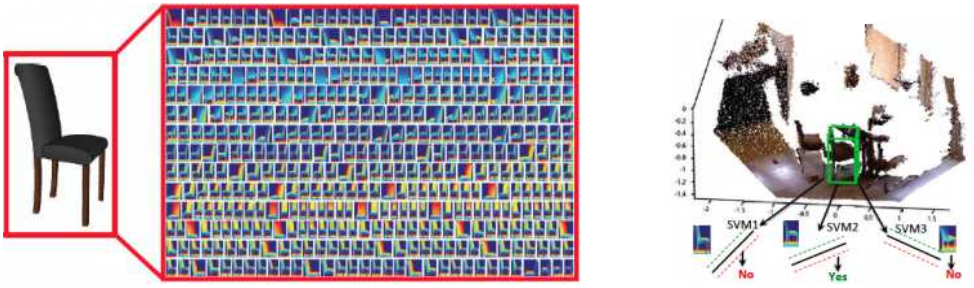


Figure 2.20: Model with different modalities. (a) Discriminative image gradients are extracted using contours. (b) Surface normals are calculated from intensity images. (c) A model is described by combining the color and depth cues. (Image taken from [35])

The approach allows the detection of objects in cluttered scenes in real-time. In [35] the framework is improved by introducing a offline training method which is based on synthetic

CAD data. Furthermore, a method to estimate the 6 DOF pose is presented. For the detections of the objects and also for the training of the templates a low-budget sensor (Microsoft Kinect) is used.

Song *et al*. [61] propose the *Sliding Shapes Detector* which is used to detect objects (furnitures) based on RGBD data. The view-dependent descriptor is created from 3D point clouds and is based on several features. In a first step, the entire point cloud is devided into cubic cells. Second, multidimensional features which encode shape, orientation and the distance to the camera are extracted from each cell. Third, a Support Vector Machine (SVM) is used to classify the features. In detail, the feature descriptor consists of density information, 3D normals, shape information and TSDF features. For the training stage the approach uses depth images which show training objects from hundreds of different view angles, locations and scales (cf. Figure 2.21b). Therefore, for each object an Exemplar-SVM is created. Objects are detected by applying a sliding window approach (cf. Figure 2.21a). Each possible bounding box is classified using an ensemble of the Exemplar-SVMs.



(a) (b)

Figure 2.21: Sliding Shapes Detector. (a) A SVM is trained showing sample objects (in range images) from hundreds of viewpoints. (b) Objects are detected by shifting a sliding window in the 3D space. Each possible bounding box is classified using an ensemble of SVMs. (Images taken from [61])

The average precision of the sliding shape detector achieves an improvement of about 1.7 times compared to the DPM detector [21]. An disadvantage of the original proposed approach is that objects are supposed to be placed perpendicular on the ground plane. Although the thereby reduced number of viewpoints, the training of one model is still time-consuming. In [61] the training of one class take about 4-8 hours. Furthermore, compared to descriptors like VFH and CVFH, real-time object detection is more challenging because of the high number of possible sliding windows.

2.3 Discussion & Implications for the Proposed Methodology

In the following we give a brief discussion of the proposed related work and give suggestions for the methodology used in this work.

2.3.1 3D Layout Estimation

We have presented frameworks for 3D layout estimation and reconstruction. While the objective of the former is to estimate geometric classes (*walls*, *floor*, *ceiling* and *clutter*) from photometric images, reconstruction methods focus on the generation of highly detailed representations using structured light sensors. Recently, multimodal RGBD data is used to combine the strengths of image-based and depth-based approaches.

The major advantage of monocular methods is that the entire scene can be represented with a parameterized 3D box. Consequently, scenes in which more than four walls are visible, or the underlying floor plan has a concave structure, are not represented accurately. The advantage of representing a scene by a single 3D box is that it is possible to easily estimate semantic attributes. Point-based reconstruction methods on the other side are able to handle random scene geometries. The bottleneck of such approaches is the complex computation of the models in terms of processing time and storage demands and that the models do not provide semantic attributes.

Implications for the 3D Layout Estimation Pipeline In this thesis we use a RGBD layout parsing method [65] (see Section 3.3) in order to estimate initial scene layouts. Based on this labeling, we detect 3D planes using a RANSAC plane-fitting method. We borrow ideas from [39] to fuse several frames over the time. Since we are interested in simplified 3D models, we adapt the simplification method used in [66]. Similar to [24], our 3D layouts consist of a combination of multiple 3D planes. Moreover, we include semantic information using a multimodal object detection an pose estimation approach which is an improvement compared to pixel-wise labeling methods.

2.3.2 2D/3D Object Detection and Pose Estimation

For both – 2D as well as 3D object detection and recognition – a broad spectrum of methods is available. Multimodal RGBD data enable the combination of the advantages of color and depth approaches.

Part-based methods (*e.g.* DPM object detector) are able to handle the instance recognition problem which is the most challenging recognition task. Therefore, annotated images are required in order to train a classification algorithm. Hence, a bottleneck of such methods is that the training is costly. Commonly, instance recognition methods return 2D or 3D bounding boxes of the object in question.

3D object detection can be solved using registration algorithms (*e.g.* ICP) or shape-based retrieval methods. The bottleneck of registration methods is that similar point clouds (source and target model) are required. The goal of 3D retrieval is to find similar point clouds in a trained database. Therefore, similarity measures are used to compare point cloud signatures. Since a numerous of different descriptors are available (*e.g.* VFH, CVFH, PFH, SH), it is important to define the requirements of the recognition framework in advance (*e.g.* scale invariance, degrees of freedom).

Objects in real world scenarios are 2.5 dimensional because they are seen from a specific viewpoint. Thus, it is necessary to train viewpoint-dependent point clouds in order to apply

shape descriptors. A bottleneck of recent methods is that the frameworks are trained online with real objects captured using a depth sensor / structured light sensor. The major problem of 3D shape retrieval is that missing points directly lead to inaccurate detection and pose results.

Implications for the 3D Object Recognition Pipeline Since in this thesis RGBD video sequences are available, we introduce a multimodal object detection and pose estimation method. In an initial step we apply the DPM object detector [21] in order to find object candidates for the classification and pose estimation task. Since the 2D detector searches for objects visible in the scene (*e.g.* a couch), we are able to reduce the 3D classification problem to one specific class. The 2D detector provides bounding boxes which help us to extract the relevant object points from the entire scene point cloud. For the point cloud classification and pose estimation approach we apply the VFH descriptor [57] because (1) the descriptor is scale-invariant and we do not know the scaling of our object in advance, (2) the VFH signature distinguishes between different viewpoints and (3) the descriptor encodes the shape of the objects which allows us to handle the high intra-class variability of furnitures. Furthermore, we borrow ideas from [57] to train our classification method offline with synthetic CAD models from the web.

CHAPTER 3

Methodology

This chapter outlines the methodology used in this thesis. We present how 3D layouts are estimated from multimodal RGBD videos. Since the objective of this thesis is to estimate semantically meaningful 3D layouts, we present an object recognition pipeline which detects objects in the fused point clouds and replaces them by synthetic CAD models. We introduce temporal optimization methods for both – the 3D layout estimation as well as the object classification pipeline.

The whole workflow for the generation of a 3D layout from a 3D point cloud as well as the object detection and the pose estimation results for the objects visible in the scene is displayed in Figure 3.1.

3.1 Implementation Details

The proposed framework relies on RGBD video sequences which are captured using a *Microsoft Kinect*[1] sensor. The Kinect sensor provides synchronized color and intensity images (RGB

[1] `http://www.microsoft.com/en-us/kinectforwindows/` (last access: 17.04.2015)

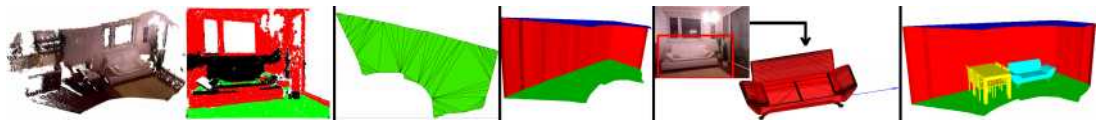


Figure 3.1: Workflow of the proposed simplification algorithm. (a) Fused input point cloud coming from multiple RGBD frames. (b) Merged 3D plane candidates (*wall*, *ground plane*). (c) Floor plan estimation: Projection of 3D points on the floor and triangulation of the 3D points within the *groud plane* segments. (d) Extrusion of *wall* and *ceiling*. (e) Object detection, pose estimation and optimization. (f) Final 3D layout. See text for details.

image / depth image) at the same time. The information of both images can be combined in order to generate colored 3D point clouds [43]. The necessary depth information is captured by an infrared sensor. Both images have a resolution of 640x480 pixels. The RGBD video sequences, which are processed by our framework, are created by manually storing a synchronized RGB and depth image pair from the data streams every second.

Since this thesis deals which 3D and point-based data, we use the *Point Cloud Library*[2] (PCL) which is an open source and standalone library for 2D/3D image processing. The Point Cloud Library (PCL) library contains basic functions for processing point clouds and therefore provides algorithms for *e.g.* plane fitting, filters, model fitting, registration, feature descriptors, segmentation and normal vector estimation. The functions are implemented in C++.

To capture images from the Kinect we use the *OpenNI Grabber*[3] interface which is also provided by the PCL library. This interface requests data streams from OpenNI compatible cameras like the Microsoft Kinect, and returns synchronized color and intensity images. Since the two images captured by the Kinect sensor do not overlap perfectly due to slightly different viewpoints, the two images are registered by OpenNI in a first step [43].

Moreover, the PCL library provides a Visualization Toolkit (VTK) for the visualization of point clouds. The 3D layouts and the corresponding mesh representations are illustrated with *matVTK*[4], which is a VTK based 3D visualization extension for MATLAB.

All experiments have been performed on a standard MacBook Pro with a 2.5 GHz Intel i5 CPU, 4 GB RAM and a Intel HD Graphics Card 4000 with 512 MB. The whole framework is implemented in MATLAB (R2013a). The core functions are implemented in C++.

3.2 3D Layout Estimation and Scene Simplification

Figure 3.2 illustrates all processing steps for the 3D layout estimation which can be summarized as follows:

1. **3D Point Cloud Estimation:** Point clouds coming from multiple shots are fused in order to estimate a dense point cloud representation of the entire video sequence. Transformation matrices between overlapping frames are estimated and stored for further use.

2. **Plane Fitting / Wall Detection:** Layout segments (*walls* and *ground plane*) are estimated in the RGBD frames using a labeling algorithm. The detected segments are refined using a plane-fitting algorithm which removes redundant points. Each initial detected wall is further represented as a straight 2D line.

3. **Floor Plan Estimation:** A rough floor plan is generated by applying an alpha shape approach on the scene points projected onto the floor. An exact floor plan is obtained by trimming the rough floor plane with the 2D lines obtained from the corresponding wall segments.

[2] `http://pointclouds.org` (last access: 17.04.2015)

[3] `http://pointclouds.org/documentation/tutorials/openni_grabber.php` (last access: 17.04.2015)

[4] `http://www.cir.meduniwien.ac.at/team/birngruber/matvtk/` (last access: 17.04.2015)

4. **Triangulation / Extrusion:** The boundary points of the floor plan are traced and triangulated. The resulting vertices are used for extruding the wall segments and the ceiling.

5. **Temporal Fusion:** The simplified layouts coming from single frames are fused in order to expand the floor plan. Consequently, a fused 3D layout is generated by extruding the updated floor plan.

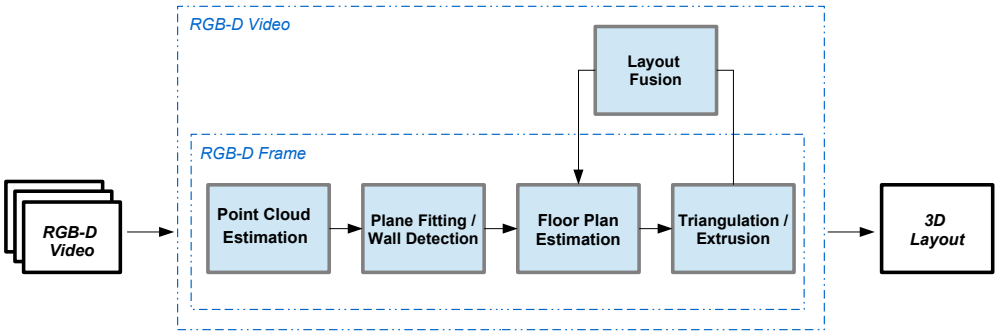


Figure 3.2: Layout estimation pipeline. Single 3D layouts are estimated by processing RGBD frames. The fused layout is obtained by merging multiple 3D layouts over the time. (See text for details.)

The whole framework is presented by processing a running example (RGBD video sequence). Sample frames of the captured video used are illustrated in Figure 3.3.

Figure 3.3: A RGBD video forms the basis for the layout estimation and object classification framework.

(a) (b)

Figure 3.4: Sample frame of the video sequence. (a) Raw RGB image. (b) Depth image.

3.2.1 Preprocessing

A sample RGB and the corresponding intensity image is illustrated in Figure 3.4. Pixels in the intensity image which are close to the sensor are colored with dark values. Consequently, bright pixels denote points which are further away. Black colored pixels indicate missing depth values. The Microsoft Kinect depth sensor ranges approximately from 80cm up to a maximum of 4m [43].

For the estimation of the 3D point cloud, the camera's intrinsic parameters are required [33]. The parameters are obtained from a MATLAB based *Calibration Toolbox*[5] (see [33] for more details). According to the pinhole model the camera matrix $\mathtt{K}_c$, describing the intrinsics, is defined by

$$\mathtt{K}_c = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.1}$$

where f is the focal length and c_x/c_y is the principal point.

The depth data obtained from the Kinect sensor (using the OpenNI framework) is a 2D image which specifies a disparity unit in in meters for each pixel of the depth image. Based on the pinhole camera model, the transformation of a depth point to a 3D point and the estimation of the entire 3D point cloud is described by Algorithm 3.1. Since organized point clouds are captured, for each pixel point of the image, an additional 3D point is available. In order to add a color information to each 3D point, the 3D points are transformed to the coordinate frame of the color image using a rigid transformation (rotation and translation) obtained from external calibration parameters.

Figure 3.5 shows the estimated point cloud of the sample scene from three different viewpoints. The position of the sensor is located at the coordinate system's origin. The visualization shows the already mentioned problems of the point-based representation:

[5] `http://www.ee.oulu.fi/~dherrera/kinect/` (last access: 17.04.2015)

Input : Depth image I_{depth} of size $H \times W$, Camera parameters f_x, f_y, c_x, c_y
Output: 3D point cloud C_{xyz}

for $j \leftarrow 1$ **to** H **do**
  for $k \leftarrow 1$ **to** W **do**
    `// z in meters`
    `z` $\leftarrow$ `getDepthValue(`$I_{depth}[j,k]$`)` ;
    `// X,Y,Z coordinates`
    $C_{xyz}[i].x \leftarrow z * \frac{(j - c_x)}{f_x}$;
    $C_{xyz}[i].y \leftarrow z * \frac{(k - c_y)}{f_y}$;
    $C_{xyz}[i].z \leftarrow z$;
    $i++$;
  end
end

Algorithm 3.1: Transformation of a 3D image points to a 3D points.

- Missing depth values lead to clutter and holes in the point cloud.
- Even a single shot consists of a high number of points (187 961 valid points in the sample scene).

Figure 3.5: Cluttered 3D point cloud of the running example viewed from three viewpoints.

3.2.2 Point Cloud Registration

Having a sequence of N overlapping RGBD frames, the proposed framework estimates a global point cloud model by fusing multiple point clouds. For the registration we combine information coming from the RGB and the depth images. In a first step, SIFT keypoints [49] are extracted from the RGB frames. Secondly, the SIFT features of consecutive frames are matched using a nearest neighbour distance. Therefore, we use the ratio-test [49] which calculates the ratio between the closest-distance to second-closest feature distance. The ratio-test excludes matches

with a ratio bigger than a given threshold T_{ratio} (in our experiments $T_{ratio} = 0.36$). The distance between two features is calculated using the Euclidean metric.

After poor matches have been removed, the remaining features with valid depth values are used to obtain the relative transformation between the point clouds of consecutive frames. Therefore, a three-point algorithm [23] is applied on the 3D points in order to estimate a rigid transformation. Having a set of at least three point pairs $(\mathbf{f}, \mathbf{f}')$ of the overlapping point clouds, the rigid transformation is obtained from the rotation matrix $\mathtt{R}$ and the translation vector $\mathbf{t}$, which maps the points $\mathbf{f}$ onto the points $\mathbf{f}'$ by $\mathbf{f}' = \mathtt{R}\mathbf{f} + \mathbf{t}$. The matrices are calculated by minimizing the error function

$$E = \sum_{k=1}^{M} |\mathbf{f}'_k - \mathtt{R}\mathbf{f}_k + \mathbf{t}|^2, \tag{3.2}$$

where M denotes the number of matching points.

For a video sequence of N frames the global point cloud is estimated by incrementally stitching the overlapping point clouds together. A sample for a fused point cloud is shown in Figure 3.6. The fused point cloud consist of 3 543 200 points and requires 163.60 MB of storage on the hard disk. Consequently, it is time-consuming to load ($\sim$ 55 seconds) and process the point cloud, even on a desktop PC.

Figure 3.6: Multiple point clouds are stitched together to generate a dense point cloud of the RGBD video.

After the estimation of a global point cloud model and the transformation matrices between consecutive frames, the next step of the pipeline (cf. Figure 3.2) deals with the generation of a simplified 3D layout for the fused point cloud.

3.2.3 Wall Detection and Plane Fitting

Having a 3D point cloud of the input scene, the objective of the following step is the detection of planar regions in the RGBD frames. Therefore, a wall detection algorithm followed by a plane-fitting approach is applied.

Initial Wall Detection Following the idea of [38] segments are found by clustering 3D point normals. The algorithm shows limitations concerning the runtime and it is also difficult to distinguish between wall segments and dominant objects in the scene (*e.g.* a cupboard). The sample scenes in this thesis are assumed to be taken in indoor scenes (*e.g.* living room, hotel apartment, dining room), where man-made geometries and rectilinear structures are available. Therefore we adapt the algorithm proposed in [65], which is summarized in the following paragraph.

- **Segmentation:** In a first step the color image is segmented by using a superpixel segmentation method. Edges in the RGB image are detected with a Canny edge detection approach. Next, a triangulation method is used to obtain a tessellation of the color image (edgels). HSV color features are then used to merge adjacent edgels.

- **Planar Segment Estimation:** The RGB image segmentation is used as a prior to suggest planar patches within each segment. Therefore, the corresponding depth values are taken into account. The estimated planar segments are further merged using a greedy algorithm which combines coplanar segments.

- **Rectilinear Structure Generation:** Under the assumption that the gravity vector is aligned with the vertical axis of the scene, the floor is detected by searching for a planar region near the bottom of the image with a normal vector which is approximately vertical. Wall candidate segments are obtained by searching for segments with a normal vector perpendicular to the normal vector of the floor. Based on the floor's normal vector, the dominant rectilinear structure (scene coordinate system) is defined for the entire scene.

- **Wall Candidate Identification:** Having the rectilinear structure of the scene enables the estimation of extended planar segments (called wall candidates). Wall candidates are determined by examining the angle between the normal vector of the wall segments and the horizontal/vertical vector of the scene coordinate system. Segments which are approximately parallel to the floor of the scene, and further have a horizontal or vertical extent, are fused.

- **Divide Image into Intervals:** The algorithm divides the RGB image into a sequence of intervals. Therefore, the extends (endpoints) of the extracted wall segments and the intersection points between perpendicular walls are calculated. Image intervals are obtained by sorting those endpoints.

- **Layout Extraction:** The final layout of the scene is obtained by a labeling procedure which labels each interval concerning the underlying wall segment. First, for each pixel in the RGB image the optimal wall label is calculated. The calculated walls are projected

onto the RGBD frame and each pixel gets the label information of the closest wall segment. A quadrilateral which describes the visible walls in the scene is defined from the endpoints of the intervals and the floor. All pixels within the rectangle are considered to estimate the number of inlier pixels, in which a high number denotes good choices. Since this labeling approach is not sufficient, a global optimization method which enforces depth continuity is used to get smooth and rectilinear layouts.

Figure 3.7 shows the resulting output of the initial labeling algorithm in which each pixel is assigned with an additional label information.

(a) (b)

Figure 3.7: Initial labeling. (a) RGB input image. (b) Detected ground plane and wall segments obtained by the labeling algorithm.

Plane Fitting First, single point clouds are extracted for each detected wall candidate and the ground plane. The layout candidates are estimated with the previous presented superpixel labeling algorithm. A statistical outlier removal filter [55] is applied on the point clouds in order to remove noise and clutter. Secondly, a plane is fitted into each point cloud in order to remove outliers and objects (*e.g.* a couch). A plane in the 3D space can be defined by the Cartesian equation

$$a\mathbf{x} + b\mathbf{y} + c\mathbf{z} + d = 0. \tag{3.3}$$

The iterative RANSAC plane fitting approach [64] is applied on the points to find the parameters of the plane equation. The RANSAC algorithm selects three random points and estimates the plane equation. The remaining points are used to calculate the number of hypothetical inliers using a distance threshold. Inliers are points which lie on the estimated plane or are situated in the immediate vicinity. We use a threshold of 3cm to check whether a point lies on the plane. The final plane is described by the configuration delivering the highest number of inliers. The algorithm determines after the maximum number of iterations is reached.

After this step, each correctly detected wall and the floor is defined by a plane equation and the corresponding normal vector. Planes with less than 1000 points are considered as incorrect segments and are therefore excluded from the further layout estimation tasks. Figure 3.8 shows the detected planes for our running example. In the example four planes (three walls and the

ground plane) are detected. The example shows that objects (*e.g.* the couch) are filtered out by the plane-fitting algorithm.

(a) (b)

Figure 3.8: (a) Raw 3D point cloud. (b) Walls detected by the super-pixel labeling / plane-fitting algorithm.

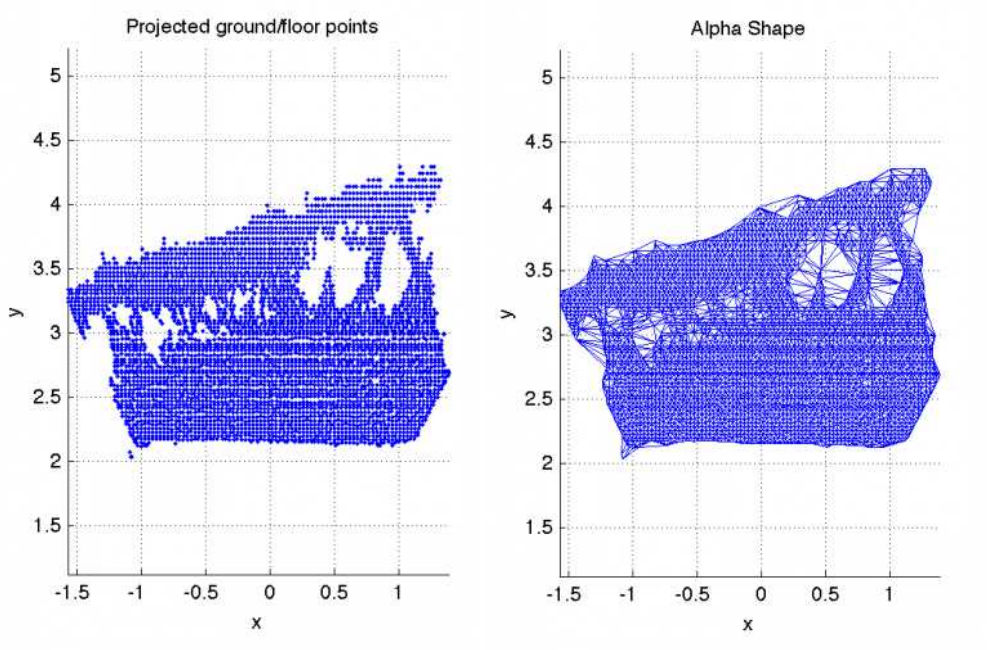
3.2.4 Floor Plan Estimation

The following summarizes the tasks to estimate the mesh-based floor plan:

- **Alpha Shape**: All scene points are projected on the floor and triangulated using an Alpha shape approach. The Alpha shape forms a rough floor plan triangulation.
- **Line Fitting**: Each wall is represented by a single line onto the ground plane using a line fitting approach.
- **Binarization**: A binary mask of the floor plan is estimated by combining the information coming from the Alpha shape and the line fitting method.

