Tampere University

POURIA BABAHAJIANI

Geometric Computer Vision

Omnidirectional visual and remotely sensed data analysis

Tampere University Dissertations 425

Tampere University Dissertations 425

POURIA BABAHAJIANI

Geometric Computer Vision
Omnidirectional visual and remotely sensed data analysis

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion remotely
on Friday 28th May 2021, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Information Technology and Communication Sciences
Finland

Responsible supervisor and Custos
Professor
Moncef Gabbouj
Tampere University
Finland

Supervisor
Professor
Joni-Kristian Kämäräinen
Tampere University
Finland

Pre-examiners
Professor
Azeddine Beghdadi
Institut Galilee, University Sorbonne Paris Nord
France

D.Sc. (Tech)
Alireza Razavi
Scania Group
Sweden

Opponents
Professor
Azeddine Beghdadi
institut Galilee, University Sorbonne Paris Nord
France

Associate Professor
Sid Ahmed Fezza
National Institute of Telecommunications and ICT
Algeria

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

Cover design: Roihu Inc.

ISBN 978-952-03-1978-6 (print)
ISBN 978-952-03-1979-3 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-1979-3

PunaMusta Oy – Yliopistopaino
Joensuu 2021

Abstract

Information about the surrounding environment perceived by the human eye is one of the most important cues enabled by sight. The scientific community has put a great effort throughout time to develop methods for scene acquisition and scene understanding using computer vision techniques.
The goal of this thesis is to study geometry in computer vision and its applications. In computer vision, geometry describes the topological structure of the environment. Specifically, it concerns measures such as shape, volume, depth, pose, disparity, motion, and optical flow, all of which are essential cues in scene acquisition and understanding.
This thesis focuses on two primary objectives. The first is to assess the feasibility of creating semantic models of urban areas and public spaces using geometrical features coming from LiDAR sensors. The second objective is to develop a practical Virtual Reality (VR) video representation that supports 6-Degrees-of-Freedom (DoF) head motion parallax using geometric computer vision and machine learning.
The thesis's first contribution is the proposal of semantic segmentation of the 3D LiDAR point cloud and its applications. The ever-growing demand for reliable mapping data, especially in urban environments, has motivated mobile mapping systems' development. These systems acquire high precision data and, in particular 3D LiDAR point clouds and optical images. A large amount of data and their diversity make data processing a complex task. A complete urban map data processing pipeline has been developed, which annotates 3D LiDAR points with semantic labels. The proposed method is made efficient by combining fast rule-based processing for building and street surface segmentation and super-voxel-based feature extraction and classification for the remaining map elements (cars, pedestrians, trees, and traffic signs). Based on the experiments, the rule-based processing stage provides substantial improvement not only in computational time but also in classification accuracy. Furthermore, two back ends are developed for semantically labeled data that exemplify two important applications: (1) 3D high definition urban map that reconstructs a realistic 3D model using input labeled point cloud, and (2) semantic segmentation of 2D street view images.
The second contribution of the thesis is the development of a practical, fast, and robust method to create high-resolution Depth-Augmented Stereo Panoramas (DASP) from a 360-degree VR camera. A novel and complete optical flow-based pipeline is developed, which provides stereo 360-views of a real-world scene with DASP. The system consists of a texture and depth panorama for each eye. A bi-directional flow estimation network is explicitly designed for stitching and stereo depth estimation, which yields state-of-the-art results with a limited run-time budget. The proposed architecture explicitly leverages geometry by getting both optical flow ground-truths. Building architectures that use this knowledge simplifies the learning problem. Moreover, a 6-DoF testbed for immersive content quality assessment is proposed.
Modern machine learning techniques have been used to design the proposed architectures

addressing many core computer vision problems by exploiting the enriched information coming from 3D scene structures. The architectures proposed in this thesis are practical systems that impact today's technologies, including autonomous vehicles, virtual reality, augmented reality, robots, and smart-city infrastructures.

Preface

This thesis owes its existence to the help, support, and inspiration of many people. First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Moncef Gabbouj for his advice, support, and patience over the years to make the thesis happen. Prof. Gabbouj has provided an excellent environment that enables me to focus on the research work. His words of wisdom, critical thinking, and admirable personality conveyed through our discussions and meetings will benefit me in the long run.
I am indebted to Prof. Joni-Kristian Kämäräinen as my supervisor for his excellent guidance and continuous encouragement for my research work. I am very grateful for his precise instruction, relentless support, and careful review of my papers.
I am also thankful to Dr. Lixin Fan. He has endless storage of innovative research ideas, and it is always motivating to brainstorm these ideas with him. He has significantly contributed to every paper in this thesis. Without him, this thesis would not exist or would be at least very different.
I would like to thank the pre-examiners Prof. Azeddine Beghdadi from Institut Galilee, University Sorbonne Paris Nord, France, and Dr. Alireza Razavi from Scania Group, Sweden, for their valuable comments on the thesis.
I have had the pleasure of working in SAMI group. The atmosphere in the group has been friendly and supportive. I am especially thankful to my awesome office mates and friends, Firas Laakom, Ali Senhaji and Fahad Sohrab. I would like to thank Virve Larmila, Ulla Siltaloppi and Elina Orava, for their great help of routine but important administrative work.
I am also thankful to Nokia Technologies for providing the opportunity to conduct my research in the Media Technologies Research LAB. My gratitude also goes to my colleagues at Nokia, Hamed Sarbolandi, Dr. Yu You, Dr. Tinghuai Wang, Junsheng Fu, Kimmo Roimela, Johannes Pystynen and Henri Toukomaa for the great work environment.
I would like to express special thanks to my great friends, in particular, Dr. Ali Vakili, Dr. Mohsen Hozan, Hasan Zirak, Hama Mamle, Kamkars, Abbas Kamandi, Naser Razazi and Najmeh Gholami for their friendship, spiritual support, and endless encouragement.
Last but not least I want to thank my family for supporting me unconditionally. Without my parents' love and support, I would not have been able to pursue my academic interests freely. I dedicate this thesis to Zhiwan and Golzhin.

Tampere 6.4.2021
Pouria Babahajiani

Contents

Abstract iii

Preface v

List of Figures ix

List of Tables xi

List of Symbols xii

List of Abbreviations xv

List of Publications xix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Background and Objectives . . . 2
- 1.3 Contributions and Publications . . . 4
- 1.4 Thesis Outline . . . 4

2 Geometry in Computer Vision 7
- 2.1 Introduction . . . 7
- 2.2 Depth Sensors and 3D Vision . . . 8
  - 2.2.1 Depth-Sensing Methods . . . 8
  - 2.2.2 Fundamentals of LiDAR Point Cloud . . . 9
  - 2.2.3 Stereo Matching . . . 11
  - 2.2.4 2D-3D Correspondences . . . 12
  - 2.2.5 Comparison of LiDAR and Stereo-Photogrammetry PCs . . . 13
- 2.3 Related Works . . . 16
  - 2.3.1 Semantic Segmentation of Point Cloud . . . 16
  - 2.3.2 360-degree 6-DOF Volumetric VR Content . . . 18

3 Contributions 21
- 3.1 MLS 3D LiDAR Point Cloud Classification . . . 21
  - 3.1.1 Semantic Segmentation of TLS LiDAR Point Cloud . . . 27
  - 3.1.2 3D HDM Reconstruction . . . 28
  - 3.1.3 Semantic Segmentation of Street View Images . . . 29
- 3.2 Depth-Augmented Stereo Panorama (DASP) Using Stereo Matching . . . 32
  - 3.2.1 Bi-Directional Flow Estimation Network . . . 39
  - 3.2.2 Datasets:Unity Synthesized Scene . . . 43

3.2.3 Perceptual Quality Test Paradigm for DASP with 6-DoF 43

4 Conclusion **51**

Bibliography **53**

Publication I **67**

Publication II **77**

Publication III **93**

Publication IV **105**

Publication V **123**

List of Figures

1.1 The schematic view of the variety of algorithms developed in this thesis. . . . 6

2.1 Examples of depth-sensing devices. 8
2.2 Registered 3D point cloud from MLS LiDAR 11
2.3 3D point cloud from TLS LiDAR 11
2.4 Geometry of epipolar line in stereo-photogrammetry for 3D reconstruction. . 12
2.5 The conversion of 3D MLS LiDAR point cloud to the 2D sparse depth map. (Left) The reference images, (Middle) 3D LiDAR point cloud images defined by projecting the 3D point cloud into the coordinate of reference images, (Right) the depth maps, generated by finding the closest point among 2D projections. 14
2.6 Alignment of TLS LiDAR with VR camera point cloud. 15

3.1 The overall workflow of the MLS 3D LiDAR point cloud classification [P2]. . 22
3.2 Example of 3D HDM [P3]. 28
3.3 2D-3D association [P3] 30
3.4 Impact of LiDAR intensity feature in 2D semantic segmentation [P1] 32
3.5 Illustration of depth-augmented stereo panorama and stitching procedure. The purpose of computing depth is to support high-quality novel view synthesis in VR [P5]. 34
3.6 Structure of the flow-based interpolation pipeline [P5]. 35
3.7 Flow-based view synthesis and depth reconstruction [P5]. 36
3.8 Example of depth-augmented stereo panorama visualization. Given the camera model and depth map, it is possible to project the image texture outward to create a colorized point cloud corresponding to the 3D surfaces seen by the sensors [P5]. 38
3.9 Summary of our end-to-end deep stereo regression architecture for bi-directional optical flow estimation. (a) Overview of the network architecture. The network takes an image pair as input and predicts both forward and backward optical flow using a 7 level pyramid setting. (b1) Feature extractor. (b2) Optical flow estimator. The siamese encoder-decoder architecture for predicting optical flows is shown at pyramid level 2. (b3) The context network used as post-processing to refine the optical flows. [P5]. 40
3.10 Our Unity Synthesized Scene (USS) dataset was built for evaluation and training. a) The artificial scene is imported into Unity3D and gets rendered using the script and the Shader b) The scenes include two indoor and three outdoor scenarios. We modified the "main camera" in Unity to generate our desired dataset. We can flexibly control and move the cameras rig and render our data by synchronized virtual lenses [P5]. 44