Alpha Shape In a first step a triangle mesh of the scene points is required which delivers a rough floor plan. Thus, all point of the scene are projected on the ground plane. In order to reduce the computational complexity of the following triangulation, the point cloud is down-sampled with a voxelized grid approach. A voxelgrid filter approximates all points within a voxel with the centroid of the voxel. We use the voxelgrid filter provided by the PCL library with a leaf size (voxel size) of 3cm. The resulting 2D point cloud for our example is illustrated in Figure 3.9a.

Next, the downsampled points are triangulated. Our first idea was to create the convex hull of all points. The problem of a convex hull method is that concave structures are not considered and outlier points (*e.g.* noise) directly lead to incorrect structures. Hence, we use an Alpha shape approach [17] to generate a rough floor plan. In a first step all 2D points are triangulated using a Delaunay triangulation approach [62]. Since such a triangulation disregards concave structures, all triangles with a circumradius greater than a certain radius are removed. In our experiments we use a radius of 6cm. Consequently, this approach handles concave structures, but ignores outliers (*e.g.* missing points which lead to holes) at the same time. The resulting rough floor plan is shown in 3.9b.



(a) (b)

Figure 3.9: Rough floor plan estimation. (a) Projected points of the entire scene. (b) The Alpha shape method takes the underlying shape into account and closes holes at the same time.

Line Fitting The estimated mesh-based floor plan does not provide a rectilinear structure. Hence, we want to include the results from the 3D plane fitting method. The points of each detected plane are first filtered. After line fitting (see Figure 3.10), the points are projected on the ground plane and a line is fitted into the projected points.

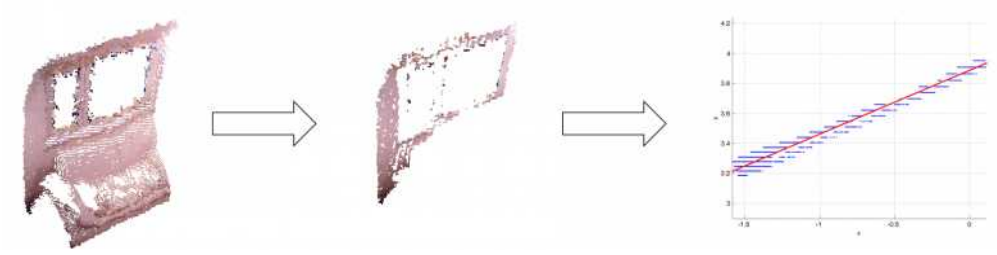


Figure 3.10: Line fitting. Points of the initial detected wall segment (left) which belong to the fitted plane (middle) are projected on the ground plane (right). Each wall is represented by a straight line.

Optimization In order to obtain a rectilinear floor map, we combine the two outcomes from the Alpha shape and the fitted lines. Each triangle of the Alpha shape is transformed into a

closed polygon by filling all polygons. Morphological operations [15] are used to fill holes in the resulting floor plan (*e.g.* caused by occlusions). The projected wall segments are used for trimming the floor map. Figure 3.11 shows an example for obtaining the map. The 2D lines are created with an Bresenham Line-Drawing algorithm [6].

(a) (b) (c)

Figure 3.11: Floor plan estimation. (a) Rough floor plan obtained by filling each triangle of the Alpha shape. (b) Each detected wall segment is transformed to a binary image (red lines). (c) The trimmed floor plan supports the underlying rectilinear geometry of the scene.

3.2.5 Floor Plan Triangulation and Wall Extrusion

Having an exact binary mask of the floor plan enables the generation of a 3D layout. We estimate the 3D layout for the scene by performing the following three steps:

- **Floor Plan Triangulation:** The binary mask of the floor plan is converted into a 2D mesh.
- **Floor Plan Simplification:** The 2D mesh of the floor plan is simplified.
- **Wall Extrusion:** Walls are created by extruding boundary wall segments.

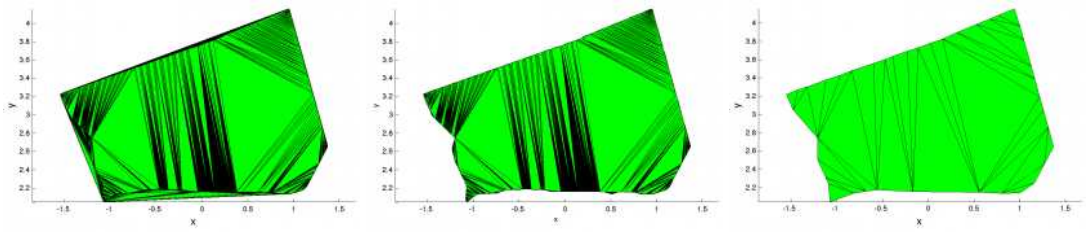
Floor Plane Triangulation The exterior boundary of the binary mask obtained in the previous step describes the dimensions of the underlying scene. The boundary is traced by applying the Moor-Neighbour tracing algorithm [54]. The coordinates of the resulting polygon course are transform back into a 3D point cloud ($z = 0$). Having the 3D coordinates and a clockwise polygon describing the exterior boundary of the floor plan, enables the usage of a Constrained Delaney Triangulation (CDT) [9]. A CDT is a generalized triangulation of a set of vertices with the following properties:

- Edges which are pre-defined are part of the triangulation.
- Triangles of the triangulation do not cross the pre-defined polygon.
- The triangulation is as close as possible to the Delaunay triangulation.

The advantage of the CDT is that the triangulation of concave polygons is possible and it is feasible to exclude all triangles which are located outside of the polygon.

The boundary of the binary mask is now used to specify the constraints for the Constrained Delaunay triangulation. The resulting triangulation supports the traced boundary and further

the concavities. Figure 3.12a displays the result of the CDT algorithm. All polygons which lie outside the specified polygon are removed (cf. Figure 3.12b). Figure 3.12 clearly shows the difference between the convex hull of the polygon and the CDT.



(a) (b) (c)

Figure 3.12: Floor plan triangulation and simplification. (a) Constrained Delaunay triangulation of the binary mask. (b) Triangles outside the polygon are removed. (c) Simplified version of the mesh-based representation.

Floor Plan Simplification In order to reduce the number of vertices in the final model, the triangulation of the floor plan is further simplified. Therefore, we use the approach described in [25], which simplifies meshes by iteratively contracting edges. The simplification algorithm works as follows: Two vertices v_1 and v_2 which define a random edge of the mesh are collapsed to a new vertex $\bar{v}$ and all incident edges are connected with the vertex $\bar{v}$. Figure 3.13 illustrates the idea of the edge contraction method.

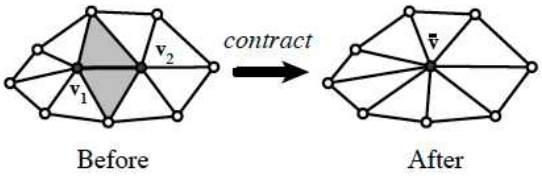


Figure 3.13: Edge contraction. The edge $(\mathbf{v}_1, \mathbf{v}_2)$ of the triangulation is contracted to a new vertex $\bar{\mathbf{v}}$. (Figure taken from [25])

Valid pairs for the contraction as well as the position of the new vertex $\bar{\mathbf{v}}$ are estimated by using a *quadric error matrix* for each vertex, based on the intersections of neighbouring planes. The so called *fundamental error matrix* $\mathtt{K}_p$ which describes the squared distance between the plane of a triangle and any point (vertex) in the space is defined by

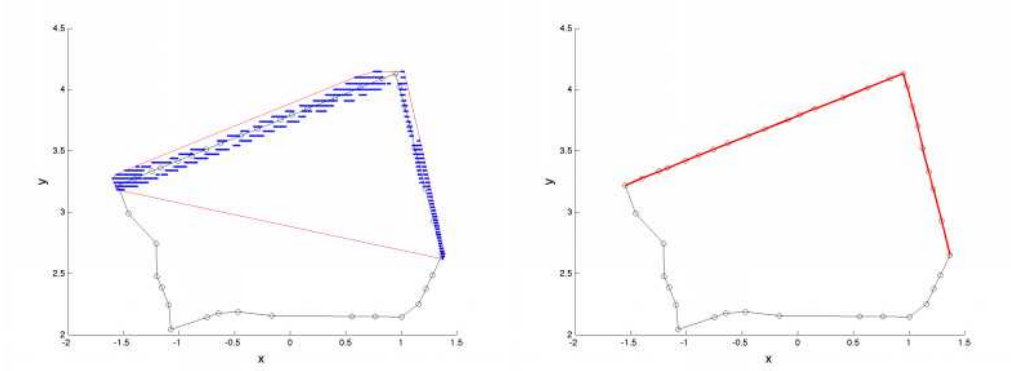
$$\mathtt{K}_p = \begin{pmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{pmatrix}, \tag{3.4}$$

where the variables are defined by a plane equation. The error matrices $\mathtt{K}_p$ are formulated for all planes referenced by a vertex and further accumulated in order to get a single error matrix $\mathtt{Q}$

for each vertex of the mesh. The simplification algorithm estimates (1) a error matrix for each vertex of the mesh and (2) iteratively contracts the edges which deliver the lowest contraction costs given by $\bar{\mathtt{Q}} = \mathtt{Q}_1 + \mathtt{Q}_2$.

The simplified 2D mesh of our running example is illustrated in Figure 3.12c. We reduce the vertices to 15% of the initial number of triangles. This value preserves the overall shape of the floor plan and results in a simplified version of the mesh at the same time (for validation compare our experiment section).

Wall Extrusion The proposed layout estimation framework describes walls by extruding relevant boundary edges of the 2D floor plan. Thus, in a first step the boundary edges of the triangulation are required. The boundary edges are determined by detecting edges which are referenced by one triangle only. Secondly, we have to find all segments which belong to walls in the scene. A straightforward way to find those edges is to check the Euclidean distance between the edge segments and the points of the plane-fitting approach which have been projected on the ground plane. Since missing points would lead to missing walls, we decide to estimate the convex hull of the projected wall points. Thus, every edge located inside the convex hull is used for the further extrusion approach. Figure 3.14 shows the boundary edges of the sample scene and the idea of the convex hull method.



(a) (b)

Figure 3.14: (a) Edge segments located inside the convex hull of the projected wall points. (b) The edges which are colored *red* indicate wall segments which are used for the wall extrusion approach.

After the edge segment of the floor plan which belong to walls have been estimated, the corresponding vertices are used for extruding wall segments and the ceiling. Thus, the roof and the walls are estimated by extruding the relevant vertices until they intersect with the ceiling, or if no ceiling is available, up to a pre-defined height of 2.5 meters. For simplification, we assume that the ceiling has the same shape as the floor plan. A compact and watertight 3D representation of the scene is then obtained by triangulating the extruded segments. The algorithm used for the extrusion and the generation of the finale triangulation is listed in Algorithm 3.2. The algorithm requires the vertices and the faces of the floor plan triangulation as well as a list of the edges

which are referenced by wall segment. The final 3D layout obtained by the extrusion algorithm is visualized in Figure3.15.

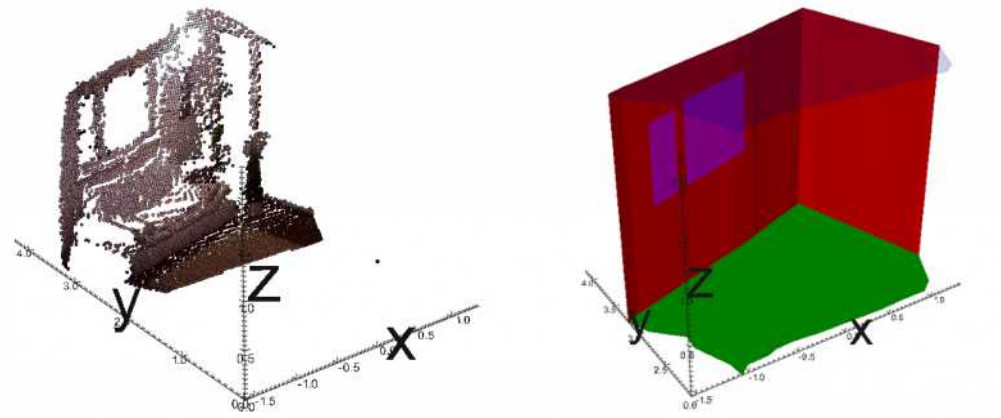


Figure 3.15: Input point cloud of the sample scene (left) and the corresponding 3D layout (right).

3.2.6 Window Detection

In order to obtain semantically meaningful 3D layouts, the objective of this step is to detect windows in the scene and further integrate the detected windows into the 3D model. Windows are detected in the intensity image of the scene. The following three steps are conducted to localize and visualize images:

- Window candidates are estimated in the depth image.
- The shape of the windows and an the position in the 3D space are examined in order to remove wrong candidates.
- The detected windows are projected onto the walls.

Window candidate estimation Windows are localized by seeking for rectangular blobs in the intensity image. For simplification we assume that windows are areas where the Kinect sensor does not return a valid depth measure. Thus, we first create a binary image where missing values are colored *white* and the remaining pixels are colored *black*. Figure 3.16a shows the resulting binary image. Since the image is noisy, morphological opening [15] (erosion followed by dilatation) is performed on the image in order to remove noise. Secondly, the closing operator (dilatation followed by erosion) is used to fill gaps and to connect labels which are located near each other. Blobs which are too large (regarding the number of pixels) are sorted out. The resulting blobs for our sample scene are illustrated in Figure 3.16b.

Rectangle detection Each previous calculated blob is a candidate for the following shape detection procedure. Since windows are supposed to be rectangular, we first estimate straight

Input : Vertices V_{xyz}, Faces of floor plan TRI_{floor}, Boundary edges $edges$, Height of scene h_{scene}
Output: Vertices of the 3D layout V'_{xyz} and the corresponding faces TRI_{model}

// Length of the triangulation and the final triangulation
nVertices $\leftarrow$ `getLenght(`$V_{xyz}.X$`)`;
nVertices' $\leftarrow$ nVertices $* 2$;
// Create vertices for floor and ceiling
V_{xyz} $[1, .., \text{nVertices'}] \leftarrow 0$;
for $iVertex \leftarrow 1$ **to** nVertices **do**
    $V'_{xyz}[iVertex].X \leftarrow V_{xyz}[iVertex].X$;
    $V'_{xyz}[iVertex].Y \leftarrow V_{xyz}[iVertex].Y$;
    $V'_{xyz}[iVertex].Z \leftarrow V_{xyz}[iVertex].Z$;
end
for $iVertex \leftarrow$ nVertices $+ 1$ **to** nVertices' **do**
    $V'_{xyz}[iVertex].X \leftarrow V_{xyz}[iVertex].X$;
    $V'_{xyz}[iVertex].Y \leftarrow V_{xyz}[iVertex].Y$;
    $V'_{xyz}[iVertex].Z \leftarrow V_{xyz}[iVertex].Z + h_{scene}$;
end
// Create faces for floor and ceiling
$TRI_{model}.Floor \leftarrow TRI_{floor}$;
$TRI_{model}.Ceiling \leftarrow TRI_{floor} +$ nVertices;
// Create faces for the walls;
$index \leftarrow 0$;
for $iEdge \leftarrow 1$ **to** $edges.size$ **do**
    // Get indices of edge and create two new triangles for each edge
    // Indices point to positions in the coordinates list V_{xyz}
    $tmpEdge \leftarrow edge[iEdge].Indices$;
    $triangle_1 = [edge(1), edge(2), edge(2) +$ nVertices$]$;
    $triangle_2 = [edge(1) +$ nVertices$, edge(2) +$ nVertices$, edge(2) + edge(1)]$;
    $index++$;
    $TRI_{walls}[index] \leftarrow triangle_1$;
    $index++$;
    $TRI_{walls}[index] \leftarrow triangle_2$;
end
$TRI_{model}.Walls \leftarrow TRI_{walls}$;

Algorithm 3.2: 3D layout estimation. A 3D model is obtained by extruding the faces and the corresponding vertices of the floor plan triangulation.

(a) (b) (c)

Figure 3.16: Window detection. (a) A binary image indicates missing depth values. (b) Morphological operations are used to remove noise and to detect window candidates. (c) Rectangular blobs are estimated and incorrect shapes are sorted out.

lines using a Hough transformation method [16]. The detected lines are sorted by their length and the four longest lines are stored. A line-line intersection method is now used to find the intersections between all lines. Equation 3.5 depicts the calculation of the intersection point (P_x, P_y) between two lines L_1 and L_2 where the distinct points (x_1, y_1) and (x_2, y_2) lie on L_1, and correspondingly (x_3, y_3) and (x_4, y_4) on L_2.

$$P_x = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}, P_y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}. \tag{3.5}$$

The estimated intersection points are sorted in a clockwise order and further paired in order to get the four sides of the rectangle. If the sides satisfy certain geometric constraints, the blob is supposed to be of a rectangular shape. Figure 3.16c shows the resulting rectangular blobs projected onto the RGB image. In addition, the position of the detected window in the 3D space is considered. Therefore, the mean distance of the neighbouring points to the nearest wall (3D plane) is examined. Window candidates with a distance > 20cm are sorted out.

Wall projection The last stage in the window detection process is the projection of the window onto the corresponding wall. Therefore, we determine the edge segment in the 2D floor plan triangulation to which the window belongs to. Having this information allows the projection of the the windows' corner points onto the corresponding edge segment. The final window is estimated by extruding the points by the dimensions (height and position on the plane) of the detected blob.

In our sample scene two windows are detected as illustrated in Figure 3.15. The two blobs on the right side (cf. Figure 3.16c) are sorted out because the position of the blobs in the layout is incorrect or no valid 3D points are available in the neighbourhood.

3.2.7 Temporal Fusion

According to our proposed layout estimation pipeline (cf. Figure 3.2) in the previous sections we have presented how 3D layouts are estimated based on single RGBD frames. In Section 3.2.2 we have seen how a transformation matrix is estimated which aligns overlapping point clouds. In this section we present how fused 3D layouts are obtained in order to get a global 3D model for an entire video sequence. Our proposed layout estimation pipeline intuitively allows the fusion of several single layouts and the extension of scenes by updating the floor plan over the time. Therefore, the following steps are performed:

- 1. The 2D floor plan of the fused point clouds is estimated.
- 2. The 3D layout is obtained by simplifying and extruding the 2D mesh of the floor plan.

For the generation of the whole 3D layout, only the updated floor plan of of the underlying fused scene (point cloud)is required. The remaining steps of the pipeline – namely the trimming of the rough floor plan concerning the detected walls, the triangulation of the floor plan, the simplification of the mesh and the layout extrusion – are analogue to the workflow of processing a single RGBD frame. In the next section we show how multiple floor plans are fused.

2D Floor Plan Estimation

The objective of this step is the estimation of a 2D mesh of the floor plan for the entire scene. Having two consequence input shots $frame_i$ and $frame_j$ where $j = i+1$, and the corresponding point clouds, we have estimated a rotation matrix $\mathtt{R}$ and a translation vector $\mathbf{t}$ which registers the two clouds. Since the single 3D layouts obtained by our pipeline are perpendicular to the ground plane, we are only interested in the rotation around the z-axis. Given a 3x3 rotation matrix

$$\mathtt{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \tag{3.6}$$

the decomposition of the matrix returns the three Euler angles θ_x, θ_y and θ_z. The angles are estimated by

$$\begin{aligned} \theta_x &= atan2(r_{32}, r_{32}) \\ \theta_y &= atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \theta_z &= atan2(r_{21}, r_{33}), \end{aligned} \tag{3.7}$$

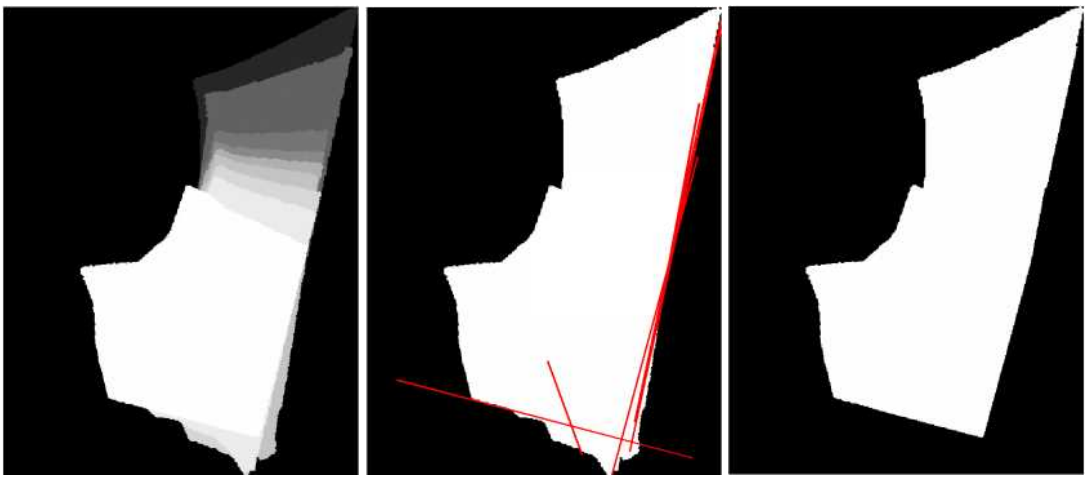
where $atan2$ describes the four-quadrant inverse tangent function. Consequently, the rigid transformation matrix $\mathtt{T}$ which aligns two layouts is expressed as a combination of the rotation matrix $\mathtt{R_z}$ and the translation vector $\mathbf{t}$ and calculated as depicted in Equation 3.8.

$$\mathtt{T} = \begin{pmatrix} cos(\theta_z) & -sin(\theta_z) & 0 & t_x \\ sin(\theta_z) & cos(\theta_z) & 0 & t_y \\ 0 & 0 & 0 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.8}$$

For a video sequence with N frames, the fused floor plan is now obtained by incrementally stitching together the adjacent 2D floor plans (meshes) of the corresponding input frames. Since we have a transformation matrix for each frame pair available, the accumulated transformation matrix $\mathtt{T}'$ at a certain position k ($1 \leq k \leq N$) is determined by

$$\mathtt{T}' = \prod_{i=1}^{k} \mathtt{T}_i, \tag{3.9}$$

where $\mathtt{T_1} = \mathtt{I_4}$ (identity matrix). After the floor plan for each input frame has been estimated, the vertices of the triangulations are transformed with the corresponding transformations matrices. The resulting overlapping mesh is converted into a binary image in the same way as for a single layout. Figure 3.17a shows the binary mask of the conducted floor plan for our sample video.



(a) (b) (c)

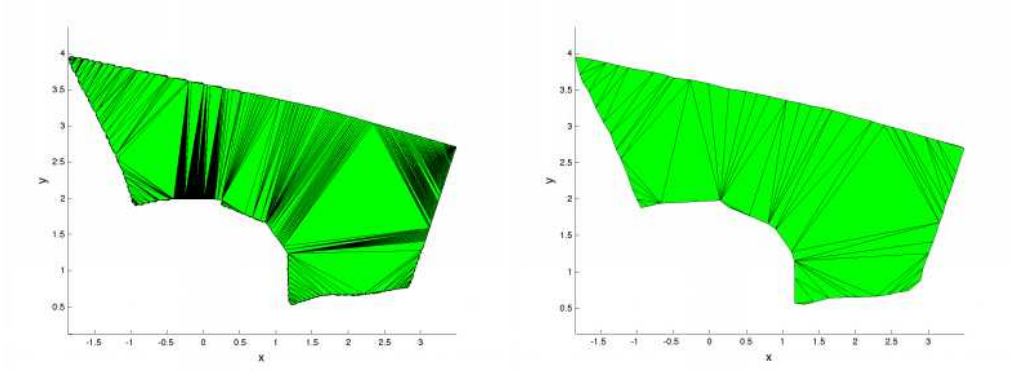
Figure 3.17: Fused floor plan. (a) Alignment of consecutive floor plans coming from multiple shots. (b) Overlapping wall segments (red lines) are used to trim the rough floor plan . (c) Final floor plan of the fused scene.

Because of missing points and clutter walls coming from the single layouts are may detected inaccurate, which leads to incorrect RANSAC planes and consequently the corresponding floor plan is not rectilinear. This artefacts lead to wall segments in the 3D layout which are not running

smooth. For that reason, we transform the fitted line segments of each RANSAC wall with the same transformation matrix which had already been used to align the layouts. The line segments are extended in both directions in order to obtain better intersections. Figure 3.17b illustrates the idea of overlapping line segments. In Figure 3.17c the trimmed floor plan is shown.

3D Layout Estimation and Simplification

The last step of the pipeline involves the estimation of the whole 3D layout for the fused floor plan. These processing steps are analogue to the single case (cf. Section 3.2.5). The triangulated floor plan as well as the simplified 2D mesh is shown in Figure 3.18a and Figure 3.18b.



(a) (b)

Figure 3.18: Updated floor plan triangulation. (a) Constrained Delaunay Triangulation. (b) Simplified mesh representation.

When fusing multiple frames, we do not use the convex hull approach to estimate the edge segments of the triangulation which are extruded. Since the fusion of several point clouds reduces the problem of clutter and missing points, valid edge segments are obtained by examining the minimum distance of the edge candidates to the 2D points of the projected RANSAC wall points. Each edge of the floor plan mesh with a distance smaller than a threshold T_{dist} is considered as a valid edge. In our experiments we use a threshold of $T_{dist} = 4cm$. In addition, the resulting edges are sorted in clockwise order and small holes (Euclidean distance smaller than $3 \cdot T_{dist}$) are closed using a traversal algorithm.

Figure 3.19 shows the finale 3D layout of our running example. It is noticeable that the simplified model consist of a reduced number of points. The simplified 3D model consist only of 360 vertices, while the raw point cloud is composed of 3 543 200 points. (Please see our experiment section for more details.)

3.3 3D Object Detection and Pose Estimation

This section outlines how objects are detected and the corresponding poses are estimated by the proposed recognition framework. The main idea is to extract 3D points belonging to a specific

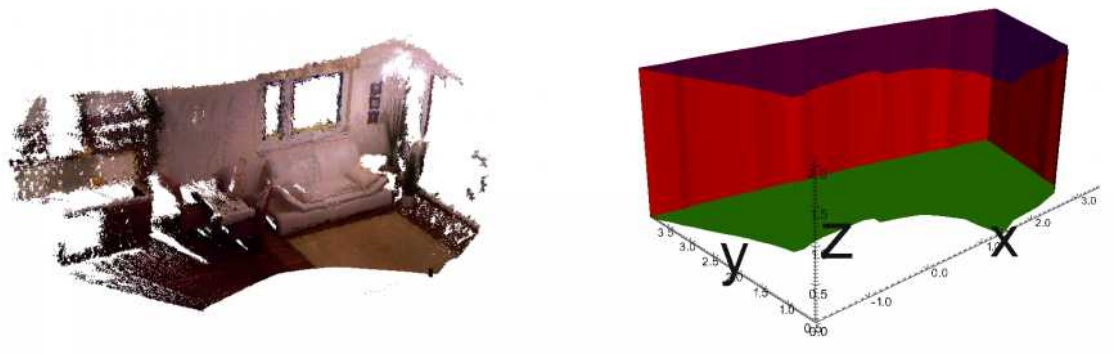


Figure 3.19: Fused point cloud of the sample video (left) and the corresponding 3D layout obtained by the simplification framework (right).

object detected in the image space, and further search for similar 3D objects in a pre-defined database. Therefore, a point cloud descriptor which encodes shape and viewpoint – the Viewpoint Feature Histogram (VFH) descriptor – is used. The detection pipeline for a single RGBD frame is shown in Figure 3.20. The classification schema is further optimized by combining the detection and classification results coming from multiple consecutive frames (cf. Section 3.3.6).

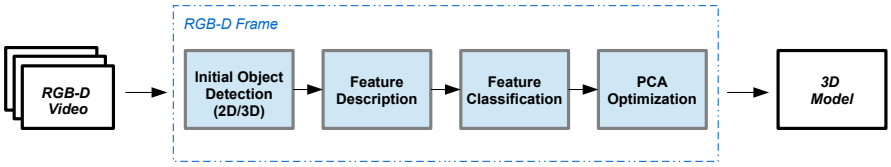


Figure 3.20: Object detection pipeline. For each frame an initial detection and pose estimation hypothesis is estimated.

Since the proposed framework is trained with synthetic models from the Googles' 3D Warehouse[6], Section 3.3.1 first presents how synthetic point clouds are generated. The resulting partial point clouds are further used to train a classifier.

3.3.1 Synthetic CAD Model Training

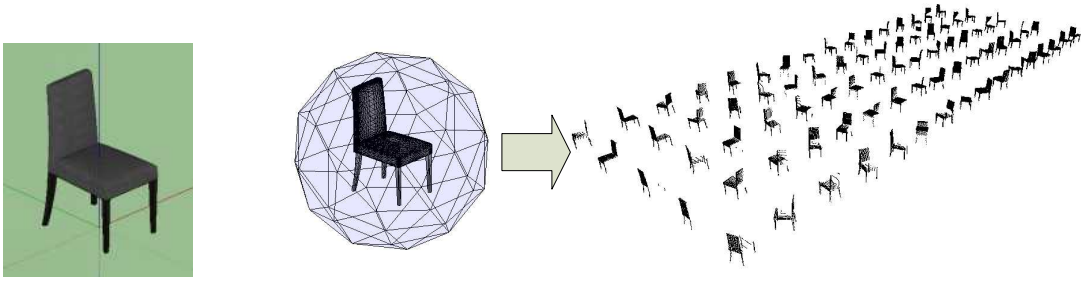
In this thesis we use shape retrieval techniques in order to detect objects visible in the scenes. Therefore, we present a 3D point cloud descriptor (VFH descriptor), which is used to search for similar point clouds in a trained database. This database consists of synthetically generated point clouds. Thus, we present how synthetic point clouds are rendered from CAD models. Since the point cloud descriptor used in this work is build from normal vectors, we outline how normal vectors are obtained from single point clouds. The whole rendering of the partial views, the

[6] `http://sketchup.google.com/3dwarehouse/` (last access: 17.04.2015)

calculation of the descriptors and the generation of a search space is done in an offline training stage.

Partial Point Cloud Generation

The point cloud descriptor presented by Rusu *et al*. [57] is trained online with point clouds captured from a stereo camera at different angles and offsets. Since such a training is time-consuming and inefficient [2, 3], we train our framework offline with CAD models downloaded from the web. Our framework is limited to four object classes – namely *couch*, *chair*, *table* and *potted-plant*. The CAD models are downloaded from a public database, centered in a single coordinate system and the meshes are exported as PLY files which define the vertices and faces. A sample CAD model used to train our system is shown in Figure 3.21a.



(a) (b)

Figure 3.21: (a) Sample CAD model downloaded from a web database. (b) Virtual cameras are uniformly placed around a bounding sphere and partial point clouds are rendered, showing the model from the different camera positions.

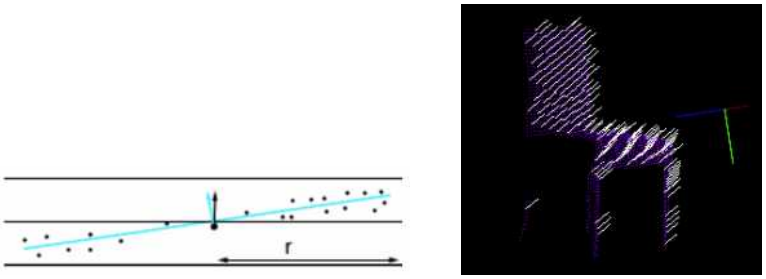
Since objects in real world scenarios are viewed from a single viewpoint, parts of the point clouds are missing because of self-occlusion. Such representations are called 2.5D point clouds [50]. Consequently, a set of distinguishable views for each CAD model is required, simulating the input given by a depth sensor. For this purpose we place a virtual camera uniformly around the CAD model. We use a icosahedron to approximate a sphere and further refine the mesh of the icosahedron using a surface subdivision algorithm [8] which generates four equilateral triangles for each face. The resulting mesh is a tessellated sphere which consists of 80 faces (cf. Figure 3.21b).

A CAD model is now placed at the center of the sphere and the barycenters of the faces are used to place virtual cameras, looking at the CAD model. The mesh-based model is next rendered from each viewpoint using a uniform sampling technique and partial point clouds are generated by reading the depth buffer of the graphics card. Consequently, 80 synthetic points clouds are obtained, showing the CAD model from all possible sides. For each partial view an additional pose information (transformation matrix) is available which describes the transformation between the model and the view coordinates. This view depicts a partial point cloud of the original CAD model as seen from the virtual camera.

We assume that the objects in our real world scenarios are sitting perpendicular on the ground plane visible in the scene. Thus, each partial point cloud is transformed on the ground plane using the Euler angles obtained from the transformation matrix. Infeasible poses, *i.e.* the model rendered from below, are sorted out, and the remaining partial point clouds are stored in a database for further use. Figure 3.21b) shows the partial point clouds of an sample CAD model. Please note that we use a rendering framework provided by the PCL library to render the partial point clouds.

Surface Normal Estimation

The normal vector to each point is required in order to calculate the VFH descriptor for a point cloud [57]. We estimate normal vectors for the generated partial point clouds with the approach provided by the PCL library (for details see [55]). First, the point clouds are downsampled using a voxel grid filter with a leaf size of 3cm. The normal vector to a point on the surface is inferred by approximating the normal vector of the plane tangent to the surface. Thus, the normal vector at a specific point is determined by fitting a plane into the k-neighborhood of the query point. Neighbouring points are those points within a sphere of a given radius r which is centered at the query point. (In our experiments $r = 30cm$.) Figure 3.22a visualizes the idea of the least square plane fitting approach. Since the viewpoints $\mathbf{V}$ of the sensor is known, the direction of the



(a) (b)

Figure 3.22: Surface normal estimation. (a) The normal vector at a query point is given by the normal vector of a plane which is fitted into the neighbouring points. (b) Normals vectors at the surface pointing towards the viewpoint.

normal vectors is calculated by orienting the normals towards the viewpoint. A normal vector $\mathbf{n}_i$ at a query point $\mathbf{p}_i$ points towards the viewpoint $\mathbf{V}$ if Equation 3.10 is satisfied.

$$\mathbf{n}_i \cdot (\mathbf{V} - \mathbf{p}_i) > 0. \tag{3.10}$$

A fast way to determine the normal vector of the planes is the calculation of the eigenvectors and eigenvalues of a covariance matrix of the points in the k-neighbourhood. (See [55] for more details). The obtained normal vectors for a point cloud of our database, pointing towards the viewpoint, is shown in Figure 3.22b (not all normal vectors are displayed).

Point Cloud Descriptor (VFH)

Having partial point clouds and the normal vectors at each point of the surface, we estimate shape-based features in order to represent the point clouds in a more comparable way. Therefore, we calculate a VFH descriptor for each point cloud. The general idea behind shape-based retrieval is to compare objects by measuring the similarity between their signatures [53]. Thus, two point clouds are compared by measuring the distance between their histograms (the VFH descriptors), using a distance metric (*e.g.* L_1 distance). The VFH descriptor is an extension of the FPFH descriptor [56], which represents the relative orientation of normal vectors and the distance between point pairs. Moreover, the VFH descriptor is invariant to scale transformations, which is a desirable feature because we do not know the dimensions of the objects in the scenes in advance.

Shape Component In an initial step the centroid $\mathbf{c}$ of the entire point cloud and the corresponding normal vector $\mathbf{n}_c$ is determined. (The normal vector of the centroid is given by the arithmetic mean of the surface normals.) Secondly, the three orthonormal vectors $\langle \mathbf{u}_j, \mathbf{v}_j, \mathbf{w}_j \rangle$ which form a Darboux coordinate system [57], are estimated at each surface point $\mathbf{p_j}$. The unit vector $\mathbf{u}_j$, the unit tangent vector $\mathbf{v}_j$ and the tangent normal vector $\mathbf{w}_j$ are determined by

$$\mathbf{u}_j = \mathbf{n}_c \tag{3.11}$$

$$\mathbf{v}_j = \mathbf{u}_j \times \frac{\mathbf{p}_j - \mathbf{c}}{\|\mathbf{p}_j - \mathbf{c}\|} \tag{3.12}$$

$$\mathbf{w}_j = \mathbf{u}_j \times \mathbf{v}_j. \tag{3.13}$$

The VFH descriptor calculates the angular deviations pan (α), tilt (ϕ) and yaw (θ) between the point's normal vector $\mathbf{n}_j$ and the normal vector at the centroid $\mathbf{n}_c$ (cf. Figure 3.23) as depicted in the following equations:

$$\alpha = \arccos\left(\mathbf{v}_j \cdot \mathbf{n}_j\right) \tag{3.14}$$

$$\phi = \arccos\left(\mathbf{u}_j \cdot \frac{\mathbf{p}_j - \mathbf{c}}{\|\mathbf{p}_j - \mathbf{c}\|}\right) \tag{3.15}$$

$$\theta = \arctan\left(\mathbf{w}_j \cdot \mathbf{n}_j, \mathbf{u}_j \cdot \mathbf{n}_j\right). \tag{3.16}$$

The values $\langle \alpha, \phi, \theta \rangle$ are calculated for all possible pairings and further binned into a histogram to create a VFH signature. This component of the signature describes the shape of the point clouds.

Viewpoint Component In addition, the VFH descriptor computes a viewpoint component between the central viewpoint direction and each point's normal vector $\mathbf{n}_j$ as shown in Figure 3.23. Therefore the angle β which depends on the viewpoint $\mathbf{V}$ is determined as depicted in Equation 3.17.

$$\beta = \arccos\left(\mathbf{n}_j \cdot \frac{\mathbf{V} - \mathbf{c}}{\|\mathbf{V} - \mathbf{c}\|}\right). \tag{3.17}$$

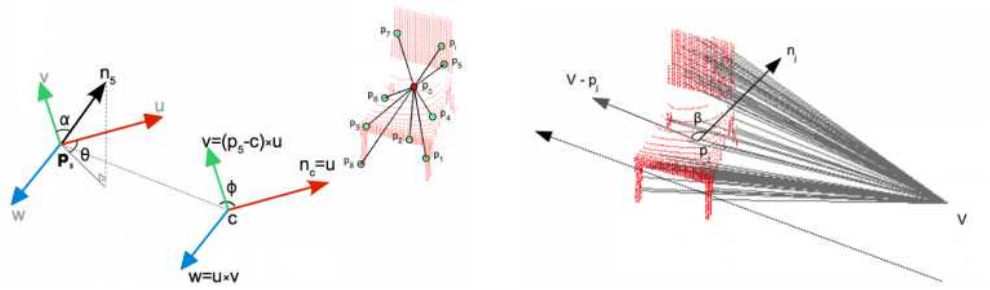


Figure 3.23: Viewpoint Feature Histogram. (a) The relative angles $\langle \alpha, \phi, \theta \rangle$ are calculated between the normal vector $\mathbf{n}_c$ of the centroid and and each surface normal $\mathbf{n}_i$. The vectors $\langle \mathbf{u}, \mathbf{v}, \mathbf{w} \rangle$ form a Darboux coordinate frame at each surface point. (b) A viewpoint component is estimated based on the relative angles between the surface normals $\mathbf{n}_i$ and a central viewpoint direction. direction.

The final descriptor is given by the concatenation of both histograms. The signature has a predefined length of 308 bins [57]. A sample signature is illustrated in Figure 3.24. Please note that the VFH descriptor is invariant against rotations around the camera axis. Since our models are supposed to be placed perpendicular to the ground plane, that is no limitation four our framework.

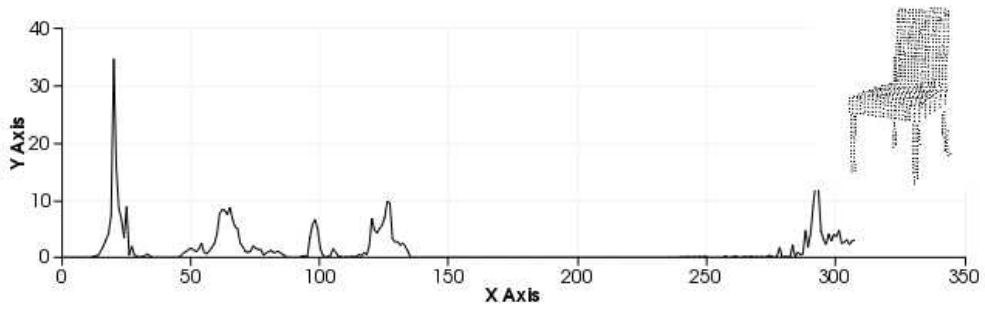


Figure 3.24: A resulting VFH signature for an partial point clouds of our database.

For each partial point cloud obtained from the CAD models a VFH descriptor is estimated in the offline training stage. The descriptors and the pose information (transformation matrix) for each trained model are stored in a database.

3.3.2 From 2D to 3D - Initial RGB Object Detection

Since multimodal data is available, first a trained RGB detector is applied on the color image in order to obtain initial object detection results for the further 3D classification. Therefore we use the DPM[7] object detector presented by Felzenszwalb *et al.* [21].

[7] `http://www.cs.berkeley.edu/~rbg/latent/` (last access: 17.04.2015)

Deformable Part Model Detection The DPM detector is based on a sliding window approach. Objects are represented with an *root filter* and additional flexible *part filters*. The idea is to represent an object at different scale levels. The DPM algorithm is based on the HOG feature descriptor [14]. The image is therefore divided into non-overlapping cells, and the gradient orientations within the cells are accumulated. The resulting 1D histogram captures local shape properties of the image.

First, a feature map (HOG features) is created at different scale levels of a standard pyramid. A coarse root filter covers the entire object at a low resolution in the pyramid. The root filter describes a detection window. Part filters are covering smaller parts of the object at higher resolutions, several levels down in the pyramid, *e.g.* the models of a face – The root filter could capture the face boundaries, while the part filters would correspond to details like the nose or the mouth. Thus, the *root* template and the *part* templates capture local properties of the entire object.

A spatial model defines spring-like connections between the parts within the global detection window. Those connections represent the possible placements of the part filters. Deformation costs are used to vote each placement. Having a spatial model, the filters and the deformation costs, enables the usage of dynamic programming and generalized distance transform methods (see [21] for more details) in order to solve the detection problem. Similar objects are detected by solving the so called matching problem. As mentioned, objects are represented as spatial models – In other words a model is described as a graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where:

- $\mathcal{V}$ describes the parts and
- $\mathcal{E}$ defines the connection between parts.

The matching problem is now solved by finding the optimal configuration $\mathbf{L} = \{l_1, l_2, \dots l_n\}$ for an object, which is determined by minimizing an error function

$$E(\mathbf{L}) = \sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in\mathcal{E}} d_{ij}(l_i, l_j). \tag{3.18}$$

The date term $m_i(l_i)$ describes the costs for placing a filter (part) and the smoothness term $d_{ij}(l_i, l_j)$ defines the deformation costs. Figure 3.25 visualizes the mentioned ideas of the DPM detector.

The result of the DPM object detector is a 2D bounding box which specifies the location of detected objects in the color image. Since we have the corresponding 3D points available for the pixels within the bounding box in the image space, we are able to create a single point cloud for each detected object. Figure 3.26 shows the 2D detection results for our running example. Please note that more than one objects is detected.

A main advantage of the 2D pre-detection step is that it is possible to extract objects which are placed close together *e.g.* in a man-made environment a shelf is placed close beside a couch. Without the 2D detection approach it is more complicated to extract such object clusters which are used for the further point cloud classification. Similar point clouds classification works [57, 69] use only a clustering approach to determine point clouds candidates.

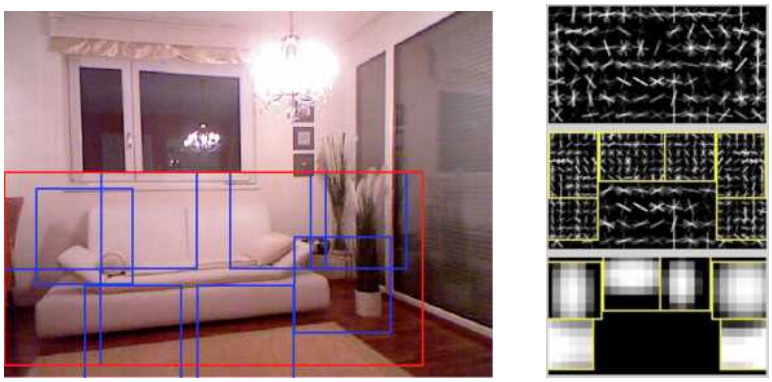


Figure 3.25: (a) A model is defined by a root filter and several part filters. (b) Sample model of a trained couch. HOG features describe the appearance of the root template (top) and higher resolution part templates (middle). A spatial model is used to define possible placement, where brighter values denote cheaper deformation costs (bottom).

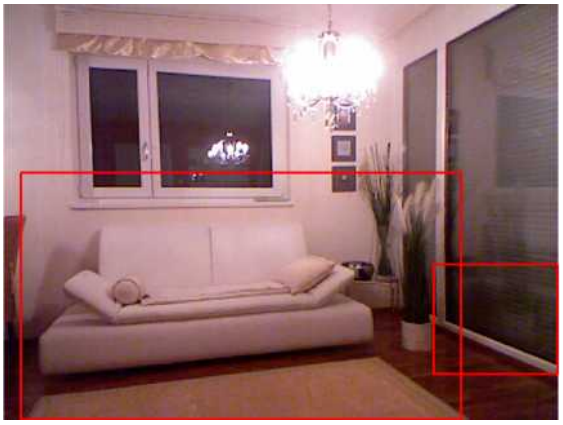


Figure 3.26: 2D Object detection outcome. The DPM object detector returns bounding boxes for all detected objects visible in the scene.

3.3.3 3D Object Cluster Estimation and Euclidean Clustering

After determining the 3D information for each detected object visible in the scene, the extracted point clouds are used in order to search for similar objects in the trained database of the partial point clouds. Since the 2D detector returns a bounding box, redundant points are eliminated by performing an object cluster extraction method and Euclidean clustering.

Object Cluster Extraction All points belonging to a *wall* or the *ground plane* are removed. Therefore, the Euclidean distance of the 3D points to all estimated RANSAC planes is determined. Having a 3D point $\mathbf{p} = (x,\ y,\ z)^T$, the coefficients of the plane equation (a, b, c, d) and the plane's normal vector $\mathbf{n} = (a,\ b,\ c)^T$, the distance d between a point and a plane is

determined as depicted in Equation 3.19.

$$d = \frac{a \cdot x + b \cdot y + c \cdot z + d}{\sqrt{a^2 + b^2 + c^2}}. \tag{3.19}$$

All points with a distance $abs(d) > 5cm$ are removed because they are supposed to be part of a layout segment. The second image in Figure 3.27 shows the point cloud after all point belonging to a *wall* and the *ground plane* are removed.

Figure 3.27: From left to right: Extracted point cloud resulting from the 2D object detection, 3D cluster extraction and the final point cloud after Euclidean clustering.

Since we assume that the objects are placed on the floor, we ignore clusters which are found above a certain height (50cm in our experiments). In addition, clusters which do not consist of a meaningful number of points are also excluded from all further classifications. This approach enables us to filter out false positives in an early step of the classification workflow.