3.11 Perceptual quality test paradigm for DASP with 6-DoF. Our proposed method takes into account the actual user sensing experiences. Stitching and depth estimation methods are evaluated quantitatively and qualitatively with synthetic data to validate their functionality by comparing generated contents rendered in different viewports with those viewports created using known 3D geometry [P5]. 46
3.12 Qualitative and quantitative evaluation of 6-DOF contents [P5]. 48
3.13 Qualitative results from challenging indoor and outdoor scenes. Left-eye equirectangular contents are shown for each scene because the corresponding right-eye is almost indistinguishable from the left-eye when seen in 2D. Our method achieves very accurate and reliable results in fast run-time inherited from the bi-directional CNN flow network. We can get the feeling of immersion when watching them in HMD [P5]. 49

List of Tables

2.1 Comparison of LiDAR systems mounted on different platforms. 10

3.1 Geometric and photometric primitives used to classify super-voxels into predefined categories [P1], [P2]. 25
3.2 Confusion matrix of our method for classification of the NAVTEQ True dataset [P4]. 26
3.3 Computing times of our method with and without the rule-based steps. Without the rule-based step all points are classified using the super-voxel and boosted decision tree method [P4]. 26
3.4 Comparison of proposed method to other reported results on 3D point cloud classification with the Paris-Rue-Madame dataset [P4]. 26
3.5 Comparison of our method to other reported results on 3D point cloud classification with the TLS Velodyne dataset [P4]. 27
3.6 Confusion matrix of pixel-wise accuracies of our method for direct 2D semantic classification of the NAVTEQ True street view images [1]. 30
3.7 Confusion matrix of the pixel-wise accuracies for 2D semantic segmentation of the NAVTEQ True images. The semantic labels of 3D point clouds that are classified based on 3D voxel features are mapped to the image plane [P4]. . . 31
3.8 Quantitative evaluation on public optical flow benchmarks. We report the average EPE for all benchmarks, except KITTI, where F1-all and Out-noc are used to benchmark KITTI 2015 and 2012, respectively. Out-noc presents the percentage of outliers with errors more than 3 pixels in non-occluded regions, whereas F1-all presents the percentage in all regions. For all measures, lower is better. Results are divided into methods trained with (−ft) and without fine-tuning. Entries in parentheses indicate methods that were fine-tuned on the evaluated datasets. *Our network uses different training data. Since Sintel and Kitti do not provide the ground-truth backward flow, we report numbers on the test sets and from the forward flow. The run times are measured based on one forward inference on the MPI-Sintel benchmark's final pass [P5]. . . 42

List of Symbols

c_i	Angular positions of virtual camera
C	Tensor's channels size
d	Pixel's motion rang
D_i^j	Depth map derived from F_i^j flow map
$\mathbf{f}$	Encoder output tensor in flow estimator
f_x	Focal length along x-axis
f_y	Focal length along y-axis
F_i^j	Flow map, where I_i^j is the reference image and I_j^i is the target image
H	Tensor's height size
I_i	Fish-eye image
I_i^j	Rotated fish-eye image I_i with respect to the view shared with I_j
$\mathbf{K}$	camera intrinsic matrix
$\mathbf{M}$	world to camera transformation matrix
p	2D projection of point cloud on the image plan
P	3D point cloud
$\mathcal{P}_{building}$	Building facades point cloud
$\mathcal{P}_{road}$	Road surface point cloud
$\mathcal{P}'$	Point cloud from which the building facades and road surface points have been removed
$\mathbf{R}$	3×3 rotation matrix
$\mathbf{T}$	3×1 translation vector
u_0	Principle point along y-axis
v_0	Principle point along y-axis
W	Tensor's width size

$\widetilde{W}^l$	Up-sampled flow map from lth level
x	Pixel index
$\mathbf{X}$	First coordinate of a point
$\mathbf{Y}$	Second coordinate of a point
$\mathbf{Z}$	Third coordinate of a point
γ	Skew coefficient between the x- and the y-axis
Θ	Seam's center position in degree
λ	Regularization trade-off parameter
λ_d	Balancing factor for equal weighting for the height and density scores
τ	Threshold
ω	Model's learnable parameters

List of Abbreviations

1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional
AI	Artificial Intelligence
ALS	Airborne Laser scanning
AR	Augmented Reality
AV	Autonomous Vehicle
BDT	Boosted Decision Trees
BDC	Bayesian Discriminant Classifier
BIM	Building Information Model
BoW	Visual Bag-of-Words
CNN	Convolutional Neural Network
DASP	Depth Augmented Stereo Panorama
DL	Deep Learning
DM	Dense Matching
DoF	Degree of Freedom
DoG	Difference of Gaussians
DPM	Deformable Part Model
DSM	Digital Surface Model
EPE	End Point Error
FoV	Field of View
GPS	Global Positioning System
GPU	Graphics Processing Unit
HDM	High Definition Map

HD	High Definition
HOG	Histogram of Oriented Gradients
HT	Hough Transform
ICP	Iterated Closest Point
IMU	Inertial Measurement Unit
IoT	Internet of Things
IR	Immersive Rendering
ITU	International Telecommunications Union
LiDAR	Light Detection and Ranging
ML	Machine Learning
MLS	Mobile Laser Scanning
MOS	Mean Opinion Score
MRF	Markov Random Field
MVS	Multi-View Stereo
MZV	Minimal-Z-Value
ODS	Omni Directional Stereo
PC	Point Cloud
PCS	Point Cloud Segmentation
PCSS	Point Cloud Semantic Segmentation
PSNR	Peak-Signal-to Noise Ratio
QF	Quality Factor
RANSAC	RANdom SAmple Consensus
RF	Random Forests
RMS	Root Mean Square
SfM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SR	Scene Recorder
SSIM	Structural Similarity Index Measure
SURF	Speeded Up Robust Features
SVM	Support Vector Machine

TLS	Terrestrial Laser Scanning
ToF	Time of Flight
USS	Unity Synthesized Scene
VQA	Visual Quality Assessment
VR	Virtual Reality
WGS	World Geodetic System

List of Publications

This thesis is based on the following articles, which are referred to in the text by notation [P1], [P2], and so forth.

P1 P.Babahajiani, L.Fan, M.Gabbouj: "Semantic parsing of street scene images using 3D LiDAR point cloud", Proceedings of the IEEE International Conference on Computer Vision Workshops, 2013.

P2 P.Babahajiani, L.Fan, M.Gabbouj: "Object recognition in 3D point cloud of urban street scene", Asian Conference on Computer Vision. Springer, 2014, 177-190.

P3 P.Babahajiani, L.Fan, J.-K. Kämäräinen, M.Gabbouj: "Comprehensive automated 3D urban environment modelling using terrestrial laser scanning point cloud", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016.

P4 P.Babahajiani, L.Fan, J.-K. Kämäräinen, M.Gabbouj: "Urban 3D segmentation and modelling from street view images and LiDAR point clouds", Machine Vision and Applications 28.7, 2017, DOI 10.1007/s00138-017-0845-3.

P5 P.Babahajiani, S.Husseini, M.Gabbouj: "Depth-Augmented Stereo Panorama from 360 Degree Camera for 6-DOF VR Rendering", submitted to Machine Vision and Applications, April 2020.

1 Introduction

1.1 Motivation

Computer vision is a multidisciplinary field of science that aims to design methods and systems that provide human-like visual capabilities so that a scene can be sensed and interpreted to take appropriate actions. The recent unprecedented advancement of Machine Learning (ML) techniques has addressed many issues in computer vision. ML solutions revolve around data generation and gathering, model training, and model deployment for a given task, such as regression, classification or prediction.
The common computer vision application is image classification, where the task is to classify an image according to its visual features. On the other hand, object detection is the task of localizing an object within an image and it is generally done by estimating a 2D bounding box around the object. Multi-class object detection, as the name suggests, provides location and instance information for multiple objects. In segmentation, a class label is assigned to each pixel in the image, which yields the localization of the object. On the other hand, semantic segmentation attempts to partition an image into semantically meaningful parts and classify each part into one of the pre-determined classes.
Semantic representations use a specific language to describe relationships in the scenes. For example, we may describe an object as a "car" or a "pedestrian". One problem with relying just on semantics to design a representation of a scene is that semantics is human rather than machine generated and understood concept. A critical distinction between an Artificial Intelligence (AI) system and a human is how each reasons about the world: a human uses high order semantic abstractions whereas an AI system uses blind adherence to statistics.
An AI system needs to understand human semantics to perform such an interface. However, visual reasoning based on semantic representations can be a challenging task for an AI system. For instance, the object class "car" may contain all four-wheeled vehicles, and such a large intra-class variation in appearance and shape makes the visual reasoning challenging. Moreover, illumination changes or inadequate lighting conditions can change the objects' appearance and lead to possible confusion with respect to their definition.
The supporter paradigm for semantic representations in computer vision is geometry. In computer vision, geometry describes the topological structure of the environment. Specifically, it concerns measures such as shape, volume, depth, pose, disparity, motion, and optical flow. The mentioned challenges motivate the research on the deployment of practical ML architectures that utilize geometric knowledge to simplify the learning problem.
Learning from both semantics and the scene's geometry is expected to produce more realistic results. Consequently, many complex relationships in a scene, such as object shape, depth, and motion, do not need to be learned from scratch with ML.

Another important application domain of computer vision is Virtual Reality (VR), where input images are acquired, processed, analyzed, and understood, the same way as humans, in order to extract meaningful data to provide a 360-degree visual experience. To create VR content from multi-lens cameras, we need a projection of a real-world scene, which requires finding and matching feature points between frames. This motivates the research on understanding scene geometry through ML, particularly Deep Learning (DL), to efficiently find the corresponding points between two shapes of an object seen in adjacent frames.
The thesis proposes novel ML architectures to address several core computer vision problems. The proposed architectures are practical systems that impact today's technologies, including Autonomous Vehicles (AV), virtual reality, Augmented Reality (AR), robots, and smart-city infrastructure [1].

1.2 Background and Objectives

The goal of the thesis is to study geometry in computer vision and its applications. In particular, the aim is to improve ML solutions' performance by exploiting the enriched information coming from 3D scene structures. This study has two primary objectives.
The first objective is to assess the feasibility of creating semantic maps of urban areas and public space using geometrical features coming from Light Detection and Ranging (LiDAR) sensors.
Accurate and efficient scene perception of urban environments is critical for different applications, including High Definition (HD) mapping, autonomous driving, 3D model reconstruction, and smart city. LiDAR is an emerging technology for collecting 3D Point Clouds (PC) of object surfaces efficiently. Mobile mapping, especially based on Mobile Laser Scanning (MLS), involves instruments mounted on a moving platform, e.g., on a car or truck. The mobile systems scan the environment and capture images by several cameras.
While LiDAR systems provide a readily available solution for capturing 3D spatial data quickly, efficiently, and accurately, semantic data labeling still requires enormous human resources if done manually. Therefore, the problem of automatic labeling (parsing) of 3D urban data to associate each 3D point with a semantic class label (such as "car" and "tree") has recently gained momentum in the computer vision community.
Automatic labeling and segmentation of the point cloud remain challenging due to some data-specific challenges: (1) High-end laser scanning devices generate millions of data points per second, and, therefore, the methods need to be efficient to cope with the sheer volume of the urban scene datasets. (2) Point cloud sub-regions corresponding to individual objects are imbalanced, varying from sparse representations of distant objects to dense clouds of nearby objects, and sometimes incomplete (LiDAR system scans only one side of objects). (3) To train the supervised object detection methods, sufficiently large labeled training data (e.g., ground-truth) is needed. Moreover, some MLS systems also integrate camera sensors to simultaneously record a video log and provide color information to improve classification and detection accuracy. Therefore, point cloud registration with RGB values is essential.
In this thesis, to reach the above stated objective, we aim to answer the following research questions: What is the best way to infer information from large 3D point clouds? How to recognize objects and semantics in 3D scenes? How to obtain robust features from the 3D point cloud? How well does the geometric information perform in semantic segmentation? Is it possible to use a machine learning model with classical computer vision algorithms

to enhance semantic segmentation model performance? How rule-based segmentation is applied, and what is the effect of this stage on the overall pipeline? How to make a 3D map of urban areas using currently available measurement technologies and represent the information on this map?
The second objective is to propose a novel approach for rendering 6-Degrees-of-Freedom (6-DoF) panoramic videos from a multi-lens camera through geometric computer vision and machine learning algorithms.
Digital representations of 3D scenes and virtual models are considered promising media in contemporary life and have strong potential in education [2], manufacturing, entertainment, and medical applications, among others. VR video and 360-VR are essentially interchangeable terms that refer to videos captured using specialized omni-directional cameras, which enable filming an entire 360 degrees simultaneously. The 360 videos can be viewed on a growing number of media devices, including mobile phones. However, the most immersive experience is created when viewed with a VR headset. By doing so, the user is free to look around the entire scene and often experiences the feeling of actually "being there".
VR videos are typically shot using multiple cameras pointing in different directions. With stitching, images of the same object captured from multiple cameras with different perspectives are blended to form an illusion of a single continuous image such as an equirectangular format. This perspective difference, or parallax, creates a disparity in equirectangular images that must be compensated for by warping, blending, optical flow, or other pixel-pushing techniques.
While monoscopic 360-videos are perhaps the most common type of content creation for VR applications, they lack 3D information and cannot be viewed with full 6-DoF; hence, one cannot just move around the 3D real world.
When depth value is available for every pixel in the stitched image, it can be utilized to view content from different viewpoints by projecting the image's pixels to their 3D locations and re-projecting them onto a new view. However, obtaining depth information from real images is challenging even for the state-of-art vision algorithms.
Stitching and depth estimation steps for delivering such rich multimedia content are the main technical focus and complex parts of service deployment for any 6-DoF VR production. Despite recent progress, several open issues related to the accuracy, speed, and scalability of reconstruction systems remain to be solved. These challenges motivate the research to deploy practical ML models, reconstructing the 3D geometry from the original imagery content to provide an accurate and real-time VR rendering pipeline supporting the 6-DoF head motion.
To achieve this objective, we aim to answer the following additional research questions in the thesis: Can the entire stereo vision problem be formulated with deep learning using our understanding of stereo geometry? Can we build a compact and effective CNN model for optical flow estimation to be used for view compositing? How much the performance of optical flow methods affect Depth Augmented Stereo Panorama's (DASP) quality? Can we improve the performance of a CNN model by training it on the synthesized VR dataset? How well does the converted optical flow to depth map perform in actual user 3D sensing experiences? How much DASP model solutions support the head motion? How can we realistically measure DASP quality?
The common objective of the aforementioned research goals is to develop efficient and accurate inference techniques and systems for computer vision applications by leveraging the scene's geometry.

1.3 Contributions and Publications

The major contribution of this thesis is the proposal of semantic segmentation of the 3D LiDAR point cloud and its applications, described in detail in publications [P1], [P2], [P3], and [P4]. Moreover, a practical, fast, and robust method to create high-resolution depth-augmented stereo panoramas from a 360-degree camera is developed and described in [P5]. The candidate developed the proposed methods, performed all the experiments, and wrote the publications' initial manuscripts, while discussing the research work with the supervisor and co-authors

In [P1], we proposed a novel framework for semantic parsing of street view images based on 3D features derived from the MLS point cloud. The improvement is achieved by circumventing error-prone 2D feature extraction and matching steps. During the offline training phase, geometrical features associated with 3D patches are extracted and used to train Boosted Decision Trees (BDT) classifier. Furthermore, the classifier's robustness is improved for certain object classes, utilizing intensity information from LiDAR data. Moreover, we introduced a novel method to register the 3D point cloud to the 2D image plane, and by doing so, occluded points are removed efficiently.

In [P2], we proposed a complete urban map data processing workflow, which annotates the 3D LiDAR point cloud with semantic labels. The method is made efficient by combining fast rule-based processing for building and ground surface segmentation and super-voxel-based feature extraction and classification for the remaining map elements (pedestrians, trees, cars, and traffic signs). The rule-based processing stage provides substantial improvement not only in computing time but also in classification accuracy.

In [P3], we developed two back ends for semantically labeled urban 3D map data that exemplify two important applications: (1) 3D High Definition (HD) urban map that reconstructs a realistic 3D model using input labeled point cloud, and (2) semantic segmentation of 2D street view images by back-projection of the 3D labels.

In [P4], the two-stage point cloud segmentation proposed in [P2] was extended by transforming the models' parameters to have physical meaning (intuitive to set and validate). We provided an ablation study for the most critical parameters, and their values are optimally set by validation against both Terrestrial Laser Scanning (TLS) and MLS LiDAR datasets.

In [P5], we built a novel and complete optical flow-based pipeline that renders stereo 360-views of a real-world scene with Depth-Augmented Stereo Panorama (DASP), which consists of a texture and depth panorama for each eye. We also presented a bi-directional flow estimation network explicitly designed for stitching and stereo depth estimation, which yields state-of-the-art results with a limited run-time budget. Moreover, we proposed and developed a 6-DoF testbed for immersive content quality assessment. Finally, due to the unavailability of synthesized datasets for training the bi-directional flow estimator network, validation, and quality assessment of VR contents, we have simulated a virtual camera and generated a rich dataset (including RGB fish-eye and equirectangular images) using the 3D Unity game engine. Our dataset consists of indoor and outdoor scenes and provides dense ground-truth depths for the full images.

1.4 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 presents the fundamental and critical concepts of geometric computer vision and scene acquisition using active and passive sensors. The background related to the topic is briefly discussed, and the literature

on 3D point cloud semantic segmentation and omni-directional visual data rendering is reviewed. The thesis' contributions are presented in Chapter 3. MLS LiDAR point cloud classification of urban environments is first introduced, then extended to on-board TLS point cloud datasets. Using the labeled point clouds, a model-based 3D High Definition MAP (HDM) for better user experience is presented. Contributions to 2D semantic segmentation of street view images are then presented. In addition to the 3D point cloud's semantic segmentation, depth-augmented stereo panorama rendering is formulated as a learning problem. The proposed algorithm includes a practical CNN model for optical flow estimation. To train the supervised CNN network, computer graphics, and virtual reality technologies are utilized to create a realistic and large-scale synthesized dataset, in which the fidelity and geographic information can match the real world well. Finally, A test-bed for quality assessment of 6-DoF immersive content is presented. A schematic diagram illustrating the relationships between the methods presented in the thesis is shown in Figure 1.1. In Chapter 4, the main results of the thesis are summarized. All five original publications [P1, P2, P3, P4, P5] can be found at the end of the thesis.

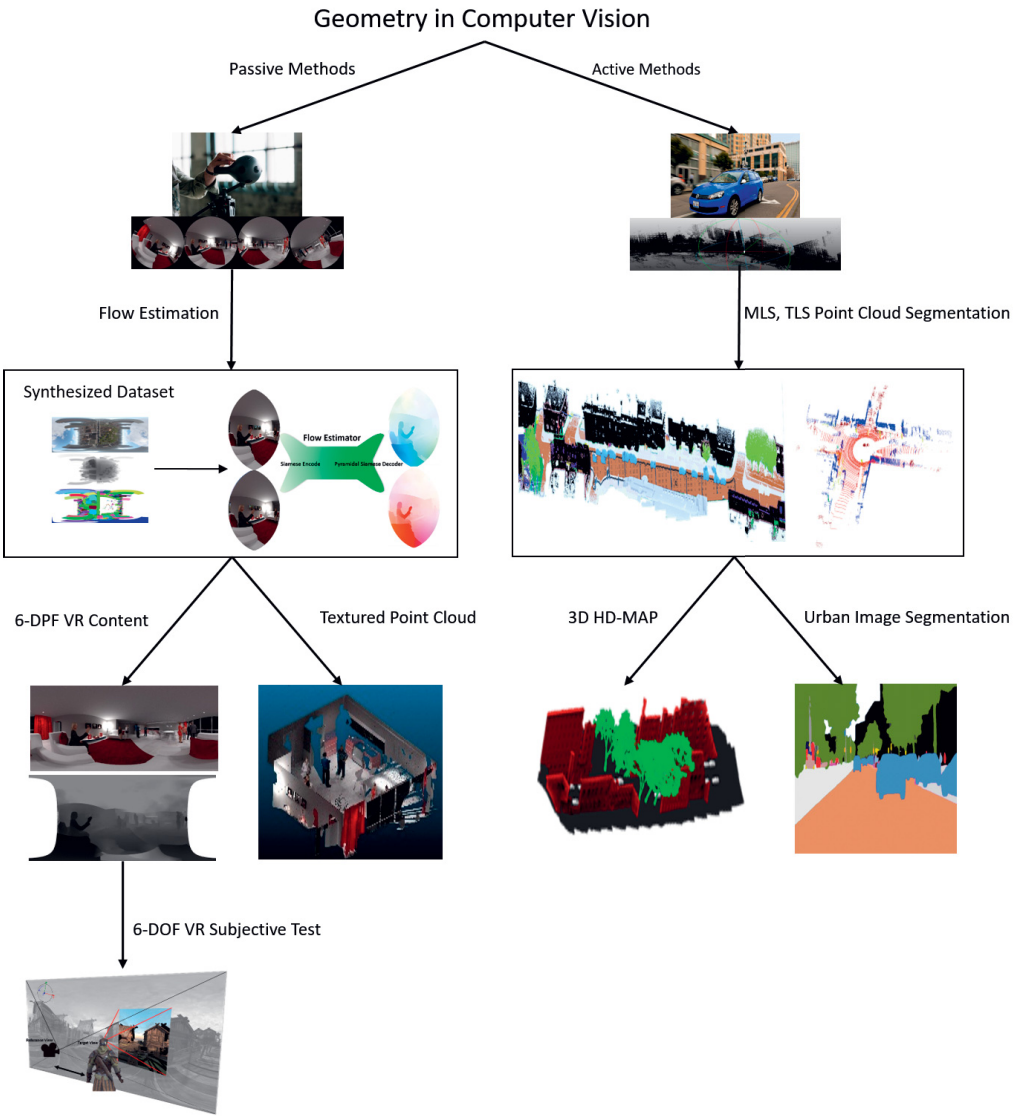


Figure 1.1: The schematic view of the variety of algorithms developed in this thesis.