Euclidean Clustering The remaining points of the estimated cluster may still consist of noise and redundant points. Reasons are points located in front of the detected object, incorrect bounding boxes or inaccurate fitted 3D planes. For example in our sample scene the extracted point cloud still consists of points belonging to the window board (cf. Figure 3.27). We therefore first apply a voxel grid filter on the point cluster. Secondly, the remaining points are clustered using the Euclidean cluster extraction approach provided by he PCL library. A clustering approach divides an unorganized point cloud into several non-overlapping point cloud clusters. For further details about the clustering approach, please see [55].

The largest cluster is then taken as representative for the object in question and is used for the following classification. The rightmost image in Figure 3.27 shows the extracted point cloud after the Euclidean clustering for our running scene.

3.3.4 Feature Description and Classification

Since a coherent and dense point cloud for the detected object candidate is available, the goal of the next step is to find similar objects – concerning pose and type – in the created training database. We first calculate a VFH descriptor for the extracted point cloud. Similar point clouds are now determined using a K-Nearest Neighbours (KNN) search algorithm based on a K-Dimensional (k-D) tree with a Chi-Square distance metric. The Chi-Square distance $d_{chi}(\mathbf{A}, \mathbf{B})$

between two signatures $\mathbf{A}$ and $\mathbf{B}$ is defined as

$$d_{chi}(\mathbf{A}, \mathbf{B}) = \frac{1}{2} \cdot \sum_i \frac{(\mathbf{A}_i - \mathbf{B}_i)^2}{\mathbf{A}_i + \mathbf{B}_i}. \tag{3.20}$$

In our framework we determine the 10 most similar point clouds for the object in question ($N = 10$). As a result we get a first ranking of similar objects. The five best results for our sample object are shown in Figure 3.28.

Figure 3.28: From upper left to bottom right: Most similar point clouds (concerning pose and type) based on the comparison of VFH descriptors.

3.3.5 PCA Optimization

Please note that the best matching result in the example does not necessarily deliver the optimal pose. The VFH descriptor has problems dealing with missing points, caused by occlusion, segmentation artefacts, direct illumination or specular surfaces [3]. Figure 3.28 shows that the result on position $K = 3$ or $K = 5$ does not describe the pose of the detected *couch* in an appropriate way. Thus, we propose a combination of the VFH descriptor with a second descriptor, namely a Principal Component Analysis (PCA) descriptor, in order to re-sort the VFH ranking. The PCA descriptor provides more robust poses even if points of the objects are missing (compare our experiments section for validation). The following paragraph shortly describes the Principal Component Analysis procedure.

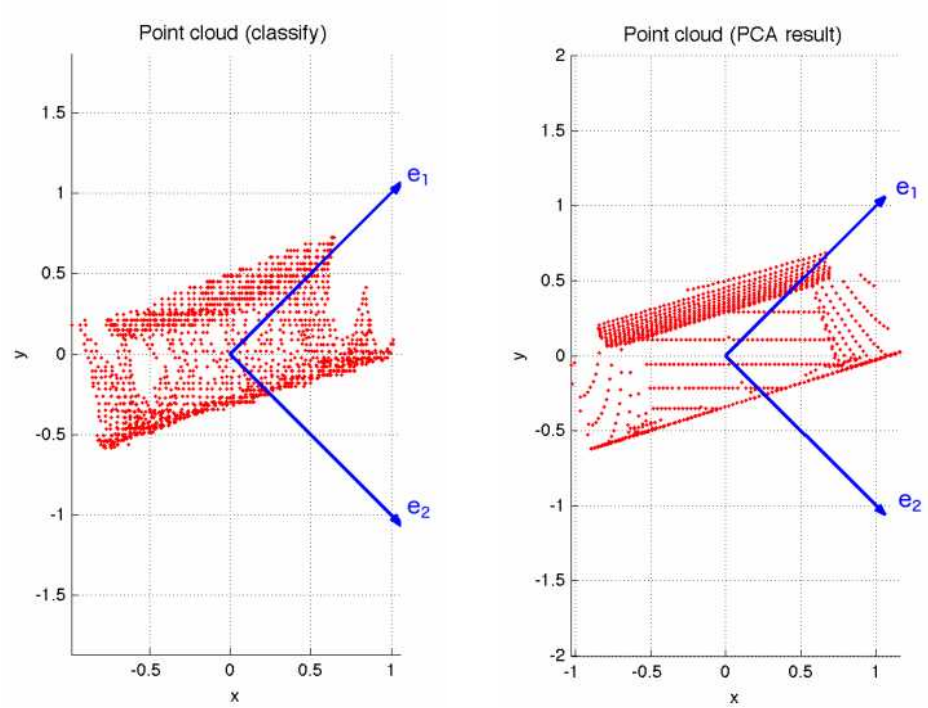
Principal Component Analysis The PCA is a statistical method which explores data by analysing the covariance structure of a set of variables. In our case the PCA method is used to obtain a coordinate system which defines the axis in which the point cloud varies a lot. In case of a 2D point cloud, the result of the PCA analysis are orthonormal eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$ (cf. Figure 3.29) in which:

- $\mathbf{e}_1$ defines the principal direction (dominant direction)
- $\mathbf{e}_2$ defines the second dominant direction

The vectors are called *principal components* and are perpendicular to each other.

The goal of the PCA optimization is to determine those point clouds of the $K = 10$ VFH results which provide similar poses regarding the point cloud in question. The orientation of a point cloud is therefore described by calculating the eigenvectors. First, all points belonging to the object cluster are projected on the *ground plane*. We then calculate the eigenvectors of the 2D point cloud's covariance matrix using a PCA algorithm. The resulting two vectors represent the orientation of the point cloud. Only the principal direction is used to describe the orientation of the entire model.

In the same manner the eigenvectors for the $K = 10$ most similar VFH results are calculated. Figure 3.29 shows the obtained eigenvectors for the *couch* of our sample scene and for a point cloud of the database with a similar orientation.



(a) (b)

Figure 3.29: PCA eigenvectors $(\mathbf{e}_1, \mathbf{e}_2)$ are used to describe the orientation of point clouds. (a) Eigenvectors of the detected object's point cloud. (b) Trained partial point cloud with a similar orientation.

We assume that point clouds with similar orientations deliver similar eigenvectors. Therefore, the angles between the principal eigenvectors of the object in question and all point clouds provided by the VFH neighbour search are calculated. The angles are sorted in descending order which results in a re-ranking of the VFH results. The re-ranking of the VFH point clouds for our running example is visualized in Figure 3.30.

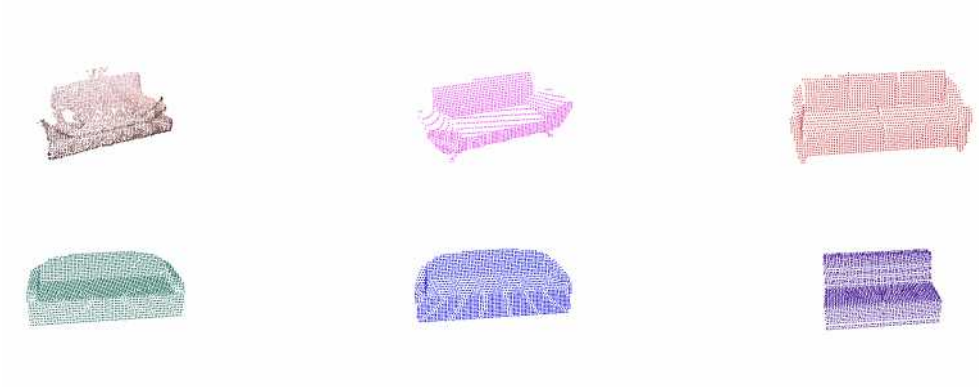


Figure 3.30: From upper left to bottom right: Most similar point clouds (concerning pose and type) provided by the PCA optimization. The obtained poses better describes the orientation of the sample point cloud.

The PCA descriptor provides pose results which are used for the temporal optimization step. Since we know the partial point cloud which describes the shape and the pose of the detected object, any CAD model of the database can be places in the 3D layout.

3.3.6 Temporal Fusion and Pose Optimization

The main problem of the proposed approach is the sensibility regarding missing points and occlusion. The PCA optimization method leads to incorrect eigenvectors if many points are missing. Flipped normal vector directions are a further problem we are faced with (please see our experiment section for validations). Since we have a sequence of RGBD frames available, the objective of this section is to estimate more robust pose results by introducing a Markov Random Field (MRF) over the time.

The objective of the MRF is to obtain temporal consistent results for the already estimated poses over time. Therefore, the optimal pose at each node is determined and insufficient poses are eliminated. The idea of the MRF is illustrated in Figure 3.31. A temporal inference, in other words the most likely configuration, is calculated using a chain-structured MRF. The inference problem is solved by finding the Maximum A Posteriori Probability (MAP) configuration of the chain-structured graph [52].

An MRF $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is described by a set of vertices $\mathcal{V}$ and edges $\mathcal{E}$. In our case the vertices $\mathcal{V}$ correspond to the N images of the video sequence, and the edges $\mathcal{E}$ represent the transitions between subsequent frames. In the following we consider a set of random variables $\mathbf{X} = \{X_1, X_2, \dots X_N\}$ and a configuration of labels $\mathbf{x} = \{x_1, x_2, \dots x_N\}$ from a discrete set of labels $\mathcal{L}$. The energy function of the chain structured model is expressed by

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i=2}^{N} \psi_{i,i-1}(x_i, x_{i-1}), \tag{3.21}$$

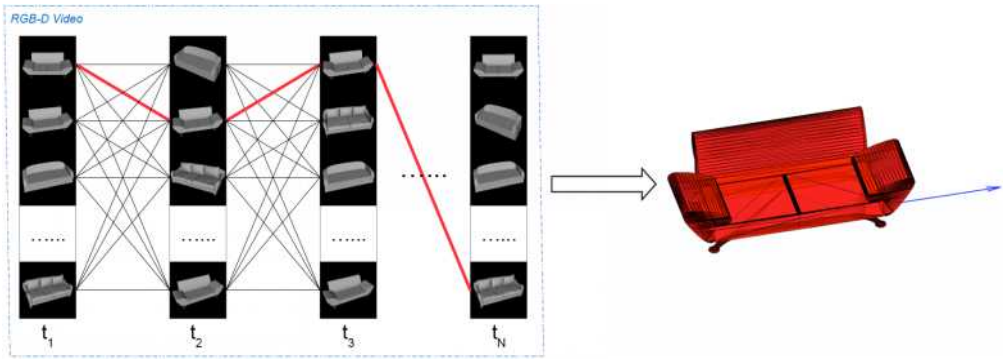


Figure 3.31: Temporal consistent poses are used to eliminate insufficient orientations. The optimal pose at each position of the sequence is estimated using a chain-structured MRF.

where ψ_i defines the *unary potentials*. The *smoothness term* between adjacent frames is expressed by $\psi_{i,i-1}$ (also known as *binary potentials*). The optimization of the MRF is determined by searching for the maximum joint probability in the graph [52]. This maximization is equal to minimizing the energy function defined in Equation 3.21. Consequently, the MAP configuration is calculated by

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}}\, E(\mathbf{x}). \tag{3.22}$$

Therefore, the estimated sequence delivers an optimized pose result at each time step in the video sequence.

For the calculation of the MAP configuration we use a dynamic programming algorithm (Viterbi algorithm [58]) provided by a MATLAB toolbox (*Undirected Graphical Models Toolbox*[8]). The runtime of the Viterbi algorithm is $\mathcal{O}(nNodes \cdot nStates^2)$ where $nNodes$ describes the number of vertices in the graph and $nStates$ the number of possible states.

In the following paragraphs we present how the unary terms and the smoothness terms are formulated.

Unary Potentials

Let us consider a partial point cloud m of the trained database and a point cloud p of the object in question at a frame i. Moreover, the corresponding orientation vectors provided by the PCA optimization method are expresses as $\mathbf{e}_m$ and $\mathbf{e}_p$ (cf. Figure 3.29). The unary potential at a certain frame i is expressed by measuring the angle between the orientation vectors. Mathematically the unary potentials are determined by

$$\psi_i = \frac{\arccos\left(\mathbf{e}_{p,i} \cdot \mathbf{e}_m\right)}{\pi}. \tag{3.23}$$

Since the range of the $arccos$ function is defined by $0 \leq \sphericalangle(\mathbf{e}_m, \mathbf{e}_p) \leq \pi$, the term results in values between $[0, 1]$, where low values denote similar orientation vectors. Orientation vectors which point in opposite directions result in high values.

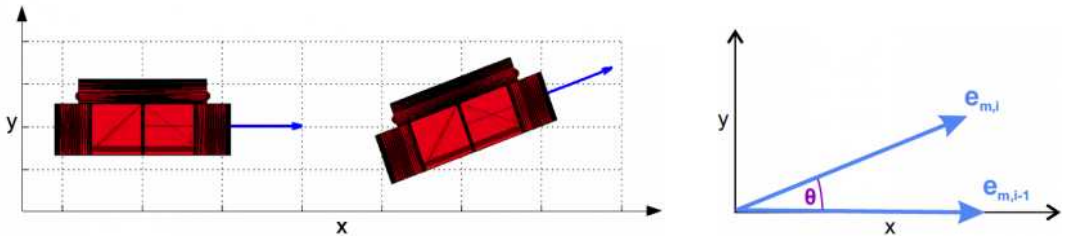
[8] `http://www.cs.ubc.ca/~schmidtm/Software/UGM.html` (last access: 17.04.2015)

Binary Potentials and Model Type Estimation

In order to achieve global correct poses, we assume that the poses between consecutive frames are not changing, because we are faced with overlapping frames and the camera's pose is inverse to the pose of the object. In other words, we try to find a configuration where the change between object poses in subsequent frames is minimized. Moreover, we assume that only frames of the same model type are considered as valid input. Frames in which no objects have been detected are ignored. The binary potential between adjacent frames is then expressed as

$$\psi_{i,i-1} = \frac{\arccos\left(\mathbf{e}_{m,i} \cdot \mathbf{e}_{m,i-1}\right)}{\pi}. \tag{3.24}$$

For each pair of frames $10 \cdot 10 = 100$ combinations are possible in our experiments. The proposed binary potential function returns low values for adjacent frames in which the objects' orientations are similar, and high values otherwise (cf. Figure 3.32).



(a) (b)

Figure 3.32: (a) The pose of each detected object is defined with one orientation vector. (b) The angles between the orientation vectors coming from subsequent frames are used to estimate the binary potentials. Objects detected in adjacent frames are supposed to provide similar orientation vectors.

Since the MRF returns an optimized model for each frame, we have to choose one model as representative because the final 3D layout only holds one CAD model for each detected object. An interesting possibility is the usage of the mean or the median pose over all estimated poses (orientation vectors). This method would focus on the estimation of correct orientations and shows promising results concerning this matter. Since we pay attention to the shape of the optimized detection results as well, our proposed method uses the object which delivers the highest similarity measure coming from the KNN classification of the VFH descriptor.

For placing the 3D model into the final 3D layout, we align the centroid of the partial point cloud and the centroid of the input point cloud for each object using the transformation information coming from the MRF. The point cloud of the object in question for our sample scene, the final detection result after the MRF optimization and the corresponding partial point cloud of the training stage are shown in Figure 3.33.

Since the VFH and the PCA descriptor are scale-invariant, in a final step a scaling between the determined 3D model and the object visible in the scene is required. Therefore, the bounding boxes of the detected object and the partial view are used. For the classes *chair* and *couch* the horizontal length of the bounding boxes is used to determine a scaling factor. For the class

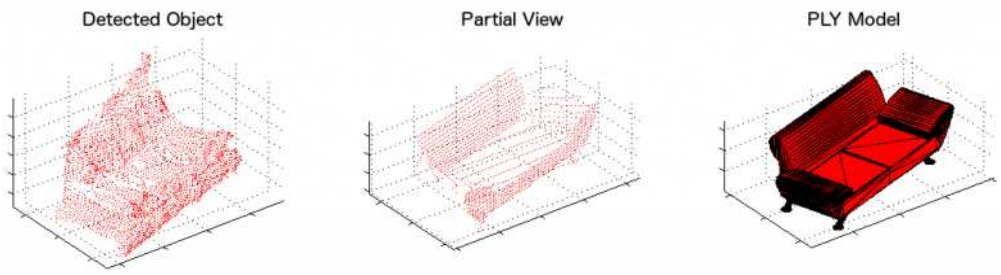


Figure 3.33: Point cloud of the object in question (left), the partial point cloud of the most similar model in the trained database (middle) and the corresponding CAD representation.

potted-plant we used the vertical directions of the bounding boxes. *Tables* are scaled based on the distance between the table surface (average maximum points) and the *ground plane*.

The finale 3D layout of our running example, together with the detected objects visible in the scene, is visualized in Figure 3.34.

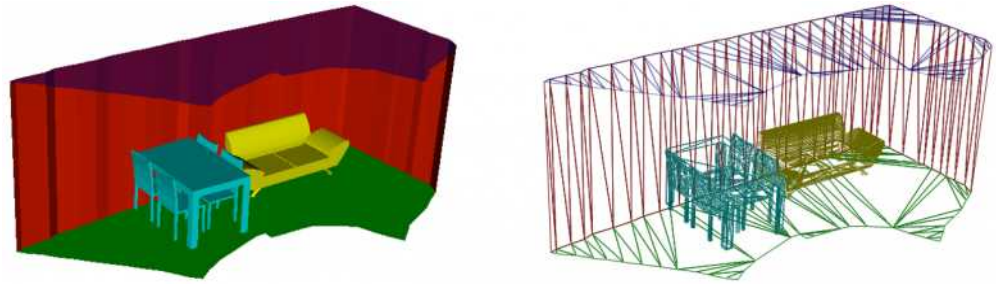


Figure 3.34: Fused 3D layout with the detected objects (determined by the MRF optimization) visible in the scene. The wire-frame model (right) of the 3D layout (left) clearly shows the date reduction compared to the dense input point cloud.

3.4 Discussion

We present a simplification framework for estimating semantically meaningful 3D layouts in which objects visible in the scene are detected and replaced by CAD models downloaded from a public database.

3.4.1 3D Layout Estimation

The proposed layout estimation pipeline estimates simplified 3D layouts by fitting 3D planes into the input point clouds. The layouts are extended by fusing multiple frames over time. Therefore, we exploit the multimodal characteristics of RGBD images. The advantage of our approach is that the complexity of the models is drastically reduced. Moreover, the framework can handle concave ground planes, which is an improvement concerning image-based approaches.

The main disadvantage of the proposed (frame-based) algorithm is that missing points lead to inaccurate layouts, especially if static images are used as input. Our proposed optimization method fuses layouts over time and leads to dense point-based representations. Hence, we are able to reduce the artefacts which are caused by missing parts in single point clouds. Compared to state-of-the-art 3D reconstruction approaches, we are still able to keep the storage demand on a minimum.

3.4.2 3D Object Detection and Pose Estimation

The multimodal data obtained from the Kinect sensor is used to estimate the type and the pose of the objects visible in a scene. In an initial step, object candidates are detected in the color image using a state-of-the-art object detector. The depth information within each detected object's bounding box is used to create an initial point cloud which represents the object in question. The advantage of using an additional 2D detector is that the detection results are not influenced by missing points.

A clustering approach is applied on each initial estimated point cloud in order to remove outliers and to get a dense and coherent point cloud of the detected object. Compared to recent 3D object detection frameworks, our approach is able to handle objects which are placed close to each other by using the additional 2D detection step.

We classify point clouds by comparing the VFH signature with trained signatures using a KNN search algorithm. Different to other works, this ranking is re-sorted using a PCA descriptor. Since we are using an additional 2D detector, we are able to reduce the search to one specific class.

A well-known problem of 3D detection approaches is their sensibility regarding missing points in the point clouds. We therefore introduce a chain-structured MRF which estimates temporal consistent results for the object's type and pose over the time.

Since objects in real-world scenarios are shown from a single viewpoint, parts of the point clouds are missing because of self occlusion. In order to be able to detect similar objects visible in certain frames, a database of viewpoint dependent point clouds is required. Therefore, we present a rendering approach which creates the relevant 2.5D point clouds and stores an additional pose information for each partial point cloud. Compared to recent recognition frameworks, this scalable approach reduces the time-consuming online training.

CHAPTER 4

Results and Experiments

The proposed framework supports the processing of single RGBD images as well as video sequences. Thus, the present chapter evaluates the framework based on single RGBD shots as well as multimodal videos. For this purpose qualitative and quantitative experiments are presented and further evaluated. After presenting the data sets and the CAD models used to train and evaluate the framework, we conduct with experimental results. The following issues are addressed:

- We present 3D layout results by processing single RGBD shots and highlight strengths and weaknesses. In order to overcome limitations caused by single shots, we outline the fused 3D layout results obtained from multiple RGBD frames and discuss improvements.

- Since an major objective of this thesis is the simplification of the fused 3D models, we show possible use cases and present results on the data reduction experiments.

- The object recognition and pose estimation experiments highlight the ability of our framework to handle cluttered and noisy point clouds, by introducing a MRF optimization approach.

For both evaluations (single shots and video sequences) we first describe the experimental setup. Secondly, qualitative and quantitative results are presented and discussed. All point clouds are visualized with the PCL *CloudViewer* and the corresponding 3D layouts (meshes) are illustrated with *matVTK*. To show the relevance of the data reduction, we further load and process the 3D layouts on a mobile device (Samsung Galaxy S3) with the *MeshLab*[1] 3D model viewer.

[1] `https://itunes.apple.com/at/app/meshlab-for-ios/id451944013?mt=8` (last access: 17.04.2015)

4.1 Data

The data set, which is used to train our framework, consists of partial 2.5D point clouds obtained from synthetic CAD models. Therefore, the CAD models are sampled by uniformly placing virtual cameras around the models. The data set for the evaluation of the recognition framework is composed of RGBD video sequences, providing multiple viewpoint-dependent point clouds. All point clouds are downsampled using a voxelized grid approach in order to provide a common resolution.

4.1.1 RGBD Shots and Videos

The RGBD videos used for the experiments are captured with a Microsoft Kinect sensor. For the evaluation of the single layout estimation method we use one frame pair (RGB and depth image) for each sample scene. For the temporal optimization experiments we capture RGBD videos. The sequences are first recorded and stored. Next, each RGBD image pair is incrementally processed by the proposed pipeline and the optimization methods are applied. The videos were taken in two different ways:

- Single position: The sensor is rotated around a fixed position (by hand) to grab a more complete area of the scene.
- Multiple positions: The videos are taken by a person walking through the scene (*e.g.* apartment).

All shots and videos are captured in typical indoor scenes (*e.g.* hotel room, living room, apartment, dining room). We take 21 single shots to evaluate the image-based layout estimation method. For the temporal optimization experiments we capture 10 video sequences which hold up to 50 frames. Examples for the single shots used are shown in Figure 4.4, 4.5 and 4.6.

RGBD images vs. video sequences The RGBD video sequences used in our experiments consist of consecutive and overlapping images. Our framework is able to estimate layout and object detection result for each single frame of the sequence. Thus, the pipeline for processing video sequences and single images is similar. The presented temporal optimization methods combine the results coming from multiple shots. The difference between single RGBD shots and multimodal videos is that the processing of single frames may lead to inaccurate results for the 3D layouts as well as for the object detection results. The main problem we are faced with are missing points caused by occlusion, illumination and specular surfaces. Missing points lead to the following limitations:

- The 3D planes estimated by the RANSAC plane-fitting method do not represent the walls in the scenes in a correct way. Thus, the corresponding lines in the 2D space are not crossing and consequently the resulting 3D layouts do not show smooth transitions between adjacent walls. Furthermore, if too much points of one wall segment are missing, the framework ignores the entire segment which leads to incorrect *floor plans*.

- Since the VFH descriptor is highly sensible to missing points, the signatures do not represent the shape and the viewpoint of the detected objects correctly. Consequently, the PCA optimization method provides poses which are not representing the correct orientation of the objects.