2 Geometry in Computer Vision

2.1 Introduction

Nowadays, computer vision systems perform challenging tasks that were not possible twenty years ago because of limited computational and sensor resources. Despite the progress reported in 2D computer vision applications, open issues related to accuracy, speed, and generalization remain to be solved. One important reason is that rendering 2D images of a 3D world is inherently a lossy process, and the majority of geometric information of the 3D world projected onto a flat 2D image vanishes. Therefore, any attempt to undo this operation is a challenging task, or in some scenarios, it is entirely impossible.

Perceiving the 3D geometric configuration of a scene or object is essential for many computer vision applications. For example, an autonomous vehicle requires depth sensing to know its location in the environment, where other relevant objects are, and most importantly, how it can safely navigate from location A to B. Depth information is also necessary for human-machine interaction of VR/AR devices. Devices must respond accurately to the 3D movement of users, which need high-performance depth sensors. Moreover, free-viewpoint video rendering from any 3D location of a space captured by 360-cameras is an example in the field of virtual reality, which is strongly dependent on the estimation of 3D scene geometry.

Geometry in computer vision deals with geometric relations between the 3D world and its projection into the 2D image. A common problem in this field relates to reconstructing geometric structures in the 3D world based on 2D images' features. In the particular case of stereo-photogrammetry, 3D vision systems recover information about the 3D scene's structure based on stereo-pair images of the scene and decoding the depth information. Depth information is implicitly encoded as relative displacements between stereo-pair images' content. The 3D scene structure can be inferred using depth information and through a geometry process known as triangulation.

Another important domain of geometric computer vision is for visualization based applications. The main objective here is to create images from new viewpoints, using several images of a scene. This field of research is called view synthesis or image-based rendering. This is achieved by creating a textured 3D model of the scene and then render it from the desired viewpoint. View synthesis algorithms generally include several geometric processes such as camera calibration, affine transformation, data fusion, optical flow, pose, and motion estimation.

This chapter covers a brief overview of geometric computer vision and its fundamental techniques, which are the basis of the proposed contributions in the thesis. Section 2.2 provides a brief overview of how depth sensors currently work and where they are used in 3D vision systems. Section 2.3 reviews the literature on geometric computer vision

(a) VR camera (MVS depth sensing method) **(b)** A vehicle equipped with LiDAR

Figure 2.1: Examples of depth-sensing devices.

applications, particularly in 3D semantic segmentation and 360-degree volumetric data rendering.

2.2 Depth Sensors and 3D Vision

The most basic of geometrical concepts is the point. In geometry, the term *point* is used to specify a unique location in a specific space. A point cloud on the other hand is a collection of points that represents a 3D shape or feature. A point in 3D space is represented by its **X**, **Y** and **Z** coordinates, and in some cases, additional attributes may be used. We can think about a point cloud as a collection of multiple points. Surprisingly, when many points are brought together, they start to show some interesting qualities of the feature they represent.
A depth map transforms a 3D point cloud into a 2D image. The depth map and point cloud are two different ways to represent a 3D scene's spatial structure. However, with a point cloud, one can see all the points, whereas a depth map typically only reflects points from the point cloud that can be seen from a particular viewpoint in the point cloud's coordinate system. Point clouds are most often created by methods used in stereo-photogrammetry or remote sensing.
For this purpose, we first introduce different depth sensors used in computer vision (Section 2.2.1). Then we provide the sensor measuring models of LiDAR (Section 2.2.2) and stereo vision systems (Section 2.2.3). How 3D point cloud data is transformed to the respective 2D image is presented in Section 2.2.4. Finally, Section 2.2.5 briefly compares the stereo-photogrammetric and LiDAR-based scene reconstruction methods.

2.2.1 Depth-Sensing Methods

Generally, depth-sensing involves matching pairs of pixels between aligned images from two different cameras with known positions and then using the resulting parallaxes (i.e., relative displacements of the contents of one of the images of the stereo-pair with respect to the other image) to obtain the depth of objects in the environment. This scheme, which mimics the binocular vision of humans, is called a two-view stereo. The location estimation of an object in the 3D world can be further refined by capturing more images. Such a passive reconstruction method where all the information is extracted purely from images is called Multi-View Stereo (MVS). While such calibrated systems in MVS are used

to simultaneously capture two or more images, motion stereo systems, as another passive method, take images from two or more locations using a single camera. The advantage of stereoscopic techniques is their capability to produce dense depth maps integrated with the surroundings' rich visual data by utilizing low-cost cameras. Recently, low-cost stereoscopic well-calibrated cameras offer good solutions for autonomous vehicles [3, 4], robotic vision [5], and AR/VR devices [6–8]. However, dense stereo depth estimation is computationally quite complex and requires high processing power. This is due to the requirement of matching corresponding points in the stereo images. Furthermore, depth estimation using the MVS system has a limited depth-sensing range and does not work well with texture-less surfaces and poor low lighting conditions (see Section 2.2.5). Figure 2.1a illustrates an example of an omni-directional camera acquiring depth information using stereo information.

Other important categories of perception sensors and systems to deliver accurate geometrical data used by computer vision algorithms are active depth sensors. Active depth sensors or depth cameras are devices, which measure distances between the device and the scene surfaces using some specific hardware. Active depth cameras are divided into two subgroups depending on the underlying measuring technique: (1) structured light-based sensors and (2) Time-of-Flight (ToF) based sensors.

Structured light sensors [9, 10] project a known pattern (static infrared structured light) on the scene's surfaces and estimate the scene's structure from the changes observed between the projected and captured patterns. In this method, known light patterns are sequentially projected onto an object, and the geometric shape of the object deforms the patterns.

ToF sensors [11, 12] on the other hand measure the time of flight of a known signal from the moment it is emitted to the time when the device receives its reflection.

One type of ToF range sensor is known as Light Detection and Ranging (LiDAR). LiDAR is a remote sensing technology that estimates the range by sending a pulsed laser to the scene, followed by detecting the reflected pulse using a photo-detector. For outdoor robotics and autonomous vehicles, LiDAR scanners are the most practical sensors to date. Other depth sensors, including structured light cameras (e.g., Kinect V1 [13]), continuous wave ToF devices (e.g., Kinect V2 [14]), and passive stereo cameras, typically have a limited range of depth sensing and do not function well in the presence of low lighting and too much ambient light (such as light from the sun).

LiDAR can measure distances even with a few microns accuracy, but such devices are expensive and bulky and are not designed for ordinary consumers. The main drawback of a LiDAR system is that the depth maps obtained from the projection of the LiDAR are sparse. A LiDAR system is most commonly mounted on a vehicle and can capture detailed geometric information of the roadway and the surrounding area in the form of point clouds. Due to the movement of the vehicle and the rotary motion of the LiDAR sensor, point cloud data density will be increased compared to the static model, and denser depth maps will be recorded. The LiDAR sensor setup is illustrated in Figure 2.1b.

2.2.2 Fundamentals of LiDAR Point Cloud

Scene reconstruction algorithms from LiDAR point clouds can be grouped based on the input data or the output model (see Table 2.1). Algorithms designed to work on Airborne Laser Scanning (ALS) considerably differ from those developed for Mobile Laser Scanning (MLS) and Terrestrial Laser Scanning (TLS).

When utilizing LiDAR data for a potential application, three primary challenges should be

Table 2.1: Comparison of LiDAR systems mounted on different platforms.

	Scanning Perspective	*Accuracy*	*PC Density*	*Application Areas*
MLS	Side view	$\pm 10cm$	Dense	Road mapping, 3D urban areas
TLS	Side view	$\pm 3cm$	Relatively sparse	Deformation monitoring
ALS	Top view	$\pm 15cm$	Sparse	Terrain mapping, forest surveys, 3D urban areas

considered: (1) LiDAR dataset comprises millions or even billions of points with geometric, colorimetric, and radiometric attributes, requiring high computational processing resources and storage to handle such a large volume of data. (2) Although LiDAR sensor provides a high amount of data with each revolution, the resulting raw point clouds are essentially a set of discrete data and do not inherently have any semantic information. (3) Because a LiDAR system acquires data with a high spatial resolution, noise present in the scene and unwanted objects are simultaneously recorded.

LiDAR point cloud is collected with the assistance of the Global Positioning System (GPS) and an Inertial Measuring Unit (IMU) to position and orient the sensor in the 3D World Geodetic System 1984 (WGS84) [P1]. Along with the 3D (XYZ) data, the sensor also gives a fourth value for each point, called intensity [P1]. This value expresses the strength of the reflection from that particular point. This intensity value mostly depends on the object's surface, i.e., shiny and flat surfaces reflect light much better than matte ad scattered surfaces (such as roads or vegetation). However, the angle of attack sometimes affects this intensity value, i.e., if the beam hits the surface at a high angle, the light can scatter, and thus the returning intensity will be lower. An important advantage of this technology over conventional optical imaging is that neither 3D coordinates of point clouds nor their intensity values are affected by lighting conditions.

LiDAR sensors and cameras are often mounted together on a mobile platform (see Figure 2.5). In this configuration, the LiDAR sensor scans the space with every rotation, and one or more cameras capture digital images in constant time or distance intervals.