Both limitations are reduced by processing video sequences. Concerning the 3D layout estimation method overlapping RGBD frames coming from multiple shots handle the problem of non-crossing line segments. Even missing wall segments (in single frames) are no limitation for the fusion pipeline. Since we are faced with overlapping point clouds, we assume that the objects' poses between consecutive frames are not changing. The object detection results are optimized by combining the pose results coming from multiple frames.

4.1.2 CAD Models

As indoor scenes are most likely described by a few number of object classes, we limit our classification framework to 4 classes. For each class we train the framework with a different number of CAD models from Google's 3D Warehouse. For classes with a higher intra-class variability more models are trained. Table 4.1 lists the four object classes with the number of trained models. Figure 4.1 shows examples for the corresponding CAD models, rendered with Google *SketchUp*[2]. We use the method described in Section 3.3.1 to obtain point clouds and

Object class	Number of trained models
Couch	12
Chair	8
Table	8
Potted plant	5

Table 4.1: The four object classes used and the number of trained models.

partial views from the CAD models, in order to train our classifier with synthetically generated 2.5D point clouds. Since the VFH descriptor is invariant to scaling, there is no need to use different scale levels.

4.2 3D Layout Evaluation

This section outlines qualitative and quantitative layout results proposed by the presented layout estimation pipeline. We present experimental results by processing single RGBD frames as well as video sequences. Hence, we discuss advantages and disadvantages of the single layout estimation method, and show that multiple input frames are useful to overcome limitations caused by static images.

[2] `http://www.sketchup.com` (last access: 17.04.2015)

Figure 4.1: Sample CAD models from Google's 3D Warehouse which are used to train the framework (rendered from two different viewpoints).

4.2.1 Single Shot 3D Layout Evaluation

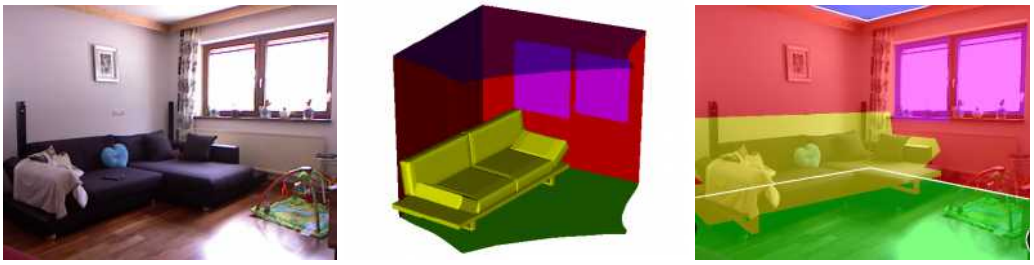
The proposed 3D layout estimation method generates mesh representations of single RGBD shots. The objective of such a representation is to create (1) a semantically meaningful and (2) a simplified mesh approximation of the corresponding point cloud with a reduced number of vertices and faces. Furthermore, missing areas and holes in the point clouds should be eliminated by re-building the missing information.

Qualitative Experiments

We compare the reconstructed point clouds of the sample scenes with the corresponding 3D layouts, provided by the proposed method. To show the relevance of our layout estimation framework, the results are compared with ground truth results and noisy ground truth results caused by missing parts in the point clouds/depth images. In addition we compare our method with two state-of-the-art frameworks namely the methods of Hedau *et al*. [29], and Choi *et al*. [10] which recover the spatial layout of indoor scenes from monocular RGB images. In [29] the room layout is estimated by fitting a 3D box into the scene. The work of [10] extends the framework of [29] by using additional geometric cues. In order to get ground truth data, we manually label the 21 test scenes using different colors as shown in Table 4.2. To compare our results with the labeled ground truth images, we project all triangles of the estimated 3D layout onto the corresponding RGB image and fill them with the corresponding label. Figure 4.2 shows an example of the layout projection and the resulting labeled image.

Color	Label
green	floor/ground points
red	wall points
blue	ceiling points
yellow/cyan	objects (couch, chair, table, potted-plant)
white	parts which are ignored (*e.g.* other objects)
black	missing values (caused by missing points)

Table 4.2: Different colors are used to distinguish between the relevant labels.



(a) (b) (c)

Figure 4.2: Projection of the estimated 3D layout on the raw RGB image. (a) Raw RGB image. (b) Proposed 3D layout. (c) Labeled image in which the white lines indicate the 3D layout.

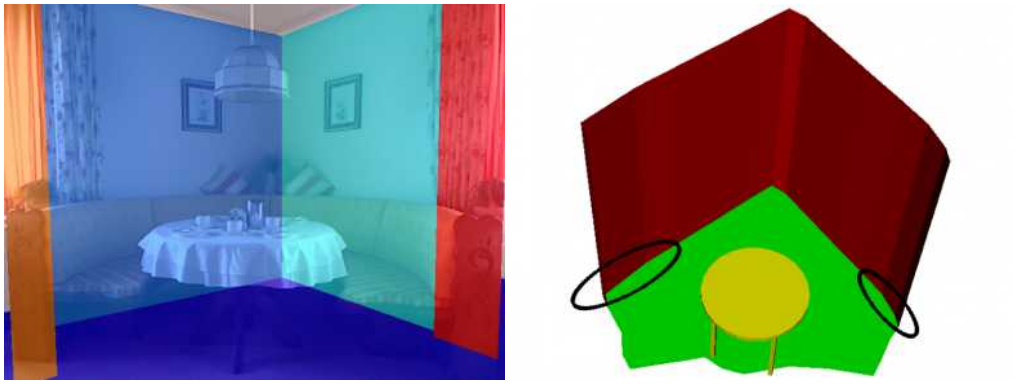
Qualitative Results

Figures 4.4, 4.5 and 4.6 show the 21 test scenes used for the evaluations, the reconstructed point clouds and the estimated 3D room layouts with the detected objects.

The qualitative results show that the calculated 3D layouts describe the dimensions of the captured point clouds. Walls are represented by planar faces with smooth transitions. The majority of the visible walls in the scenes are represented by one planar surface. Nevertheless, displacements or uneven transitions between adjacent walls are possible (*e.g.* Figure 4.6 scene 16), which indicates that two or more walls are detected instead of one single wall (*e.g.* because of texture and illumination changes). Figure 4.3 shows an example for the mentioned problem. On the other hand the example also shows the ability of the proposed framework to reconstruct walls although notable parts of the layout are occluded (for example the leftmost and rightmost walls in the scene).

The qualitative results show the ability of the proposed framework to deal with cluttered scenes in which parts of the room layout are missing or occluded by objects. Although a numerous of scenes (*e.g.* scene 11, 13, 16, 17, 18, 19, 20) show a considerable amount of clutter, the framework still estimates the ground floor and the walls in a correct way. An example is illustrated in Figure 4.7, where a reasonable amount of the points of the right wall are missing and a notable part of the left wall is occluded. Nevertheless, the framework is still able to estimate a correct room layout and rebuild the missing information.

In individual test scenes (*e.g.* scene 4) our algorithm is not able to replace the 3D points of



(a) (b)

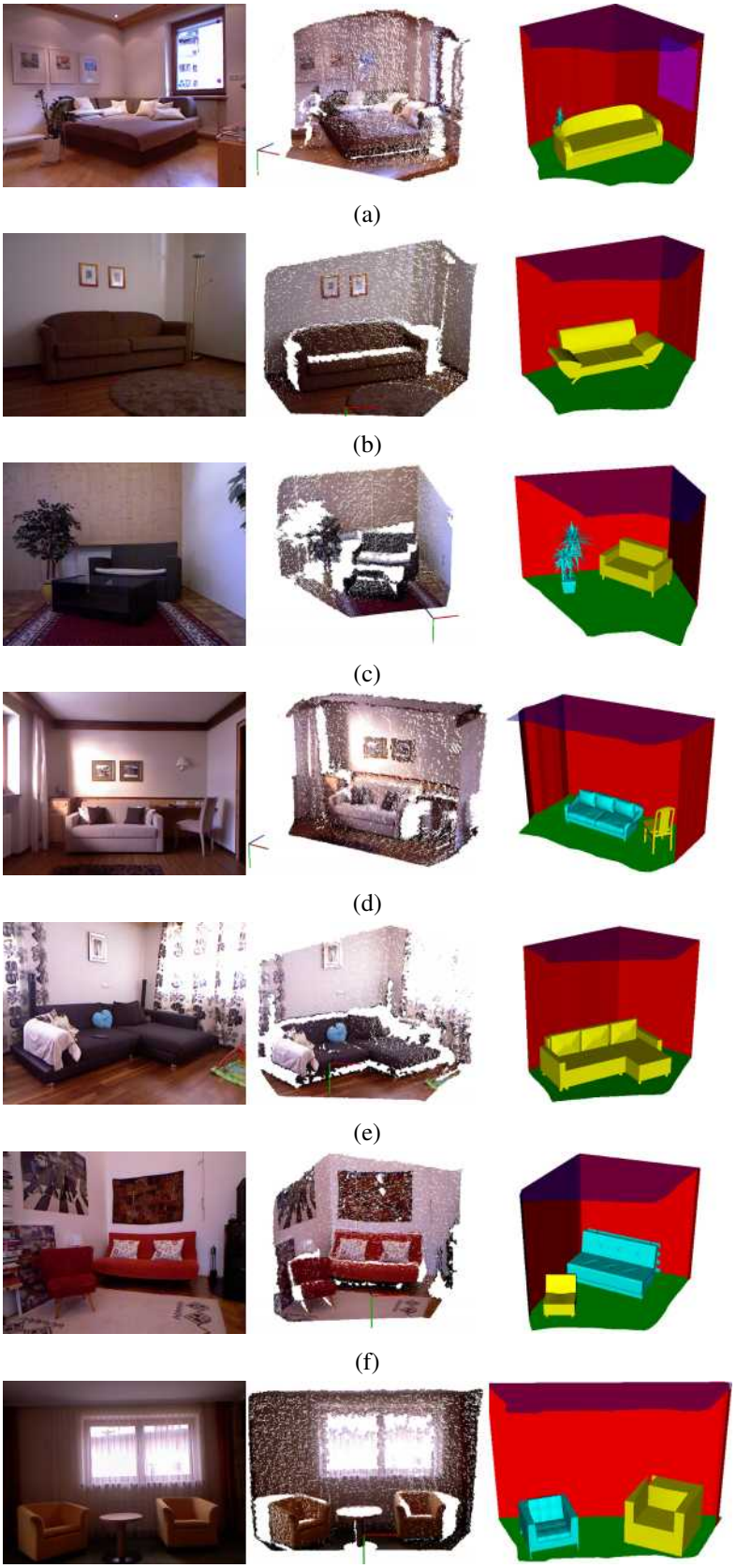
Figure 4.3: (a) The initially detected wall candidates in the RGB image. (b) The resulting 3D layout shows unsmooth transitions between the adjacent walls. Although the walls are occluded by objects, the framework still estimates useful layouts.

the initially detected wall candidate by a smooth surface because too much information is missing. The mentioned issue is illustrated in Figure 4.8, where parts of the wall are occluded by the curtain and an object. Furthermore, the direct sunlight leads to additional clutter. The problem of missing walls could be reduced by applying the before mentioned temporal optimization method, in which several scenes are fused in order to estimate a denser point cloud.

Figures 4.9, 4.10 and 4.11 show the labeled results for the 21 sample scenes (from left to right: ground truth image, noisy ground truth image with missing values, two state-of-the art frameworks – namely the algorithms of Hedau *et al*. [29] and Choi *et al*. [10] – and our proposed 3D layout. Please note that the images visualize the 3D layouts as well as the detected objects. The empty layouts are indicated by the white lines (column 4, 5 and 6).

The results of Hedau. et at. [29] show limitations concerning the correct estimation of the room layouts (compare Figure 4.9, 4.10,4.11 column 4). For example the scenes 1, 2, 3, 7, 8, 11, 12, 13, 15, 17 and 20 show insufficient results for the ground plane. In a numerous of test samples parts of the wall are labeled as ground plane (*e.g.* scene 1, 8, 12, 15, 17, 20). In the remaining scenes (scene 2, 3, 7, 11, 13) the walls hide parts of the ground plane. The additional geometric cues introduced by Choi *et al*. [10] reduce the problem of incorrect labeled wall segments. Nevertheless, scenes where only one wall is dominant are challenging (*e.g.* scene 7, 12, 15, 17). Concerning this fact, our proposed layouts provide more accurate results. The two state-of-the-art methods fit a 3D box into the scene. Therefore, the algorithm works well for scenes where three walls (left, middle, right) are visible (*e.g.* scene 2 and 10). Consequently, the framework is not able to handle concave scenes. Scene 2 and scene 13 are examples for rooms with concave ground planes. Hence, our proposed framework outperforms the two state-of-the-art methods specially if non-convex scenes are present. This is a significant improvement concerning monocular methods which are based on a three-point perspective.

The results also show that the state-of-the-art frameworks have problems to align the 3D box exactly with the dominant scene orientations. The layouts provided by Hedau *et al*. [29]) show



(a)

(b)

(c)

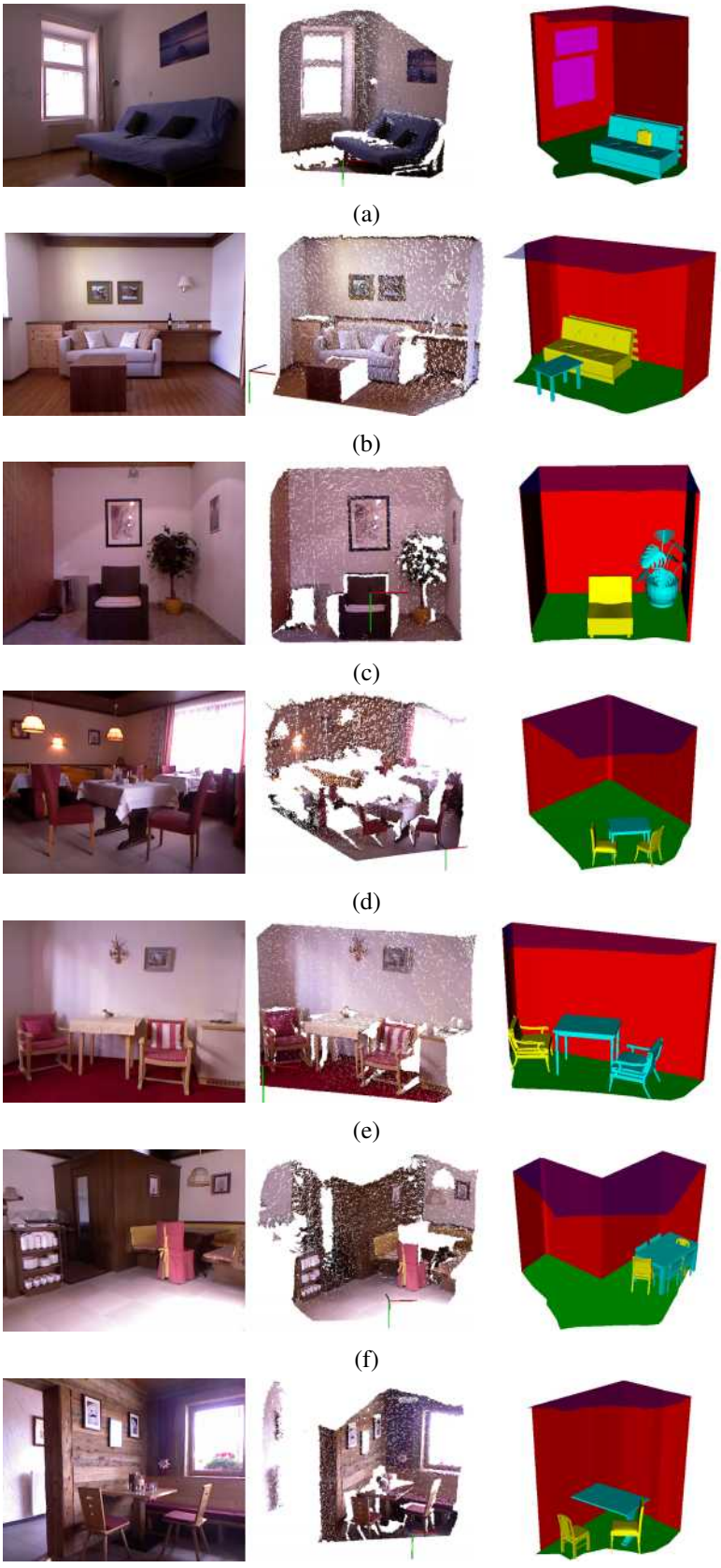
(d)

(e)

(f)

(g)

Figure 4.4: Qualitative results for the sample scenes (1-7). Each column shows the RGB image (left), the reconstructed point cloud (middle) and the proposed 3D layout (right).



(a)

(b)

(c)

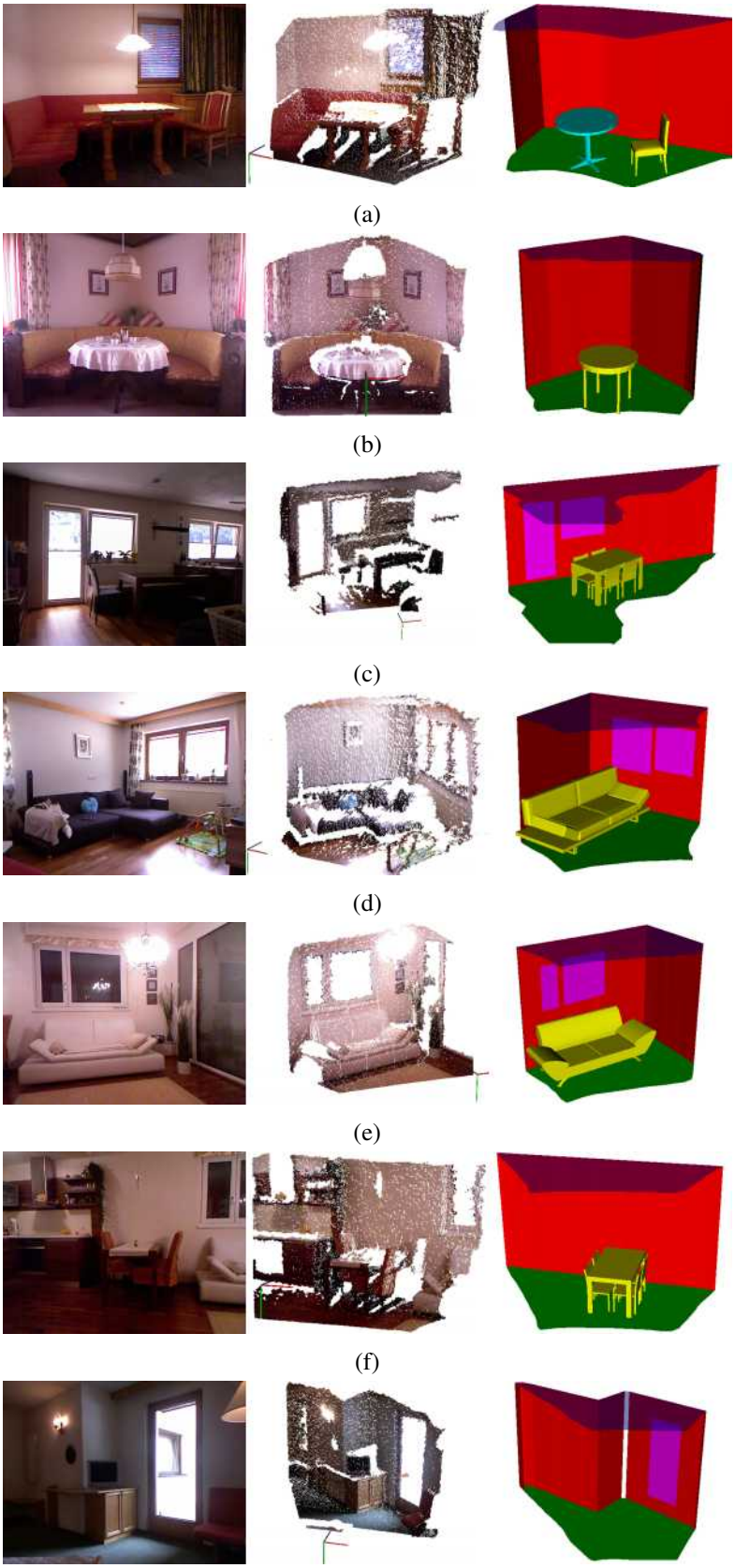
(d)

(e)

(f)

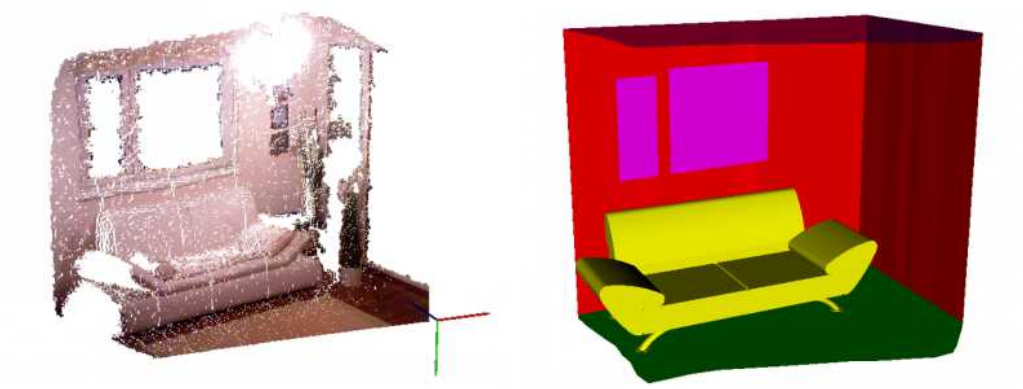
(g)

Figure 4.5: Qualitative results for the sample scenes (8-14). Each column shows the RGB image (left), the reconstructed point cloud (middle) and the proposed 3D layout (right).



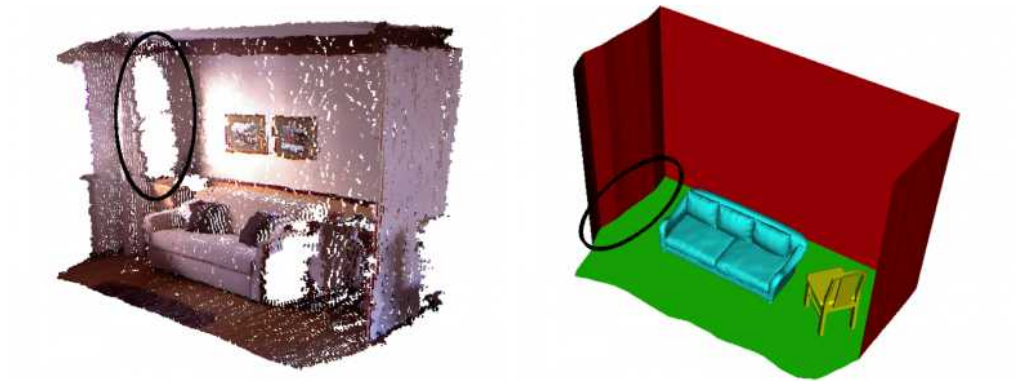
(a) (b) (c) (d) (e) (f) (g)

Figure 4.6: Qualitative results for the sample scenes (15-21). Each column shows the RGB image (left), the reconstructed point cloud (middle) and the proposed 3D layout (right).



(a) (b)

Figure 4.7: Comparison between point cloud and 3D layout. (a) The scene shows a considerable amount of clutter and missing values. (b) The proposed framework reconstructs the walls and provides a correct 3D layout in which the two visible walls are represented by two smooth planes.



(a) Cluttered point cloud caused by occlusions (b) Insufficient 3D layout

Figure 4.8: (a) Large parts of the point cloud are missing and occluded. (b) The leftmost wall in the scene is reconstructed incorrectly.

that in numerous scenes the ceiling is missing (*e.g.* scene 1, 11, 17, 21) or labelled incorrect (*e.g.* scene 7, 8, 18). In scenes where the ceiling is visible in the ground truth image (scene 4, 7, 17, 18), our proposed framework provides more accurate results compared to both monocular methods.