In the MLS LiDAR system (Figure 2.2), data preprocessing is required to register the point cloud form with every rotation. Once we have the registered point cloud, the scanned scene's resolution will be uniform, and most of the occlusion will be gone due to the moving platform.

One challenging and current application of the MLS LiDAR system equipped with cameras is developing Highly-Definition Maps (HDM) for autonomous driving. MLS system has significant advantages in obtaining 3D HDMs with high precision and clarity. Compared to ALS, continuous collection of MLS point clouds of high point density allows the capture of detailed road features such as curbs and surface road marking. The point density of TLS data could reach the same level as MLS, but TLS data has a non-homogeneous point distribution and a much lower productivity than MLS. TLS (Figure 2.3) is generally utilized as an on-board sensor in autonomous driving.

One problem with using the MLS system is that moving objects (e.g., vehicles and pedestrians) lead to inaccurate 3D models, and such 3D models cause undesirable artifacts in virtual views generated using the 3D models. For example, if we produce virtual views using view-dependent image-based rendering techniques from 3D models, including moving objects, implausible textures often appear. In Figure 2.5, the red car as a moving object appears like a long-drawn shadow, and only one side is completely scanned. Moreover, the background of the car is continuously occluded and is seldom observed.

To cope with this problem, point cloud semantic segmentation and classification are

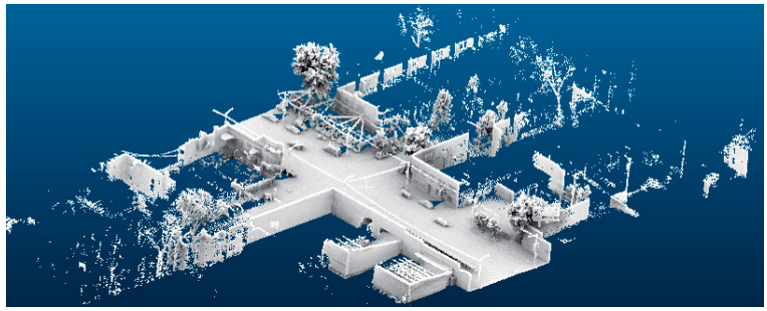


Figure 2.2: Registered 3D point cloud from MLS LiDAR

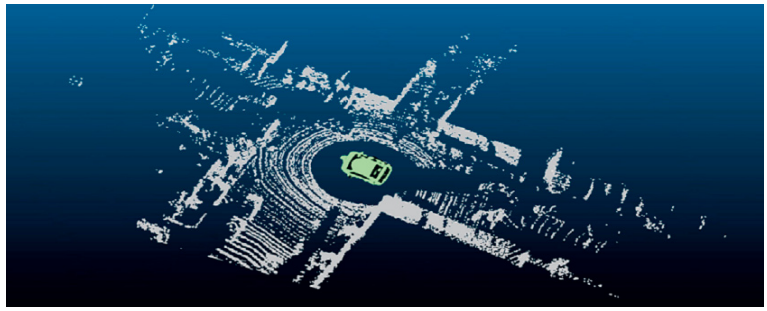


Figure 2.3: 3D point cloud from TLS LiDAR

needed to filter these points from the 3D model and associate the remaining points with semantic labels.

2.2.3 Stereo Matching

In order to reconstruct 3D structures of a scene with reasonable ambient lighting conditions from optical images, different approaches have been presented, such as stereo matching techniques [15], Structure-from-Motion (SfM) [16, 17] and shape from shading [18]. We will focus on the stereo matching technique, which exploits two or more images of a scene with reasonable ambient lighting conditions, acquired with either multiple cameras or one moving camera, and estimates the respective 3D structure of the scene by finding the corresponding points in the images' plane and converting their 2D locations into 3D depth values based on triangulation.
Classical stereo matching techniques are based on using two images captured in two different positions and rely on one fundamental theory: if a large part of the scene is seen in both images, there will be differences (per pixel shifts) between the images due to the camera's different position. We call 2D shifts "parallax." The amount that the 2D coordinates of an object (3D point) in one image are shifted with respect to the same object in the other image is inversely proportional to that object's depth.
In a stereo vision system, when the cameras are well-calibrated, and both intrinsic and extrinsic parameters are known, computing 2D coordinate correspondences under the epipolar geometry constraint becomes a 1D search problem.
In the absence of calibration, rectification will require the computation of the fundamental matrix [19], which will be subjected to errors, thus making the estimation of 3D geometry from stereo images practically impossible.

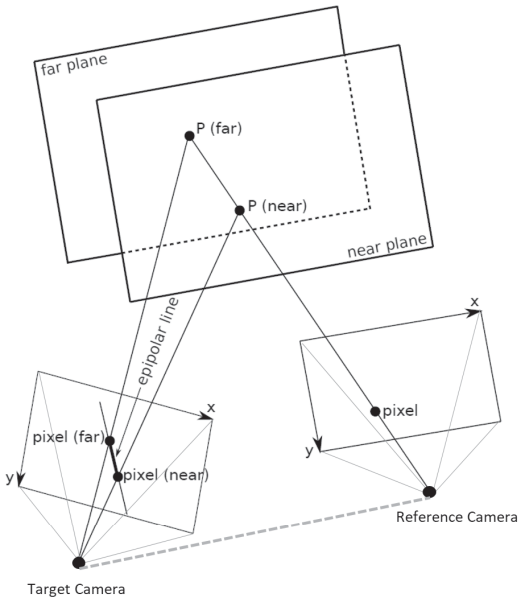


Figure 2.4: Geometry of epipolar line in stereo-photogrammetry for 3D reconstruction.

Unrectified stereo cameras can also estimate depth maps as long as they depict the same objects in a scene (Figure 2.4). With two rectified stereo images, the process is simplified as matches can be made along the horizontal scan-lines. The process is slightly complicated if we want to estimate the depth map from unrectified stereo cameras. In this case, It is easier to consider one camera as the reference camera and the second one as the target camera and assume the camera parameters are known (both extrinsic and intrinsic). The depth map will be generated with respect to the reference camera. By considering a series of planes perpendicular to the reference camera axis (which pass through the camera center and are perpendicular to the image plane), 3D objects seen by the reference camera will be fitted between a far plane and a near plane. When a pixel in the reference image is projected onto planes between the far and near planes, the corresponding projected pixels in the target image define a line segment. This line on the image plane of the target image is referred to as the epipolar line. In rectified stereo matching, the epipolar line is horizontal and it is called a scan-line.
Note that epipolar geometry defines a mapping from a point in the reference image to an epipolar line in the target image and only depends on camera calibration properties and is independent of the scene's 3D structure. There is not much difference between getting the depth map using the epipolar line (unrectified stereo) or the scan-line (rectified stereo) in the case of accurate camera parameters.

2.2.4 2D-3D Correspondences

Given a 3D point cloud and a virtual image plane with a known viewing camera pose, the association module described in this section aims to establish correspondences between collections of 3D points and 2D image pixels. We project the 3D point cloud $P = [\mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \ldots, \mathbf{P}^{(m)}]$ onto the reference image plane p. For the i-th 3D point, $\mathbf{P}^{(i)} =$

$[x^{(i)}, y^{(i)}, z^{(i)}, 1]^T$, we generate a 2D projection $\mathbf{p}^{(i)} = [u^{(i)}, v^{(i)}]^T$ on the image plane by:

$$\mathbf{p}^{(i)} = \mathbf{K} \quad \mathbf{M} \quad \mathbf{P}^{(i)} \tag{2.1}$$

where $\mathbf{M}$ is the world to camera transformation matrix and $\mathbf{K}$ is the intrinsic matrix of the image plane. $\mathbf{M}$ and $\mathbf{K}$ can be represented by (2.2) and (2.3):

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{T} \end{array} \right] \tag{2.2}$$

where $\mathbf{R}$ is a 3×3 rotation matrix, and $\mathbf{T}$ is a 3×1 translation vector.

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Where f_x and f_y are the focal lengths in terms of pixels along the x- and y-axis; γ represents the skew coefficients between x- and y-axis, and it is often 0; u_0 and v_0 represent the location of the principal point which would ideally be in the center of the image. Using the projection step in Equation (2.1), we can identify those 3D points projected within a specific patch (e.g., pixel or superpixel) in the image plane. Since we assume that only one dominant 3D point is associated with the given 2D patch, outlier 3D points that are far from the patch should be removed. To successfully find the closest point among 2D projections, a plane-fitting method can be used, e.g., as in [20]. The mapping uses Z-buffering [21], which compares surface depth at each patch position (surface depth is measured from the view plane along the Z-axis). When using a Z-buffer, if another object of the scene must be rendered in the same patch, the algorithm compares the two depths and overrides the current patch if the object is closer to the camera projection center. The chosen depth is then saved to the Z-buffer. 3D point cloud features such as intensity, density, and majority vote label projected to the same image patch could also be stored (see Figure 2.5)

2.2.5 Comparison of LiDAR and Stereo-Photogrammetry PCs

Recent image-based stereo-photogrammetry take advantage of the following main components: (a) stereo imaging configuration (e.g., 360-degree rigs or cameras, stereo sensors selection), (b) cost-free increase of overlap between images when sensing digitally, (c) multi-view matching algorithms, and (d) Graphics Processing Unit (GPU), making complex image matching algorithms very practical. These enablers lead to an improved photogrammetric method, so that point clouds are created at dense intervals and almost in real-time. On the other hand, LiDAR point clouds have conquered a major position ever since point clouds have become a mapping data product. The advantages of one method over the other have been the topic of studies and discussions during the last decade [22 – 26].

Both stereo-photogrammetric and LiDAR-based approaches have shown good performance in the literature and several industrial applications. However, more academic research recommends to combine LiDAR with imaging data [27, 28]. The use of both techniques in combination means that LiDAR can add details that photogrammetry data may have missed. The 3D measurement remains a domain of the LiDAR approach; the images serve as a 2D augmentation. Thus, when LiDAR and photogrammetry are combined, they bring more detail to a perception system that may not have been achieved individually. The fusion of 3D LiDAR data with stereoscopic images is addressed in [28, 29]. Various

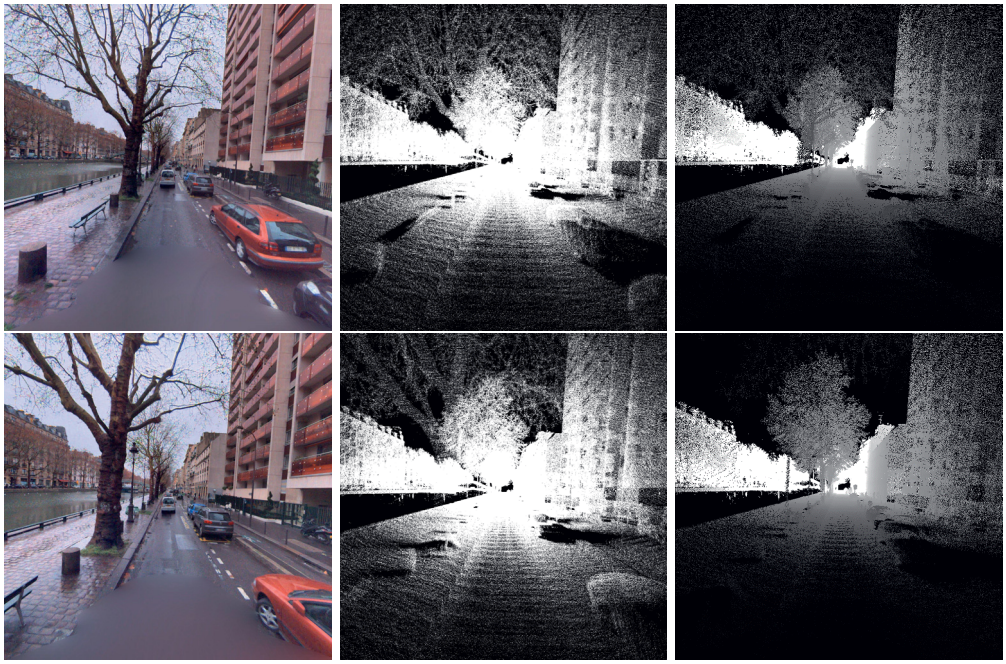
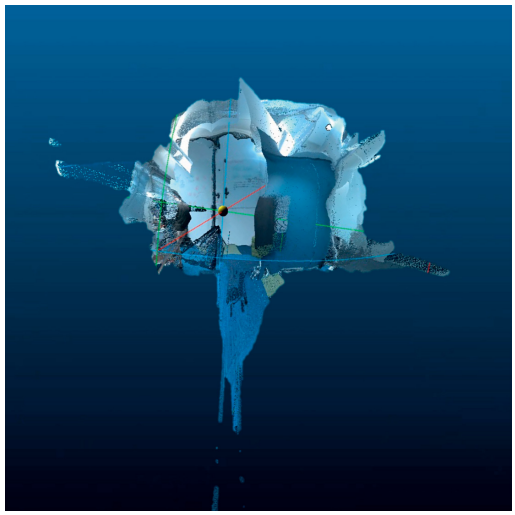
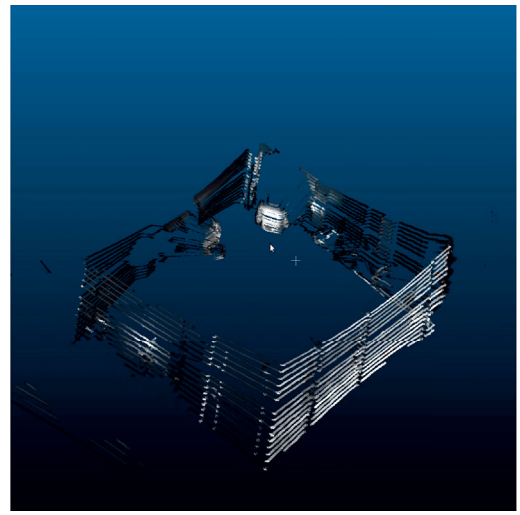


Figure 2.5: The conversion of 3D MLS LiDAR point cloud to the 2D sparse depth map. (Left) The reference images, (Middle) 3D LiDAR point cloud images defined by projecting the 3D point cloud into the coordinate of reference images, (Right) the depth maps, generated by finding the closest point among 2D projections.

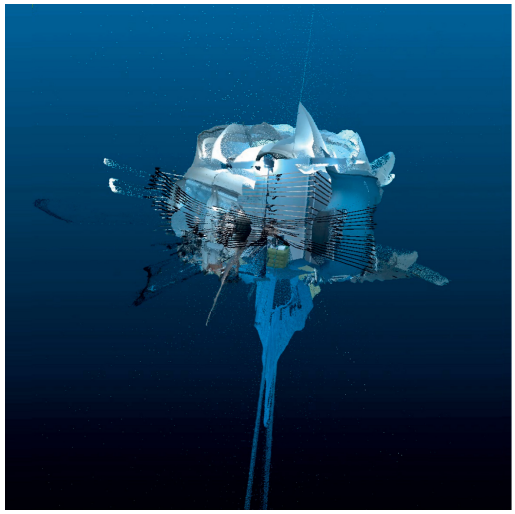
important issues combining photogrammetric and LiDAR data may prove to be quite a difficult task due to the significantly different characteristics of optical images and LiDAR data, e.g., the alignment of both data sets may present technical challenges. Moreover, the structural characteristics recorded by optical imaging may not be present in LiDAR data or vice-versa. Nevertheless, the main disadvantage of using LiDAR combined with high quality passive optical sensors is typically a more expensive and bulky system. Figure 2.6 compares point clouds from the LiDAR system with those created from the VR camera as two depth sensors, which are the basis of our main contributions. As can be seen, compared to the LiDAR point cloud, the photogrammetric method produces a higher density and more semantic. However, it is less accurate than the LiDAR method.

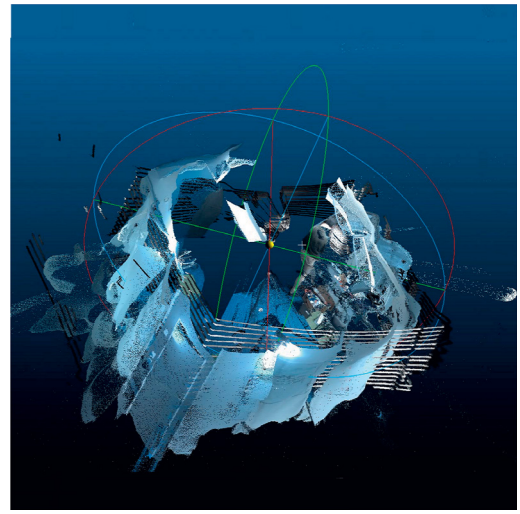


(a) 3D stereo-photogrammetric point cloud



(b) 3D LiDAR point cloud





(c) Alignment of the textured point cloud from VR camera with LiDAR point clouds using the Iterative Closest Point (ICP) algorithm.

Figure 2.6: Alignment of TLS LiDAR with VR camera point cloud.

2.3 Related Works

This section reviews and summarizes the literature on semantic labeling of LiDAR point cloud and street view image approaches. Then, relevant methods of depth-augmented stereo panorama for 6-DoF VR are discussed.

2.3.1 Semantic Segmentation of Point Cloud

3D data classification and segmentation using the image and point cloud data of urban areas have many potential applications in virtual reality, autonomous driving, and robotics. Research on these topics has thus gained momentum during the last few years. In the following, we briefly review the most important image-based approaches but keep the main focus on 3D point cloud processing and methods particularly tailored for urban 3D modeling. Several important surveys have been recently published where more details of specific approaches can be found in [30 – 33].

Image-Based Methods

Progress in image-based object segmentation and classification over the years has been remarkable [34, 35]. In particular, the Visual Bag-of-Words (BoW) [36, 37], Scale Invariant Feature Transform (SIFT) [38] and the Deformable Part Model (DPM) [39] are considered among the successful methods.
Besides the single image parsing techniques, multiple methods have tried various strategies to exploit video data's temporal features. Zhang et al. [40], Sturgess et al. [41], and Brostwo et al. [42] obtain the 3D structure information (e.g., depth maps or sparse point clouds) from video frames and then combine the 3D information and image textures to parse each frame. Xiao and Quan [43] propose a region-based parsing system on each frame and enforce temporal coherence between regions in adjacent frames by temporal fusion in a batch mode.
Structure from Motion (SfM) [44 – 46], Dense Matching (DM) [43], and Multiple View Stereovision (MVS) [47, 48] changed the image-derived point cloud, and opened the era of multiple view stereovision. SfM can automatically determine the camera's positions and orientations, making it capable of processing multiview images simultaneously, while DM and MVS algorithms provide the ability to generate a large volume of point clouds. However, the quality of generated point clouds from SfM, DM, and MVS is not as good as those formed by LiDAR techniques. Especially using SfM is not feasible to generate dense points for large scenes [49].
Recently, these approaches have been outperformed by methods using deep convolutional neural networks, e.g., AlexNet [50], and R-CNN [51]. Such networks have shifted the paradigm of hand-crafting features towards end-to-end learning, where learning feature embeddings is part of the optimization process. Direct applicability of these neural networks is unclear because the datasets utilized in the training phase contain objects in different non-urban areas (Pascal VOC [52], ImageNet [53]), and sources of detection failures may consequently be different. Furthermore, deep neural networks require large annotated datasets, which may present a limitation of the approach.

Point Cloud-Based Methods

Point Cloud Semantic Segmentation (PCSS) is the 3D form of semantic segmentation, in which irregular distributed points in a 3D scene are used instead of regular distributed pixels in an image. LiDAR point clouds are the most commonly used data in PCSS. In the

PCSS workflow, Point Cloud Segmentation (PCS) can be employed as a pre-segmentation step, affecting the ultimate results. The primary purpose of the PCS is to group raw point clouds into non-overlapping regions. Those areas correspond to specific objects or structures in the scene. The delivered results from PCS have no clear semantic information because no supervised prior knowledge is required in such a segmentation procedure. The PCS approaches could be classified into four main groups: edge-based, region growing, model fitting, and clustering-based approaches.

Edge-based PCS approaches in [54 – 57] were directly transferred from images to 3D point clouds. Similar to image segmentation approaches, the edge-based methods' principle is to locate the points that have a rapid change in intensity [32]. In [58], the authors proposed a gradient-based method for edge detection, fitting 3D lines to a set of points and detecting changes in the direction of unit normal vectors on the surface. The edge-based algorithms enable a fast PCS due to its simplicity; however, their good performance could only be achieved where simple scenes with ideal points are presented (e.g., low noise, even density) [59, 60].