A further goal of our framework is to interpolate missing parts in the point clouds. In the labeled images the missing parts are highlighted with black regions. In numerous examples (scene 5, 7, 8, 11, 14, 17, 18, 19, 20, 21) huge regions are missing because of the already mentioned point cloud artefacts. Nevertheless, the proposed framework is able to handle noisy scenes and restore the missing parts. Our final 3D models are watertight, as this is an required attribute concerning further post-processing steps [39, 66] (*e.g.* loading the models into a third-

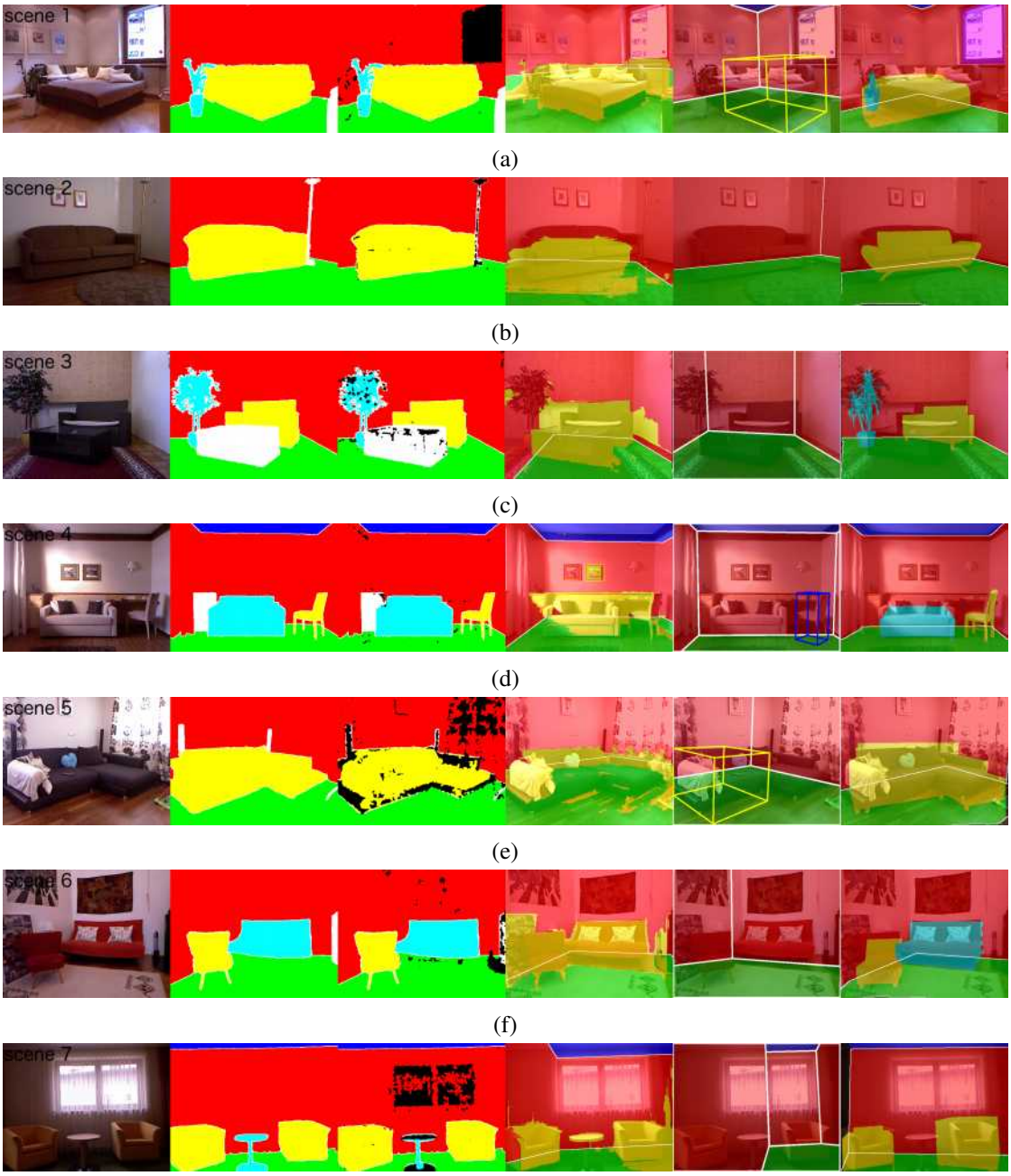


Figure 4.9: (a-g) Comparison of the labeling results for the test scenes 1 - 7. From left to right: Raw RGB image, ground truth image, noisy ground truth image, Hedau *et al*. [29], Choi *et al*. [10], proposed 3D layout.

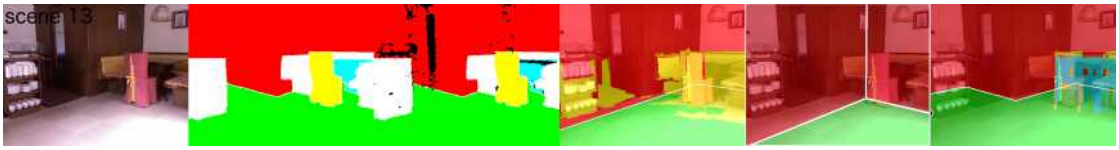
(a)

(b)

(c)

(d)

(e)



(f)

(g)

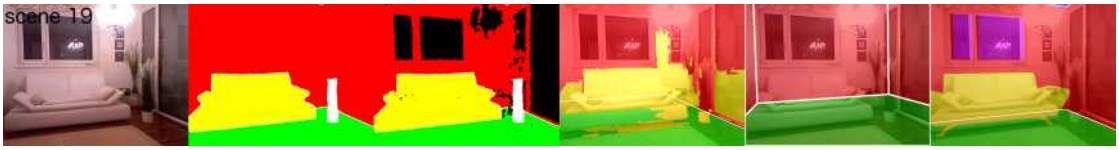
Figure 4.10: (a-g) Comparison of the labeling results for the test scenes 8 - 14. From left to right: Raw RGB image, ground truth image, noisy ground truth image, Hedau *et al*. [29], Choi *et al*. [10], proposed 3D layout.

(a)

(b)

(c)

(d)



(e)

(f)

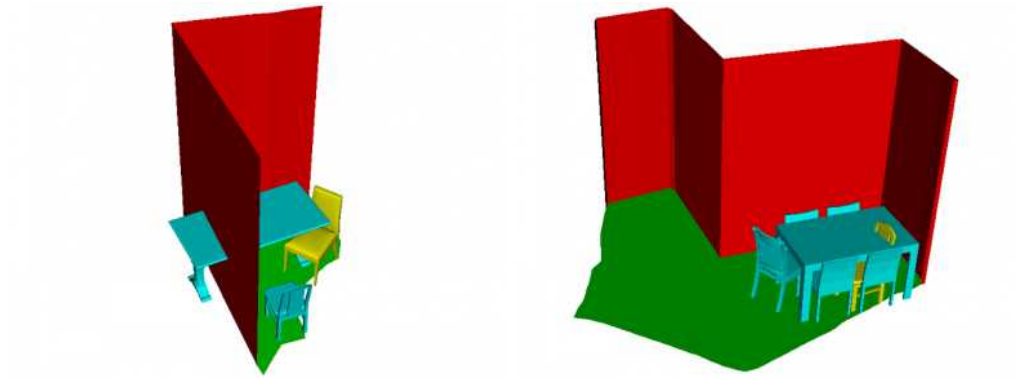
(g)

Figure 4.11: (a-g) Comparison of the labeling results for the test scenes 15 - 21. From left to right: Raw RGB image, ground truth image, noisy ground truth image, Hedau *et al*. [29], Choi *et al*. [10], proposed 3D layout.

party software or 3D printing).

The previously presented results also show that our framework is able to detect objects in the scenes. Compared to the pixel-based results of the state-of-the-art frameworks, the proposed results use 3D models. While most scenes show suitable pose estimation results, other scenes (*e.g.* scene 1, 6, 7, 8, 12, 14) come along with insufficient pose results (deviation between original and detected object is greater than 20 degrees). A more detailed evaluation is presented in Section 4.4. In order to estimate semantically meaningful layouts, we described a simple depth-based method to detect windows as well. In the result images the detected windows are marked *violet* (cf. Figure 4.9, 4.10 and 4.11).

Limitations Although our framework provides visually sufficient results, we are faced with problems concerning the geometric configuration. In numerous test scenes (*e.g.* scene 11, 13, 14) detected objects are overlapping each other, or multiple objects are detected at the same position (*e.g.* scene 8 and 13). For example in scene 13 a *chair* is detected although a *table* was already detected at the same place (cf. Figure 4.12b). Another problem occurs within the relation of the detected objects and the 3D layouts. Our detection method scales objects along one direction (horizontal direction for the class *chair* and *couch*, vertical direction for the class *table* and *potted-plant*), to get proportionally correct results. Consequently, it may happen that a object reaches beyond the 3D layout. An example (scene 13) is illustrated in Figure 4.12a).



(a) (b)

Figure 4.12: Incorrect geometric configuration. (a) A detected object reaches beyond the wall. (b) Two (or more) objects are detected at the same position.

Quantitative Experiments

This section presents quantitative experiments. Having a ground truth image F and a result image R enables the calculation of a pixel accuracy $\frac{|G_l \cap R_l|}{|G_l \cup R_l|}$ for an given object class $l \in \mathcal{L} = \{groud\ plane,\ wall,\ ceiling\}$ over all sample images. Equation 4.1 defines the calculation of the pixel accuracy A_{cc}, in which N_{ii} determines the number of correct labeled pixels. N_{ij}

determines the number of pixels of label i labelled j.

$$A_{cc} = \frac{N_{ii}}{\sum_j N_{ij}}, \forall i, j \in \mathcal{L}. \tag{4.1}$$

Furthermore, we calculate the *average error* and the *global error* over all object classes and test images. In the following experiments we compare our result again with the ground truth image, the noisy ground truth images (raw depth) and the results of [29] and [10]. Pleas note that the method of [10] does not return object labels.

Quantitative Results

We perform two kinds of quantitative experiments. The result of the first experiment describes the comparison of the empty 3D layouts. In the second experiment we compare the 3D layouts taking into accounts the four object classes. Table 4.3 and Table 4.4 list the number of correctly classified pixels for each object class and compares the addressed methods.

	Raw depth	Hedau *et al.* [29]	Choi *et al.* [10]	Proposed layout
ground plane	**94.14**	89.07	87.48	90.74
walls	88.58	84.82	91.65	**96.44**
ceiling	**82.75**	32.08	51.69	69.22
mean	90.03	84.91	86.70	**95.03**
global	91.25	81.65	90.62	**91.38**

Table 4.3: Percentage of correctly classified pixels for each label (empty 3D layouts).

	Raw depth	Hedau *et al.* [29]	Choi *et al.* [10]	Proposed layout
ground plane	**94.74**	64.14	/	80.39
walls	88.23	84.97	/	**96.47**
ceiling	**82.75**	32.03	/	75.89
mean	**90.54**	75.25	/	88.10
global	**91.35**	70.84	/	87.31

Table 4.4: Percentage of correctly classified pixels for each label (3D layouts and objects).

The pixel accuracies confirm the already presented qualitative results and statements presented in the previous section. Both experiments show that our framework outperforms the monocular methods for all labels (*ground plane*, *wall*, *ceiling*). In both experiments the highest accuracy rates (proposed by our method) are reached for the label *wall*. For the empty layouts we achieve a mean classification rate of 95% which outperforms the noisy ground truth results by approximately 5%. This indicates that our algorithm successfully reproduces missing areas in the raw clouds.

In the second experiment we take into account the four object classes (*couch*, *chair*, *table*, *potted-plant*). Again, the best classification rate is achieved for the label *wall*. This value is again

greater than the noisy ground truth rate. Compared to the empty layouts, our pixel accuracies have deteriorated slightly. We would like to mention that among others (*i.e.* classification error) this is also caused by the formulation of our detection scheme. The detected objects in our framework are only scaled along one direction. Thus, we get a higher error because regions of the *wall* ore the *ground plane* may be labeled incorrect.

4.2.2 Multiple RGBD Shots Evaluation

This section outlines the 3D layout results provided by the proposed temporal fusion pipeline. The objective is to create extended and more complete 3D layouts while still keeping the memory load low. The following results show that the limitations caused by missing points are reduced. The memory experiments are presented in Section 4.3.

Experimental Setup

In order to create geometrically exact and extended 3D models, we capture several shots from different viewpoints and fuse them into a single model. The method described in Section 4.2.2 is used to create the extended 3D layouts. In the following, qualitative and quantitative results as well as the improvements achieved by the presented optimization method are presented.

Qualitative Results

Figure 4.13 and 4.14 show the test scenes used (fused point clouds) and the fused 3D layouts. The results show the ability of our framework to fuse several shots, while still providing simplified mesh-based models.

Our proposed framework provides simplified approximations of the fused point clouds. The floor, the walls and the ceiling are represented by smooth planes. The framework is able to rebuild all visible walls of the test scenes. Figure 4.15b shows the mentioned problem (single case) that walls are not reconstructed correctly because too much information is missing. The example shows that parts of the leftmost wall are occluded by the *curtain* and a *box*, which leads to the missing wall in the final 3D layout. The direct sunlight at the left wall is an additional problem. By using the proposed optimization method, the wall is reconstructed in a correct way, because more information is available after the point cloud registration (cf. Figure 4.15c).

An further improvement of our framework compared to image-based methods is that non-convex ground planes can be processed. This advantage is still available after several frames have been fused to a final model. An example for a fused and concave scene is illustrated in Figure 4.14 (video 6).

The 10 resulting 3D layouts also show that the fusion is geometrically exact because the walls merge smoothly. Nevertheless, single scenes (*e.g.* video 4 and video 9) show limitations concerning the transitions of the shots (cf. Figure4.16). One reason for this artefact is that too much SIFT feature are missing which leads to an inaccurate transformation matrix. Nevertheless, the framework still provides closed 3D models.

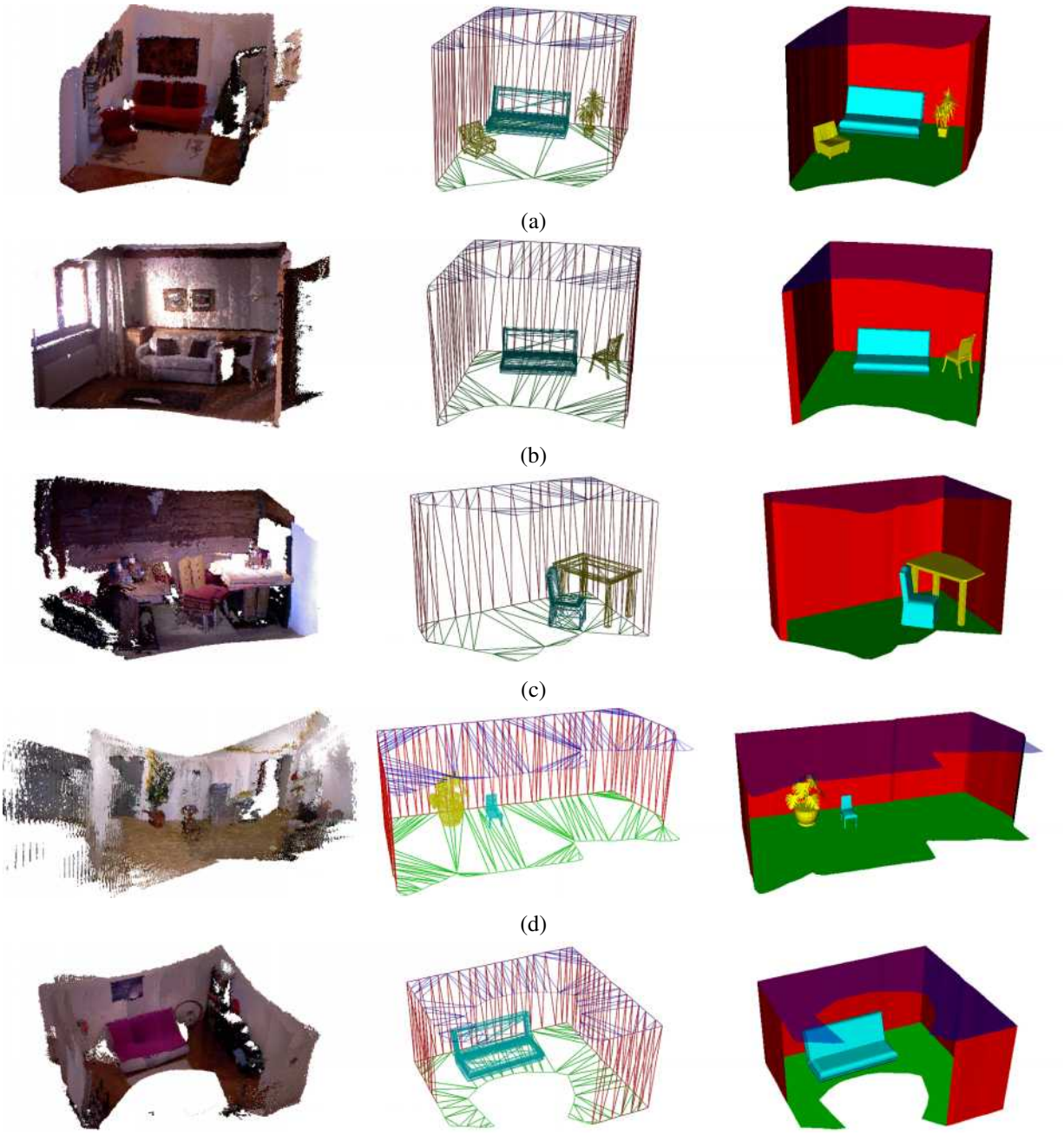


Figure 4.13: Optimized results for the sample videos (1-5). A RGBD video sequence is used to create a dense point cloud (column 1). The proposed framework estimates a mesh-based, smooth and simplified model of the scene with a limited number of vertices and faces (column 2 and 3). Each sample shows the MRF optimized object classification result.

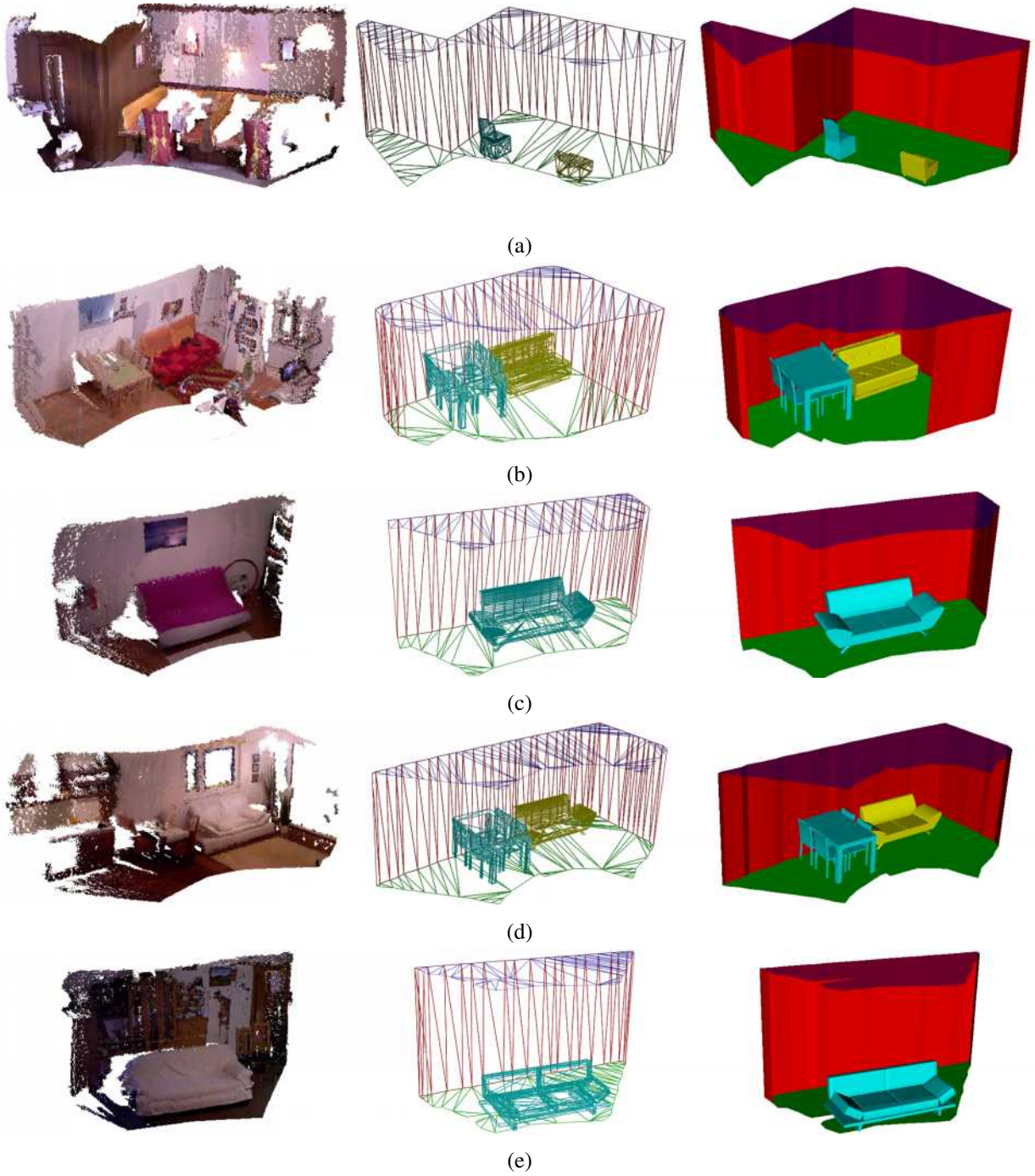
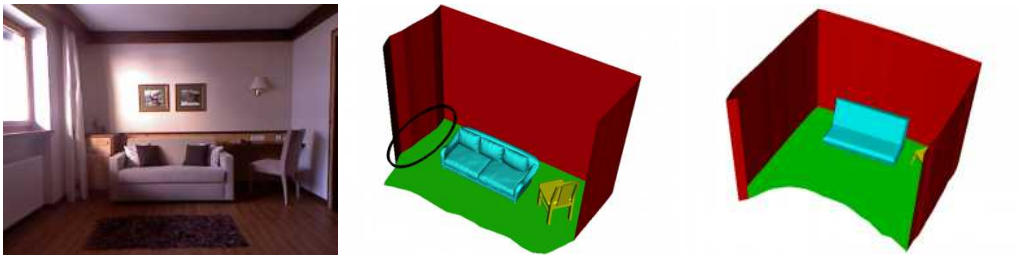
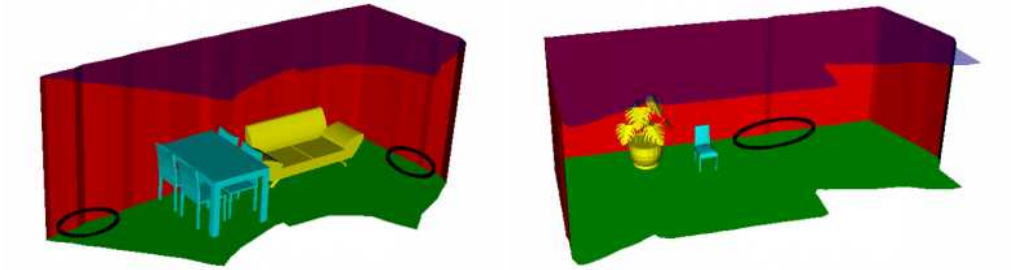


Figure 4.14: Optimized results for the sample videos (5-10). A RGBD video sequence is used to create a dense point cloud (column 1). The proposed framework estimates a mesh-based and simplified model of the scene with a limited number of vertices and faces (column 2 and 3). Each sample shows the MRF optimized object classification result.



(a) (b) (c)

Figure 4.15: Wall artefacts due to missing points caused by direct sunlight and specular surfaces. (a) RGB input scene. (b) The processing of one single frame leads to an incomplete layout. (c) The temporal optimization method provides a more complete and correct 3D model.



(a) (b)

Figure 4.16: The resulting 3D layouts (video 4 and 9) show an offset between adjacent shots. The mesh model is still closed.

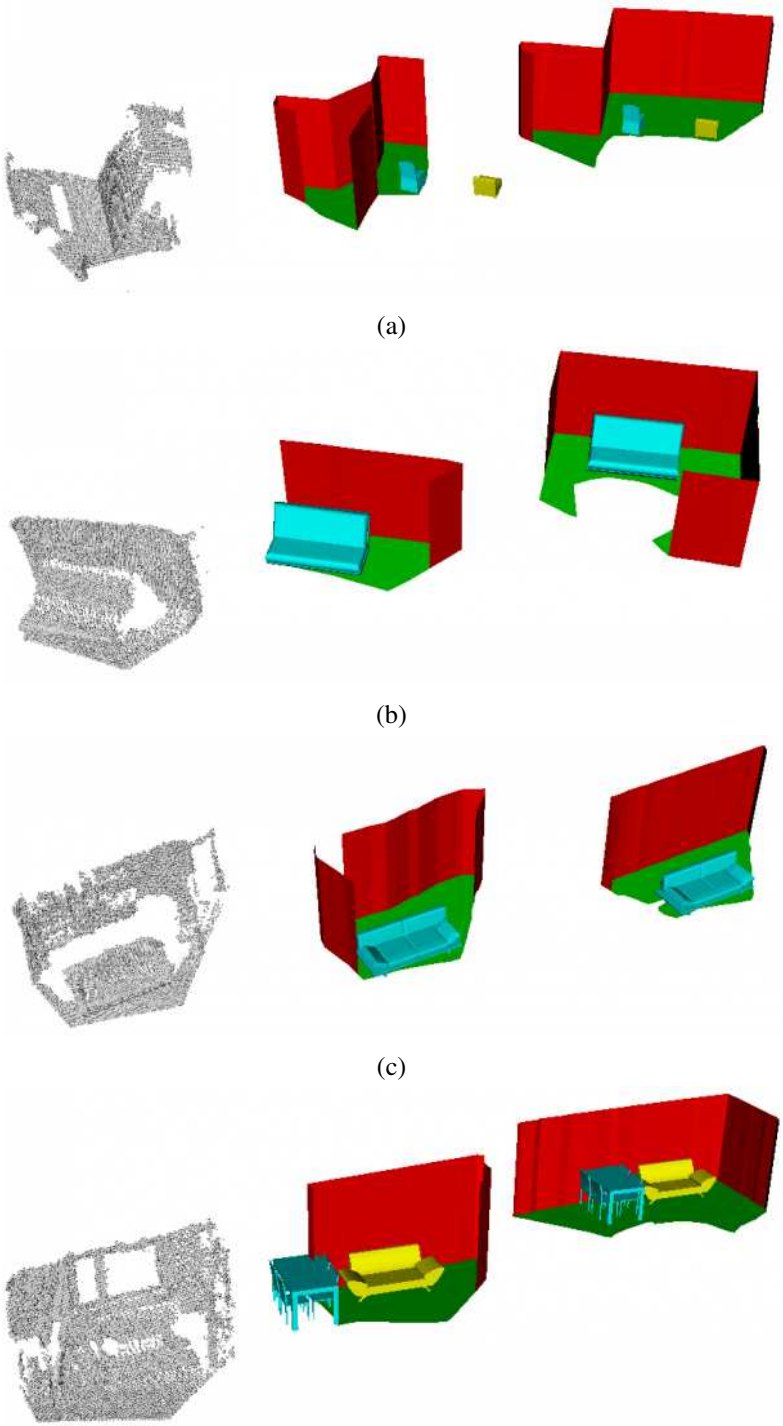
Comparison of Single Layouts and Fused Layouts When using single images missing points lead to inaccurate plane-fitting results and consequently to inaccurate or even missing line segments. In the following we show that the optimized 3D layouts generated by the optimization approach reduces these artefacts. Figure 4.17 shows single point clouds (column 1), the corresponding single 3D layouts (column 2) and the fused 3D layouts provided by processing the entire video sequences (column 3).

As can be seen in the illustrations, the following improvements are achieved:

- Redundant wall segments (at the exterior edges) which are caused by the convex hull approach are mostly eliminated.
- Unsmooth walls are replaced by planar segments.
- Inaccurate transitions between consecutive wall segments are corrected.

Quantitative Results

This sections demonstrates that the fused 3D layouts lead to an improvement concerning the scene dimensions. Therefore, we first create ground truth wall segments by manually extracting



(a)

(b)

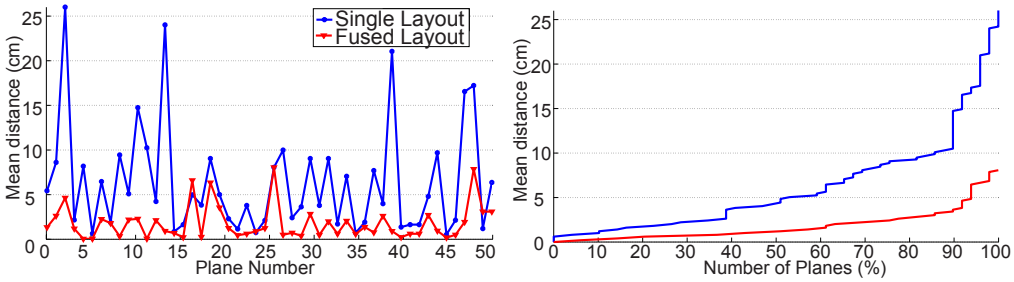
(c)

(d)

Figure 4.17: 3D Layout improvements. Left: Raw point clouds. Middle: Corresponding single 3D layouts coming from one frame of the video sequence. Right: Fused 3D layouts provided by multiple shots.

all walls from the fused point clouds. Furthermore, we remove points which do not belong to the walls (*e.g.* objects). The remaining point sets describe the ground truth wall. Next we calculate the average distance of all these points to the closest underlying wall segment of our proposed 3D layout coming from (1) the fused 3D layout and (2) single layouts coming form the registration step.

Random frames are taken from every video sequence and the distances over all planes and shots are estimated. Figure 4.18 shows (a) the mean distances in centimetres between the ground truth points and the estimated RANSAC planes and (b) the mean distances of the ground truth points to the RANSAC planes for the corresponding percentage of planes over all shots and videos.



(a) (b)

Figure 4.18: Comparison of the estimated 3D layouts when using single layouts separately and when using the fused and optimized layouts.

The comparison shows that for 92% of all wall segments (planes), the distance between the ground truth points and the room layout is less than 5cm when using the optimized layouts. When using the single 3D layouts instead, the same percentage of planes lead to an mean distance of 17cm. This indicates that an increasing amount of input frames leads to more exact 3D layouts.

4.3 Memory Capacity, Data Reduction ans Use Cases

In the previous evaluation we have seen that our framework is able to fuse several RGBD shot over the time. Similar works like the *KinectFusion* application [39] create a dense and more detailed 3D reconstruction of indoor scenes. The problem of such solutions is that third-party applications are unable to use the 3D models efficiently, because of their geometric complexity and huge amount of data [39, 66]. When fusing our scenes using KinectFusion, the filesize may increase up to 300 MB, which can not be manipulated (*e.g.* in a CAD software) in real-time anymore.

In general, the computational cost of processing a 3D model, respectively a point cloud, relates directly to its complexity [25]. Thus, small 3D models with limited complexity are required. The following evaluation shows the ability of our framework to provide fused 3D models, while keeping the memory load low and constant.

Experimental Setup

In the following we compare the number of vertices/faces of the proposed 3D layouts with the number of raw points coming from the fused point clouds. In addition, the file size of the raw point clouds and the stored mesh representations (PLY output) is compared. To show the relevance of the data reduction, we further load the final 3D layouts on a mobile device (*Samsung Galaxy S3*). Moreover, we show a possible use case in which our 3D layouts are useful – namely a *Google SketchUp Scene Composer*.

Quantitative Results

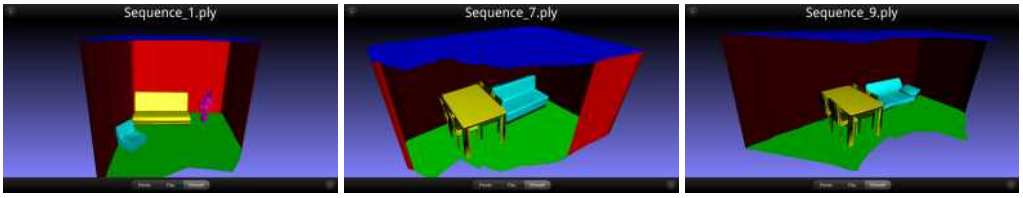
Table 4.5 compares the test scenes concerning the file size and the number of points/vertices. The values show that the proposed 3D layouts consist of a drastically reduced number of vertices. While the saved point clouds are memory intense (up to 230 MB), the corresponding mesh representations only require memory storage in the range of 10 - 300 kB. Hence, our proposed method leads to a data reduction of over 99.9% concerning the number of points/vertices of the models as well as for the corresponding memory demand.

Scene	Size		Number of Points			
	Input[MB]	Ours [KB]	Input	Ours [Layout/Obj./Tot.]		
1	137.98	465	3 379 200	260	7 412	7 672
2	113.46	59	2 764 800	270	756	1 026
3	75.14	100	1 843 200	205	1 337	1 542
4	228.89	331	5 529 600	520	4 776	5 296
5	174.78	29	4 300 800	380	220	600
6	88.18	104	2 150 400	320	1 355	1 675
7	164.34	58	3 993 600	295	732	1 017
8	62.74	171	1 536 000	260	253	2 513
9	163.50	196	3 543 200	360	2 565	2 925
10	74.85	41	1 843 200	265	440	705

Table 4.5: Comparison of the memory consumption and the number of 3D points/vertices for the fused input cloud and the proposed 3D layout (number of points split up in layout, objects, total) .

Qualitative Results

Since our simplified 3D models require only a limited amount of storage, we are able to visualize the layouts on a mobile device. Figure 4.19 shows three models (video 1, 7 and 9) which are displayed on a *Samsung Galaxy S3*. The resulting models are loaded from a web browser. It is further possible to manipulate (*i.e.* drag, pan, rotate, zoom) the models in real-time.



(a) (b) (c)

Figure 4.19: Three fused 3D models loaded and manipulated in real time on a mobile device (*iPhone 4s*).

Use Case - Google SketchUp Scene Composer

Since the 3D layouts which are generated by our framework are stored in a standard three-dimensional data format (PLY, OBJ), we are able to load the entire model into a third-party CAD software. In the following we demonstrate a possible use case for the output of the framework: The 3D layouts are visualized in *Google SketchUp* and new models (*e.g.* furnitures) are placed into the 3D layout. The 3D layouts are perpendicular to the coordinate system of the software which helps to place models straight on the ground plane. The *new* models are downloaded from the *Google 3D Warehouse* database. Figure 4.20 shows images of the proposed use case.

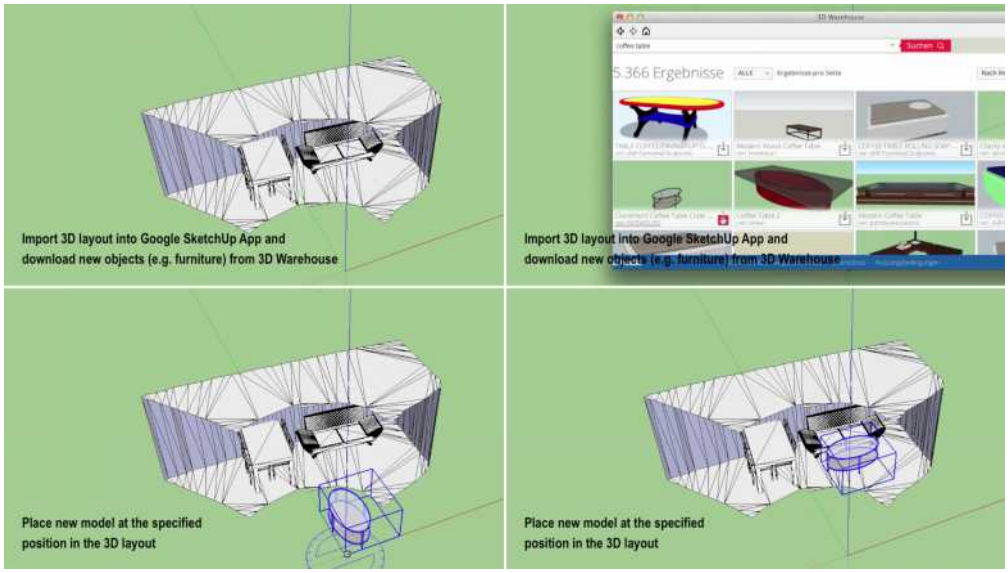


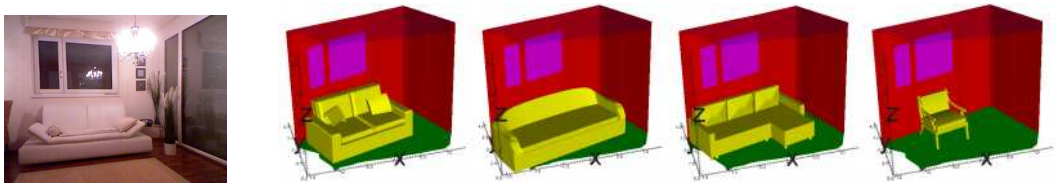
Figure 4.20: Use case – Scene Composer. From top left to bottom right: (a) 3D layout is imported into SketchUp. (b) New 3D model is downloaded from a web database. (c) Downloaded model is rotated and shifted. (d) Finale composed 3D layout with the new model on the correct position.

4.4 3D Object Classification Evaluation

This section presents the evaluation of the presented object detection and pose estimation approach. First, the results obtained from processing single RGBD shots are discussed. Therefore, we compare the detection results coming from the VFH descriptor and the optimized VFH+PCA descriptor. Secondly, we evaluate the MRF-based optimization method and show that the usage of video sequences and the temporal optimization lead to more robust pose results. Since the results for the type of the detected objects are almost exclusively based on the state-of-the-art 2D object detector, in the following experiments we focus on the evaluation of the pose results.

4.4.1 Single 3D Object Classification Evaluation

A main advantage of the framework architecture is that for each detected object information about the class type, the pose and the spatial position in the scene is available. This allows the replacement of the detected object in the final 3D scene with a random CAD model of the trained database. Such a scenario is a possible use case for augmented reality applications. Figure 4.21 illustrates the mentioned idea.



(a) (b)

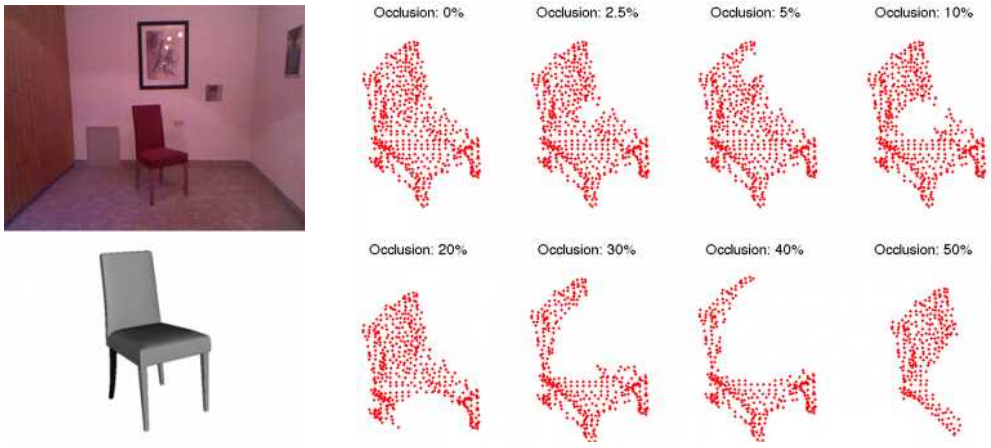
Figure 4.21: 3D model replacement. (a) Input Scene. (b) Different CAD models from the trained database are placed into the 3D layout.

The qualitative results of the single 3D layout estimation experiments have shown that the proposed framework provides accurate poses for most of the test scenes. Nevertheless, inaccurate poses are possible because of missing points which lead to VFH signatures [3,56,57] which do not represent the entire shape and the correct viewpoint of the detected objects. Consequently, the PCA orientation vectors do not correspond with the correct orientation vectors. We now take a closer look on how occlusion affects on the detection and orientation results.

Experimental Setup

In this experiment we analyse the performance of the VHF descriptor and the combination of VFH+PCA when the input point cloud is noisy or incomplete. Therefore, we place a *chair* in a test setting. The *chair* is not occluded by other objects, the illumination is constant and not changing and the the Kinect sensor is placed on a fixed position which means that the viewpoint is not changing. Figure 4.22a illustrates the configuration and shows the object used for the experiments. Next, the *chair* is uniformly rotated $(0 - 180^{\circ})$ and for each orientation the classification is performed, using the VFH descriptor as well as the combined VFH+PCA descriptor.

In this experiment only the most similar object from the trained database is used. For each orientation we then simulate occlusion by randomly removing a defined percentage of connected points from the object's point cloud. Figure 4.22b shows the simulation of the occlusion for different percentages of occlusion.



(a) (b)

Figure 4.22: (a) Sample input image and the 3D model used for the classification. (b) Simulation of occlusion.

For each orientation and amount of occlusion the pose and an the resulting pose error are determined. The pose error describes the angle between the ground truth orientation vector of the detected object and the vector coming from the classification. Angles between $(0 - 180°)$ are possible, in which $0°$ describes the optimal pose result. For each shot 30 occlusions are simulated in order to handle different shape variations caused by the occlusion simulation, and the median pose error is estimated for each orientation.

Quantitative Results

The classification results obtained from the previous presented occlusion experiment are presented in Figure 4.23(a-c). Each line in the plot represent an orientation $(0 - 180°)$. As can be seen in the diagrams, the pose error decreases when the amount of occlusion decreases, for the VFH descriptor as well as for the VFH+PCA descriptor. Nevertheless, the mean deviation error over all shots shows that our proposed VFH+PCA descriptor outperforms the single VFH descriptor. Since the experiments for the VFH+PCA descriptor show that an occlusion of 30% results in an mean deviation error of over $35°$, the usage of a temporal MRF optimization is desirable. Our assumption that missing points lead to inaccurate pose results is further highlighted in Figure 4.23d. In the diagram we plot the deviation error (VFH+PCA) with the corresponding *missing point ratio*. This value describes the ratio between the number of points of the synthetic point cloud and the number of points of the input point cloud. A hight value (close to 1) denotes that a lot of points are missing (*e.g.* because of clutter). Complete models deliver a value close to 0. Please not that the two clouds have the same size, are not occluded and further are grid-

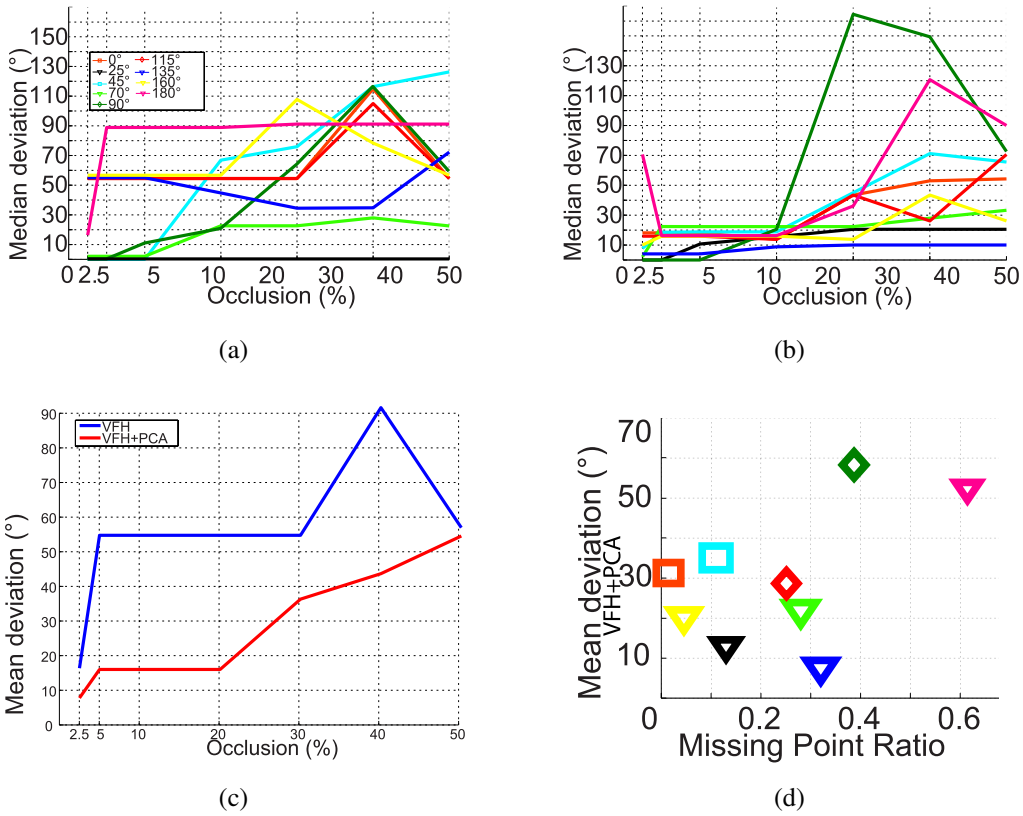


Figure 4.23: Median pose error provided by (a) the VFH descriptor and (b) the combined VFH+PCA descriptor. (c) Mean error of the descriptors over all shots. (d) Pose error concerning the number of points.

filtered. As can be seen, the shots from the side (90°) and from behind (180°) deliver a highest pose error in the experiments because of the self-occluded characteristics of the views.

4.4.2 MRF optimized 3D Object Classification Evaluation

In the previous section we have shown that our proposed classification method (VFH+PCA) has problems to deal with missing points and clutter. Such artefacts deliver PCA eigenvectors which are inaccurate or in the worst case shifted if a considerable amount of points (> 30% of occlusion) is missing. In the following, we evaluate our MRF-based optimization method. The objective of the temporal optimization method is to handle incorrect poses caused by missing points.

Experimental Setup

In order to evaluate the pose results obtained from the MRF optimization method, we first calculate the ground truth poses for all objects visible in the 10 video sequences. Secondly, in the same

manner as for the occlusion experiment the deviation in degrees is determined for each detected object. We compare the results provided by the VFH descriptor, the VFH+PCA descriptor and the results obtained from the optimization method (in the following called VFH+PCA+MRF).

Quantitative Results

We accumulate the pose errors over all sequences and detected objects. In Figure 4.24 the error curves for the three methods are visualized, in which the deviation is plotted depending on the percentage of considered objects. As can bee seen, the proposed MRF optimization method

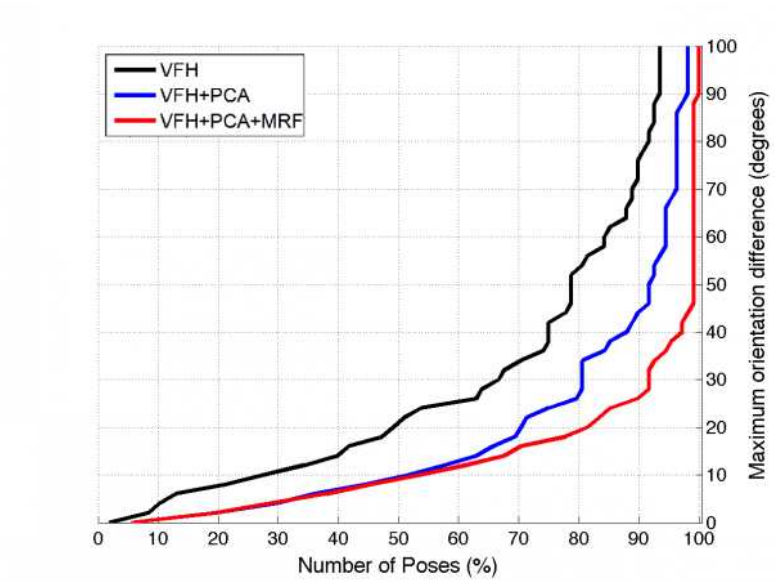


Figure 4.24: Poses estimation error over all sequences and detected objects using VFH, VFH+PCA and VFH+PCA+MRF

outperforms the frame-based classification results. 80% of the all poses yield an error less than $20°$. For the same amount of poses the VFH+PCA descriptor delivers an error of $28°$ and the raw VFH descriptor even an error of around $53°$. Nevertheless, the proposed VFH+PCA optimization provides more accurate poses compared to the VFH descriptor.