Region growing [4, 61 – 64] utilizes criteria, combining local features between points or two region units in order to measure the similarities among points or voxels, and merge them if they are spatially close and have similar surface characteristics. Local features describe some properties of the local neighborhood of an object's surface points. In order to describe a complete object, a set of these local descriptors have to be used. The success of local descriptors in image classification has inspired the development of 3D local descriptors for point cloud data, e.g., 3D Speeded Up Robust Features (SURF) [65], 3D Histogram of Oriented Gradients (HOG) and Difference of Gaussians (DoG) [66], and their comparison can be found in the two recent surveys [67, 68]. Since most real-world applications deal with varying scales of objects, as well as a variety of occlusions and deformations, feature detectors and descriptors must be invariant to scaling [69], rigid and non-rigid deformations [70, 71]. A comprehensive study on surface detectors and descriptors has been published in [72].

The core idea of model fitting matches the 3D points to different primitive geometric patterns. Hence, it has been usually regarded as a shape extraction or detection technique. However, when dealing with scenes with parameter geometric shapes/models, e.g., planes, cylinders, and spheres, model fitting could also be regarded as a PCS method. The most widely used model-fitting algorithms are built on two classical methods, RANdom SAmple Consensus (RANSAC) [73], and Hough Transform (HT) [74, 75]. Model fitting algorithms allow fast running time and achieve good results in a simple scenario. The main drawbacks are that it is challenging to choose the model's size when fitting objects, it is sensitive to noise, and it is not working well in complex scenes [76].

Clustering-based approaches [77 – 79] are widely used for unsupervised PCS. This group of methods share a similar aim, i.e., grouping points with similar geometric features, spectral features, or spatial distribution into the same homogeneous pattern. Unlike region growing and model fitting, these patterns are usually not defined in advance. K-means [80] , mean shift [81], and fuzzy clustering [82] were the main algorithms in the clustering-based point cloud segmentation.

The workflow of PCSS is similar to clustering-based PCS. However, in contrast to non-semantic PCS methods, PCSS techniques generate semantic information for every point. Accordingly, PCSS is usually realized by supervised learning models. There has been a considerable amount of research to classify each point individually or each point in clusters using a cascade of binary classifiers [83], Support Vector Machines (SVM) [84, 85], AdaBoost [12, 86, 87], Bayesian Discriminant Classifiers (BDC) [88], and Random Forests (RF) [89].

Many DL models applicable to point cloud classification have recently been proposed, such as PointNet [90], PointNet++ [91], PointCNN [92], PointSIFT [93], and KPConv [94]. These models allow the direct input of the original point cloud data without having to convert the data to another form (distributed 2D images or 3D voxels). Such networks have shifted from hand-crafted features towards end-to-end learning, where learning feature embeddings is part of the optimization process. While the powerful DL models are capable of learning features automatically, complicated network architecture and a relatively long calculation time are often required to achieve a better classification result [95]. In addition, most of the DL models are derived from the computer vision field, and the point clouds processed are mostly for indoor scenes without considering the characteristic and application requirements of outdoor point cloud data in the remote sensing field, such as ALS or MLS point clouds [96].

3D Modeling using LiDAR Point Cloud

3D digital city modeling is becoming a popular research field [1]. Vosselman et al. [97] presented two strategies to reconstruct building models from segmented planar surfaces and ground plans. They employ a 3D Hough transform (HT) to detect planar segments and merge them using least squares estimation. The generated Digital Surface Model (DSM) from aerial LiDAR contains terrain and building roof information. The main drawback of their method is the limited resolution of the resulting point clouds. The limited resolution is caused by a limited number of points per square meter, which is caused by the plane's speed and distance to the ground. Another disadvantage is that building facades are not clearly visible [98].
With dense MLS LiDAR data, not only different types of objects in urban scenes (e.g., ground surface, cars, building facades, utilities) can be detected and classified, but also a complete city model could be 3D regularized and reconstructed geometrically. MLS 3D point clouds can overcome the limitations of low productivity and low geometric accuracy in real-world high-quality 3D city modeling [4, 99]. Point clouds and images acquired by the MLS system are usually combined for texture mapping, and semantic labeling for constructing 3D city mesh [100]. Furthermore, MLS data provide an efficient solution for automatic geometry generation and shape detection for Building Information Models (BIM) [101] and city 3D models [102 – 104].

2.3.2 360-degree 6-DOF Volumetric VR Content

This section gives an overview of different VR content acquisitions. Then the literature on CNN based flow estimation methods is reviewed. Finally, we review the related work for VR quality assessments.

Overview of VR Cameras and DASP Representations

Due to the significant progress in Virtual Reality (VR) displays in recent years, immersive scene exploration based on virtual reality systems has gained much attention with diverse applications in entertainment, teleconferencing, remote collaboration, medical rehabilitation [105], and education [106]. While traditional cameras record a narrow range of Field of View (FoV), panoramic content breaks such a limitation by providing a 360-degree visual experience. In VR applications, contents are usually generated by creating a panorama of a real-world scene. Although many capture devices are being released, getting high-resolution panoramas and displaying a virtual scene in real-time remains challenging due to its computationally demanding nature [105, 107].

Different optical systems have been proposed to capture a wide field of view frames [108, 109]. They may be captured from structured [110] or unstructured camera arrays [111], or even multiple moving cameras [112, 113]. The use of a rotating camera for generating panorama images from static scenes [114, 115] has also been proposed. Other methods have constructed stereo panoramas with 60-degree vertical FoV using a multi-camera system with mirrors [105, 116]. A mirror-based multi-camera system typically has several cameras in a circular configuration shooting up into a collection of mirrors that are facing out into the scene at an angle [117]. A broad discussion on the different possibilities for the acquisition of omni-directional content is provided in [108].
Current techniques for converting real scenes to be displayed on VR equipment mainly rely on multi-camera capture rigs that produce two panoramas (e.g., omni-directional stereo) providing both stereo visions and 360-degree viewing experiences to users. Surround 360 [7] and Google Jump [118] and Jaunt [119] are proposed to generate stereo panoramas from camera rigs. These systems are, however, bulky and expensive and cause self-occlusion between cameras. For example, Google Jump consists of 16 GoPro cameras in a circle of diameter 28 cm, and Facebook Surround 360 has a similar design with 17 cameras. More cameras mean more data in transmission and storage and a higher equipment cost. Other systems, such as Ricoh Theta [120], or Samsung Gear 360 [121] produce 360-degree video using fewer cameras. This allows for compact cameras but makes it more difficult to align images well in the stitching and change the minimum depth sensation distance, and hence they do not meet the target of immersion since the stereo cue is missing.
Depending on the type of camera, problems often arise when merging raw images from different cameras to make VR panoramas. For this purpose, exact calculations must be carried out at the edges so that stitching seam are no longer visible. Stitching VR raw images coming from 360-degree cameras is based on either an approximation of concentric mosaics [122–125], view interpolation between adjacent cameras for each eye [7, 118, 126], or depth-image based rendering techniques [127, 128].
While omni-directional stereo systems are popular, they lack 3D information and cannot be viewed with a full 6-DOF, e.g., they do not support head motion parallax and produce artifacts when the viewer tilts or rolls their head [126, 129, 130]. More recent systems such as [110, 118, 129] use geometric proxies and estimate per-camera depth maps, or pair-wise flow fields [131] and use these to warp source images into the desired novel viewpoints. Other approaches focus on adaptive meshing and mesh tracking to create temporally coherent geometry [132].

Optical Flow Estimation with CNNs

When stitching panoramas, parallax has to be accounted for. A common strategy, which is also the basis for our work, is to warp the images in 2D space using image correspondences. Optical flow is a method to compute a dense pixel correspondence, and its performance is crucial to image-stitching and depth recovery. Classical optical flow estimation approaches [133, 134] use variational approaches to minimize energy based on brightness constancy and spatial smoothness.
Convolutional neural networks (CNN) have proven successful for a variety of computer vision tasks, such as recognition, classification, and segmentation [135–137]. However, only in recent years, there has been some progress in applying CNN to reconstruct the 3D geometry (such as depth, flow, disparity estimation).
One of the earliest actionable applications in this scope can be traced back to Eigen et al. [138], who trained a CNN to predict depth from a single image by learning objects shape and scene structure. However, monocular depth estimation methods have problems

generalizing to previously unseen types of images [139 – 141].
Dosovitskiy et al. [142] demonstrated the use of CNN in the estimation of optical flow among image pairs with FlowNet. This work was subsequently improved in terms of accuracy and speed by Ilg et al. [143]. Recently, Kendall et al. [144] proposed a new CNN-based disparity estimation method for rectified stereo pairs, and more recently, Ummenhofer et al. [140] achieved the state-of-the-art depth estimation from unconstrained, successive image pairs from sequential image data. However, to solve the stereo-based depth estimation problem, their model relies on a chain of encoder-decoder networks that iterates over optical flow, disparity, and ego-motion estimation, which takes 110 ms on an Nvidia GTX Titan X, for an input image resolution (256×192). PWC-Net [145] succeeded to build the classical principles of optical flow into the network architecture, which yields comparable or even better results and requires less computation than FlowNet2. There have also been several attempts at optical flow estimation using unsupervised learning; however, these methods achieved a lower accuracy on standard benchmarks [146, 147].

Perceptual Quality Assessment of Immersive Contents

To ensure the best immersive user experience, developments throughout the whole chain from content capture to the VR playback (e.g., camera calibration, 3D experience, overall image quality) are needed.
Several works have been published proposing various performance evaluation for compression of omni-directional visual content [148 – 150]. Panorama quality assessment methods in [151 – 153] pay more attention to photometric error assessment such as color correction and intensity consistency, while geometric error assessment has been neglected in their evaluations. In [154], an omni-directional camera system is considered to assess video consistency among subsequent frames using a luminance-based metric around the seam. Recently, Yu et al. [155] proposed a framework for objective evaluation of omni-directional video, which allows comparing different sphere-to-plane mappings, e.g., equirectangular, Lambert cylindrical equal-area, dyadic, and cubic. A test-bed for subjective assessment of omni-directional content using HMD is presented in [156], where the authors assess the performance of monoscopic omni-directional visual contents in the field of compression. Huang et al. [157] proposed the quality evaluation of monoscopic 360-degree images viewing on HMDs, mainly focusing on spatial resolution and JPEG compression Quality Factor (QF). To the best of our knowledge, neither objective nor subjective quality evaluations of DASP omni-directional content has been investigated.

3 Contributions

This chapter summarizes the main contributions in the publications included in the thesis. These contributions can be grouped into two categories:

1. The major contribution of this thesis is the proposed semantic segmentation of 3D LiDAR point cloud and its applications, described in detail in publications [P1], [P2],[P3], and [P4].