Qualitative Results

Figure 4.25 show that incorrect pose results obtained from single shots are sorted out by the proposed MRF optimization method. Since the MRF establishes a temporal inference between consecutive objects, *high* deviations between adjacent orientation vectors indicate wrong poses. For example Figure 4.25a shows that the VFH+PCA descriptor (column 2) delivers an inaccurate pose for the *chair* in the third frame (row 3). The MRF optimizes the pose result (column 3 / row 3) and returns a more exact orientation.

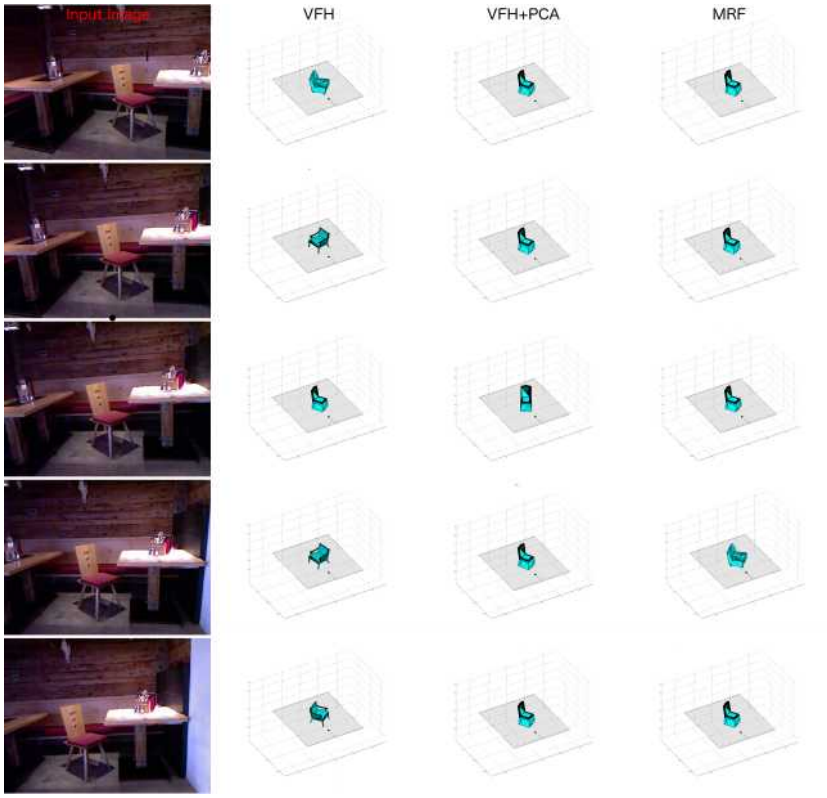
4.5 Discussion

We evaluated our framework by processing single RGBD shots as well as video sequences. For the experiments we evaluate 21 RGBD shots and 10 video sequences. The main issues of the proposed layout estimation framework are:

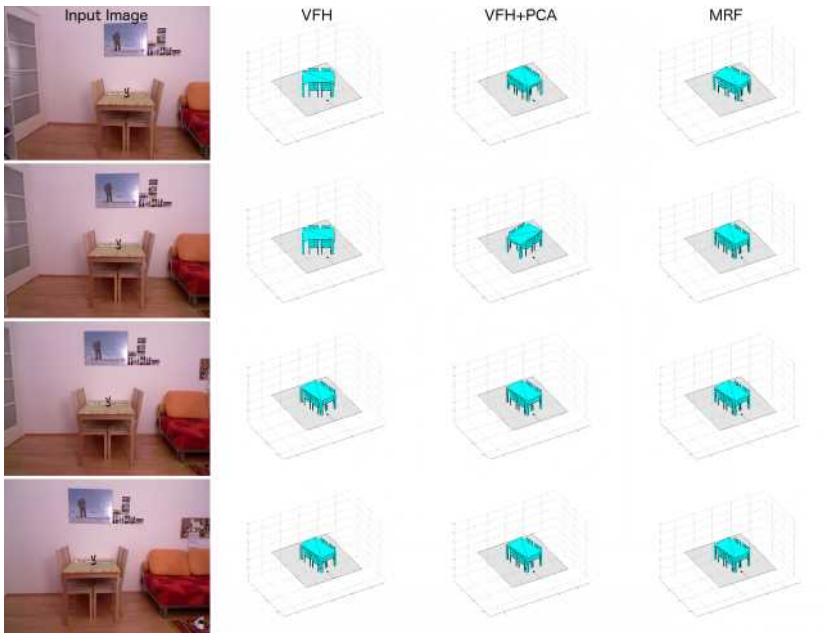
- The proposed layout estimation framework is able to estimate the spatial layout of incomplete, cluttered and noisy scenes, by providing simplified and smooth 3D models which consist of multiple 3D planes.
- While monocular methods (*e.g.* [29], [10]) fit 3D boxes into the images, our proposed framework is able to handle convex as well as concave ground planes.
- The framework recovers the spatial layout of a single RGBD frame by detecting the *ground plane*, the *walls* and the *ceiling*. Compared to monocular state-of-the-art methods, the usage of additional depth information leads to more exact 3D layouts. The proposed algorithm achieves a mean and global pixel accuracy over 90%.
- The pipeline restores missing parts which are caused by specular surfaces, direct illuminations, occlusions or other point cloud artefacts.
- Several RGBD shots can be stitched together by moving the sensor through the scene (*e.g.* rotating or walking). The fused 3D layouts consist of smooth transitions between adjacent walls and shots.
- Compared to the static layouts obtained from single shots, the fused 3D layouts lead to more accurate dimensions. The evaluation of the video sequences yield to an accuracy of 5cm for 92% of the evaluated wall segments, whereas the 3D layouts obtained from the single shots result in a mean distance of more than 17cm.
- The final 3D layouts consist of triangulated meshes. Hence, the representations are simplified and smooth. Furthermore, the resulting models are supported by common 3D file formats like PLY or OBJ.

The fused 3D models consist of a reduced number of points/vertices compared to the raw point clouds. The simplification method achieves the following results:

- A data reduction (concerning file size and number of points) of 99.9%.
- The final models require a memory storage of around 14 - 272 kB. (Requirement for the raw point clouds: 76 - 229 MB.)
- The ability to load and manipulate the 3D models on a mobile device or in a third-party software. We demonstrate an possible augmented reality use case where additional 3D models are placed into the 3D layout.



(a)



(b)

Figure 4.25: Detection results based on the three presented descriptors for (a) sample video 7 and (b) sample video 3. The arrow denotes the camera viewing direction.

Concerning the object classification the following initially addressed problems are solved:

- The framework supports the usage of synthetically rendered points clouds (CAD models) for the training stage.
- Our method is able to reject false positives in an early stage by using the additional information about the position of the detected objects in the 3D space.
- Through the usage of the combined RGB and depth information, we are able to handle objects which are placed close to other objects or the scene background.
- The PCA-based method provides an accurate (initial) pose for each single frame. Moreover, the proposed VFH+PCA descriptor outperforms the raw VFH descriptor.
- The MRF-based temporal optimization method provides more robust pose result over time. The initially estimated pose guesses for each frame are re-ranked by exploiting a temporal inference between consecutive frames. In our experiments we reach a deviation error less than 20° for around 80% of the sample scenes. For the same percentage of poses, the raw VFH descriptor results in an error of 53°.

CHAPTER 5

Conclusion and Future Work

This final chapter gives a summary of the entire work and suggestions for future research, for both the layout estimation as well as the object classification framework.

5.1 Conclusion

Three-dimensional reconstruction using a depth sensor results in a massive amount of 3D measures and consequently in complex 3D files which do not allow the real-time manipulation on mobile devices. Even the post-processing using a third-party software is a challenging task. Therefore, the present thesis presents a framework for simplifying point clouds captured using a Microsoft Kinect sensor. The framework processes multi-modal RGBD video sequences and mainly consists of two pipelines: (1) a layout estimation approach which generates simplified 3D layouts on the basis of 3D point clouds and (2) an object detection and pose estimation method which detects and further replaces objects visible in the scenes by complete CAD models. The main problem we are faced with is missing depth information caused by occlusion, direct illumination or specular surfaces. Missing points lead to holes in the point cloud reconstruction and to inaccurate pose results. Therefore, we present temporal optimization methods – for both the layout estimation as well as the object detection approach – in order to reduce the aforementioned problems.

3D Layout Estimation The layout estimation method calculates a floor plan for each frame of the input video. The framework determines planar layout segments (*ground plane*, *walls*) using a plane-fitting approach based on a labeling algorithm. For each frame a 2D mesh of the floor plan is calculated using binary image operations, morphological operations and constrained triangulation techniques. Consecutive floor plans are fused over the time and the wall segments and the ceiling are generated by extruding exterior edges of the triangulated floor plan. The resulting 3D layout is compact and watertight and further consists of a dramatically reduced number of vertices compared to the fused input point cloud. The proposed simplification allows the real-time manipulations of the 3D layouts on mobile devices as well as the post-processing using a

third-party CAD software. The temporal fusing method provides more accurate scene dimensions compared to the layouts which are obtained from single shots. Moreover, the framework detects windows using depth cues in combinations with a rectangular shape detection method. Experiments on 10 video sequences show that the amount of data can be reduced by more than 99% in terms of both – number of points and memory consumption.

Object Detection and Pose Estimation In order to obtain semantically meaningful 3D layouts, objects are detected in the scenes. The presented object and pose estimation approach uses a state-of-the-art 2D object detector which provides bounding boxes of the detected object candidates. All points within the bounding box of the detected object are extracted and each generated point cloud is classified using point clouds descriptors. We search for similar point clouds in a pre-defined database using the Viewpoint Feature Histogram which encodes the shape and the viewpoint of the point cloud at the same time. In an occlusion experiment we depict the high sensibility of the VFH descriptor concerning occlusion and missing depth information. Therefore, we present an additional descriptor which is based on a PCA approach. The objective of the VFH+PCA descriptor is to re-rank the VFH detection results by examining the angle between the eigenvectors of the object in question and the point clouds coming from the trained database. Since missing points are still a limitation, we introduce a Markov Random Field over the time. The objective of the MRF is to determine the best matching configuration (pose and type) over the entire video sequence. The unary potentials are calculated from the PCA similarity measures coming form single frames. The binary potentials are obtained from the camera movement between consecutive frames. The detection results obtained by the MRF optimization outperform the VFH descriptor as well as the combined VFH+PCA descriptor.

The framework is trained offline with synthetically generated 2.5D point clouds. First, CAD models are downloaded from a public database. Second, partial views are rendered by placing virtual cameras uniformly placed at the vertices of a bounding sphere. Third, for each generated partial point cloud a VFH descriptor is calculated and a K-Nearest Neighbour classifier is trained using a fast k-d tree structure.

5.2 Future Work

In the experiment section we already have discussed limitations of the framework. In this section we conduct with suggestions for future research.

3D Layout Estimation The presented layout estimation method fuses frames along the horizontal direction. Future work could cover the integration of a method to fuse shots along the horizontal direction as well. This would be useful to process scenes where the ground plane and the ceiling are not visible simultaneously.

The results have shown limitations concerning the scene configuration. Since objects are scaled along one direction only, it is possible that detected objects reach beyond a wall segment. Thus, a method to establish a geometrically consistent configuration in the 3D space (*e.g.* [10, 46, 60, 75]) could be integrated into the framework.

Selected scenes have shown a displacement between adjacent shots. To overcome this artefact, a MRF-based optimization method could be incorporated in order to force smooth transitions. This problem could also be reduced by the usage of additional plane-fitting techniques [42] on the vertices of the final 3D model in an post-processing step.

A further improvement would be the usage of additional mapping techniques, with the goal to provide visually more realistic room layouts.

Concerning our experiments, in future work we could evaluate our layout estimation framework using the NYUD dataset presented in [60] which contains of RGBD video sequences of indoor scenes.

The layout estimation pipeline presented in this thesis is independent of the object detection approach. Therefore, future work could cover on the integration of a joint layout estimation and object detection approach. Similar works (*e.g.* [60,72]) exploit depth and appearance features at the same time in order to estimate 3D layouts and clutter present in the scene.

Object Detection and Pose Estimation Concerning the object classification framework, the usage of more sophisticated object detection methods would be interesting. We use an image-based state-of-the-art detector in order to pre-detect object candidates in the RGB image. Since RGBD data is available, in future work we could apply a multimodal object detector instead (*e.g.* [28,47,61]).

We have seen that incomplete point clouds are the major problem we are faced with. An interesting future approach would be the usage of more complete 3D models instead of single 2.5 point clouds for the object classification. Several 2.5D point clouds could be fused together in order to get a densely sampled representation of the object in question. The classification could be performed on the fused point cloud using descriptors like spin-images [40] or spherical harmonic invariants [7].

Another idea to handle incomplete point clouds is the usage of more sophisticated machine learning methods. The pose estimation problem could be formulated as regression problem to handle the problem of missing depth information. Thus, a random forest classifier could be trained with the synthetic 2.5 point clouds (*e.g.* [19]).

We have seen that the proposed object detection method assumes that the objects are placed perpendicular to the ground plane. Thus, a future task could cover the integration of a full 6 DOF point cloud descriptor [3] which allows the detection of 6 DOF poses.

The incorporation of additional object classes, and the opportunity to detect objects which are placed on other objects [60] (*e.g.* a potted-plant placed on a table) are further limitations which could be addressed in future research.

Bibliography

[1] Sven Albrecht, Thomas Wiemann, Martin Günther, and Joachim Hertzberg. Matching cad object models in semantic mapping. In *Proc. of the ICRA-11 Workshop on Semantic Mapping, Perception and Exploration*, 2011.

[2] A. Aldoma, Zoltan-Csaba Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics Automation Magazine*, pages 80–91, 2012.

[3] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *IEEE International Conference on Computer Vision Workshops*, pages 585–592, 2011.

[4] Alexander Hermans andGeorgios Floros and Bastian Leibe. Dense 3d semantic mapping of indoor scenes from RGB-D images. In *Proc. of the International Conference on Robotics and Automation*, pages 2631–2638, 2014.

[5] Mathieu Aubry, Daniel Maturana, Alexei A Efros, Bryan C Russell, and Josef Sivic. Seeing 3d chairs: Exemplar part-based 2d-3d alignment using a large dataset of cad models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3762–3769, 2014.

[6] Jack E Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, pages 25–30, 1965.

[7] Gilles Burel and Hugues Hénocq. Three-dimensional invariants and their application to object recognition. *Signal Processing*, pages 1–22, 1995.

[8] Edwin Catmull and James Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, pages 350–355, 1978.

[9] L. P. Chew. Constrained delaunay triangulations. In *Proc. of the 3rd Annual Symposium on Computational Geometry*, pages 215–222, Chew, L. P.

[10] Wongun Choi, Yu-Wei Chao, Caroline Pantofaru, and Silvio Savarese. Understanding indoor scenes using 3d geometric phrases. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 33–40, 2013.

[11] Jon Cook, Simon Gibson, Toby Howard, and Roger Hubbold. Real-time photo-realistic augmented reality for interior design. In *ACM SIGGRAPH 2003 Sketches & Applications*, 2003.

[12] James M Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. *Neural Computation*, pages 1063–1088, 2003.

[13] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision*, pages 1–22, 2004.

[14] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[15] Edward R. Dougherty. *An Introduction to Morphological Image Processing*. SPIE, 1992.

[16] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, pages 11–15, 1972.

[17] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, pages 551–559, 1983.

[18] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, pages 303–338, 2010.

[19] Gabriele Fanelli, Thibaut Weise, Juergen Gall, and Luc Van Gool. Real time head pose estimation from consumer depth cameras. In *Proc. of the 33rd International Conference on Pattern Recognition*, pages 101–110, 2011.

[20] Li Fei-Fei, Asha Iyer, Christof Koch, and Pietro Perona. What do we perceive in a glance of a real-world scene? *Journal of Vision*, pages 1–29, 2007.

[21] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1627–1645, 2010.

[22] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, pages 55–79, 2005.

[23] David A. Forsyth and Jean Ponce. *Computer Vision A Modern Approach*. Prentice Hall, 2003.

[24] Axel Furlan, Stephen Miller, Domenico G Sorrenti, Li Fei-Fei, and Silvio Savarese. Free your camera: 3d indoor scene understanding from arbitrary camera motion. In *Proc. of the 24th British Machine Vision Conference*, 2013.

[25] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proc. of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 209–216, 1997.

[26] Flavia Grosan, Alexandru Tandrau, and A Nuchter. Localizing google sketchup models in outdoor 3d scans. In *International Symposium on Information, Communication and Automation Technologies*, pages 1–6, 2011.

[27] Abhinav Gupta, Alexei A Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proc. of the 11th European Conference on Computer Vision*, pages 482–496, 2010.

[28] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Proc. of the 13th European Conference on Computer Vision*, pages 345–360, 2014.

[29] Varsha Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *Proc. of the 12th International Conference on Computer Vision*, pages 1849–1856, 2009.

[30] Varsha Hedau, Derek Hoiem, and David Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *Proc. of the 11th European Conference on Computer Vision*, pages 224–237, 2010.

[31] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering free space of indoor scenes from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2807–2814, 2012.

[32] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *International Journal of Robotics Research*, pages 647–663, 2012.

[33] C Herrera, Juho Kannala, and Janne Heikkilä. Joint depth and color camera calibration with distortion correction. *Transactions on Pattern Analysis and Machine Intelligence*, pages 2058–2064, 2012.

[34] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 876 – 888, 2012.

[35] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *11th Asian Conference on Computer Vision*, pages 548–562, 2012.

[36] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, page 2007, 151–172.

[37] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. *ACM Transactions on Graphics*, pages 577–584, 2005.

[38] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *Proc. of the 15th RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317, 2011.

[39] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. of the 24th Annual ACM Symposium on User Interface Software and Technology*, pages 559–568, 2011.

[40] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Transactions on Pattern Analysis and Machine Intelligence*, pages 433–449, 1999.

[41] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Proc. of the SIGGRAPH Symposium on Geometry Processing*, pages 156–164, 2003.

[42] Charalampos Koniaris, Darren Cosker, Xiaosong Yang, and Kenny Mitchell. Texture mapping techniques for volumetric mesostructure. *Journal of Computer Graphics Techniques*, pages 18–59, 2014.

[43] Jeff Kramer, Nicolas Burrus, Florian Echtler, Daniel Herrera, and Matt Parker. *Hacking the Kinect*. Springer, 2012.

[44] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *International Conference on Robotics and Automation*, pages 4007–4013, 2011.

[45] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.

[46] David Changsoo Lee, Abhinav Gupta, Martial Hebert, and Takeo Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Advances in Neural Information Processing Systems*, pages 1288–1296, 2010.

[47] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *IEEE International Conference on Computer Vision*, pages 1417–1424, 2013.

[48] David G Lowe. Object recognition from local scale-invariant features. In *Proc. of the 7th International Conference on Computer Vision*, pages 1150–1157, 1999.

[49] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.

[50] Ajmal S Mian, Mohammed Bennamoun, and Robyn A Owens. Automatic correspondence for 3d modeling: an extensive review. *International Journal of Shape Modeling*, pages 253–291, 2005.

[51] Joseph Newman, Friedrich Fraundorfer, Gerhard Schall, and Dieter Schmalstieg. Construction and maintenance of augmented reality environments using a mixture of autonomous and manual surveying techniques. In *Proc. of the 7th Conference on Optical 3-D Measurement Techniques*, pages 18–59, 2005.

[52] Sebastian Nowozin and Christoph H Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, pages 185–365, 2011.

[53] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3d models with shape distributions. In *Proc. of the International Conference on Shape Modeling & Applications*, pages 154–166, 2001.

[54] T. Pavlidis. *Algorithms for Graphics and Image Processing*. Springer, 1982.

[55] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München, 2009.

[56] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proc. of the International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[57] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. 3d object classification for mobile robots in home-environments using web-data. In *Proc. of the 23rd International Conference on Intelligent Robots and Systems*, pages 2155–2162, 2010.

[58] Matthew S. Ryan and Graham R. Nudd. The viterbi algorithm. *Proc. of the IEEE*, pages 268–278, 1993.

[59] Alexander G Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2815–2822, 2012.

[60] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. of the 12th European Conference on Computer Vision*, pages 746–760, 201.

[61] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *Proc. of the 13th European Conference on Computer Vision*, pages 634–651, 2014.

[62] Peter Su and Robert L. Scot Drysdale. A comparison of sequential delaunay triangulation algorithms. In *Proc. of the 11th Annual Symposium on Computational Geometry*, pages 61–70, 1995.

[63] Min Sun, Gary Bradski, Bing-Xin Xu, and Silvio Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *Proc. of the 11th European Conference on Computer Vision*, pages 658–671, 2010.

[64] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[65] Camillo J Taylor and Anthony Cowley. Parsing indoor scenes using rgb-d imagery. In *Robotics: Science and Systems*, pages 401–408, 2013.

[66] Eric Turner and Avideh Zakhor. Floor plan generation and room labeling of indoor environments from laser range data. In *Proc. of the 9th International Conference on Computer Graphics Theory and Applications*, pages 22–33, 2014.

[67] Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision*, pages 137–154, 2004.

[68] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. Kintinuous: Spatially extended KinectFusion. *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[69] Walter Wohlkinger and Markus Vincze. 3d object classification for mobile robots in home-environments using web-data. In *Proc. of the 19th International Workshop on Robotics in Alpe-Adria-Danube Region*, pages 247–252, 2010.

[70] Christian Wojek, Stefan Walk, Stefan Roth, Konrad Schindler, and Bernt Schiele. Monocular visual scene understanding: Understanding multi-object traffic scenes. *Transactions on Pattern Analysis and Machine Intelligence*, pages 882–897, 2013.

[71] Chenxi Zhang, Liang Wang, and Ruigang Yang. Semantic segmentation of urban scenes using dense depth maps. In *Proc. of 11th the European Conference on Computer Vision*, pages 708–721, 2010.

[72] Jian Zhang, Chen Kan, Alexander G Schwing, and Raquel Urtasun. Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In *IEEE International Conference on Computer Vision*, pages 1273–1280, 2013.

[73] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, pages 213–238, 2007.

[74] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, pages 119–152, 1994.

[75] Yibiao Zhao and Song-Chun Zhu. Image parsing with stochastic scene grammar. In *Advances in Neural Information Processing Systems*, page 2011, 73–81.

Nomenclature

2D	2 Dimensional
2.5D	(2-and-a-half-Dimensional
3D	3 Dimensional
BOW	Bag Of Words
CAD	Computer-Aided Design
CRF	Conditional Random Field
CVFH	Clustered Viewpoint Feature Histogram
DOF	Degrees Of Freedom
DPM	Deformable Part Model
FPFH	Fast Point Feature Histograms
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
ICP	Iterative Closest Point
kB	Kilobyte
k-D	K-Dimensional
KNN	K-Nearest Neighbours
MAP	Maximum A Posteriori Probability
MB	Megabyte
MRF	Markov Random Field
OBJ	Object File (3D Model Format)

PCA	Principal Component Analysis
PCL	Point Cloud Library
PFH	Point Feature Histogram
PLY	Polygon File Format
RANSAC	RANdom SAmple Consensus
RGB	Red Green Blue
RGBD	Red Green Blue Depth
SH	Spherical Harmonics
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SVM	Support Vector Machine
TSDF	Truncated Signed Distance Function
VFH	Viewpoint Feature Histogram
VOC	Visual Object Challenge
VTK	Visualization Toolkit