2. A practical, fast, and robust method to create high-resolution depth-augmented stereo panoramas from a 360-degree VR camera is the second major contribution of the thesis presented in [P5].

This chapter is organized as follows: We will begin with the MLS LiDAR point cloud classification of urban environments (Section 3.1). Its extension to TLS LiDAR point cloud classification is subsequently described in Section 3.1.1. Then, two back ends for semantically labeled 3D point cloud are introduced as (1) reconstruction and visualization of 3D HDM (Section 3.1.2), and (2) semantic segmentation of 2D street view images (Section 3.1.3).
In addition to the semantic labeling of objects using geometric 3D point descriptors, we formulate a seamless depth-augmented stereo panorama as a learning problem described in Section 3.2. We presented a bi-directional flow estimation network explicitly designed for stitching and depth reconstruction in Section 3.2.1. We utilized computer graphics and virtual-reality technologies to create a realistic and large-scale synthesized dataset suitable for training and quality assessment of DASP contents (Section 3.2.2). Finally, we will present a novel Visual Quality Assessment (VQA) for VR video suitable for 3D content rendering on HMDs with 6-DoF in Section 3.2.3. As this is a compound thesis, only the most important results are summarized in this chapter. The reader is referred to the original publications for further details and results.

3.1 MLS 3D LiDAR Point Cloud Classification

This section addresses these research questions: What is the best way to infer information from large 3D LiDAR point clouds? How to recognize objects and semantics in 3D scenes? How to obtain robust features from the 3D point cloud? How well does the geometric information perform in semantic segmentation? Is it possible to use a machine learning model with classical computer vision algorithms to enhance semantic segmentation model performance? How rule-based segmentation is applied, and what is the effect of this stage on the overall pipeline?
The overall processing steps of the MLS LiDAR point cloud classification approach are

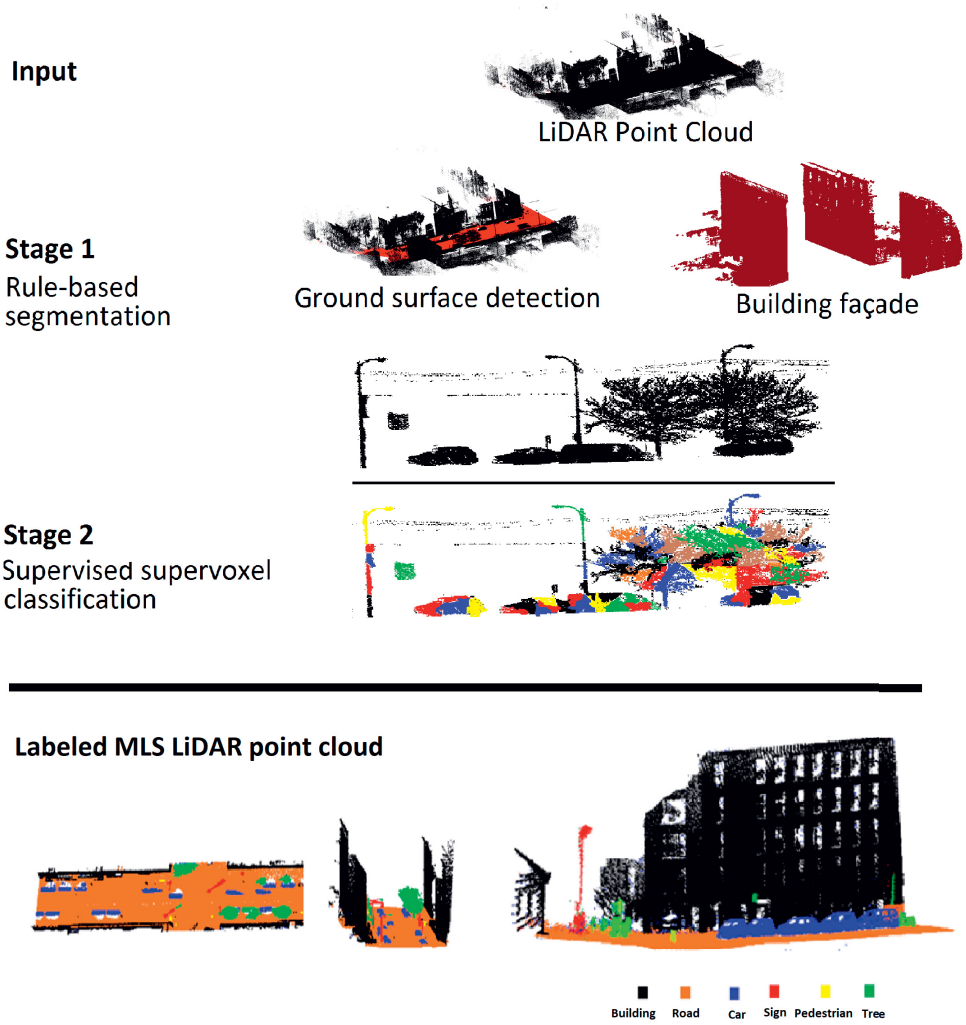


Figure 3.1: The overall workflow of the MLS 3D LiDAR point cloud classification [P2].

illustrated in Figure 3.1. The input to our processing algorithm is 3D LiDAR point cloud $\mathcal{P} = \{\vec{p}_i\}$ ($\vec{p} \in \mathbb{R}^3$). The first processing step of our methodology is the rule-based segmentation of the ground surface and building facades. The next step is super-voxel clustering and feature extraction from each super-voxel, after which the super-voxels are classified using the boosted decision tree classifier.

The rule based steps detect and label the road surface points $\mathcal{P}_{road}$, and building points $\mathcal{P}_{building}$ which are removed from the original point cloud $\mathcal{P}' = \mathcal{P} \setminus (\mathcal{P}_{road} \cup \mathcal{P}_{building})$ and then passed to the next processing step (learning-based segmentation).

First, we apply the fast and robust Random Sample Consensus (RANSAC)-based plane fitting to detect road surface points, including the car path and side-walk, and as a result, the original point cloud is divided into road surface ($\mathcal{P}_{road}$) and other points ($\mathcal{P}_{other} = \mathcal{P} \setminus (\mathcal{P}_{road})$. Starting from the road surfaces is also beneficial for the later steps

that are based on point cloud connectivity, as the road and ground surfaces connect almost all points together.
Accordingly, the original point cloud is divided into smaller point clouds $\{\mathcal{P}_{10m\times10m}\}_k$ which span 10×10 meters square areas. Secondly, each $\{\mathcal{P}_{10m\times10m}\}_k$ point cloud is further divided into $0.25m \times 0.25m$ cells, and for each cell the Minimal-Z-Value (MZV) is computed by averaging 10 lowest z value points [1]. Thirdly, all points lying within $\pm\tau_{MZV}$ distance from MZV are selected for plane fitting for each cell. The selection process reduces the number of points to around 0.1% from the original, and in all experiments, we fixed the threshold to $\tau_{MZV} = 0.02m$. For each local point cloud the points that are within the distance $\tau_{road} = \pm0.08m$ from the fitted plane are added to $\mathcal{P}_{road}$.
Our rule-based building facade segmentation workflow is derived from the dominant characteristics of them in LiDAR point clouds: LiDAR provides high (z dimension) and dense regions. We form a GPS-defined x-y plane similar to the road surface detection. Since the x-y plane after detection and removing ground surface is divided into the discrete cells, Δ_x, Δ_y, we can compute height and density features. As a height feature, we use

$$P_h(\Delta_x, \Delta_y) = \frac{\underset{z}{\text{argmax}}\ P(\Delta_x, \Delta_y, z)}{\underset{z}{\text{argmax}}\ p(:, :, z)} \ , \tag{3.1}$$

and as a density feature

$$P_d(\Delta_x, \Delta_y) = \frac{|P(\Delta_x, \Delta_y, z)|}{\max\limits_{\Delta_x,\Delta_y} |P(\Delta_x, \Delta_y, z)|} \ . \tag{3.2}$$

The equations (3.1) and (3.2) are combined to the final "building score":

$$P_{building}(\Delta_x, \Delta_y) = P_h(\Delta_x, \Delta_y) + \lambda_d P_d(\Delta_x, \Delta_y). \tag{3.3}$$

In our experiments, we used simple maximum heights, but a more robust score can be constructed by adopting robust statistics (rank-order statistics) where the maximum value is replaced, e.g., by the value that is higher than 95% of all points. The maximum value performed well in our experiments, and we fixed the balancing factor $\lambda_d = 1.0$ for equal weighting for the height and density scores.
On the range image $\mathcal{P}_{building}$, an interpolation is required in order to fill holes caused by occlusions, missing scan lines and LiDAR back projection scatter. We applied morphological operation (e.g. close and erode) to merge neighboring point and filling holes in the binary range images [P2]. The morphological interpolation does not create new regional maxima, furthermore it can fill holes of any size and no parameters are required.
The rule-based segmentation of buildings is achieved by computing the building score in (Equation 3.3) to the cells of size $0.25m \times 0.25m$ (the same as before) and thresholding each cell by $\tau_{building} = 1.80$. This generates a binary x-y map for which we compute the standard shape compactness features for each connected shape as:

$$P(S_i) = \frac{\pi \cdot \text{diameter}^2(S_i)}{4 \cdot \text{area}(S_i)} \ . \tag{3.4}$$

Again $P(S_i)$ score is thresholded by $\tau_{P(S_i)} = 15$ and the binary label is backprojected to each 3D point within each cell. Note that this process is executed for a point set from

[1] Using 10 lowest z-value points is to make the MZV estimation insensitive to outliers.

which the street surface points have already been removed $\mathcal{P}_{other}$ and this rule-based step creates another set $\mathcal{P}\prime_{other} = \mathcal{P}_{other} \setminus \mathcal{P}_{building} = \mathcal{P} \setminus (\mathcal{P}_{road} \cup \mathcal{P}_{building})$.
After the initial segmentation, the remained point clouds are classified based on super-voxels features correctly. The super-voxel classification goal is to process the remained point cloud $\mathcal{P}\prime_{other}$ related to thin and small objects. The first step is 3D point-wise agglomerative clustering that over-segments the input point cloud to voxels. The clustering algorithm incrementally picks a random seed point, adds points to the seed cluster to construct a voxel until no more points pass a distance-based merging rule, and then pick a new seed point until all points have been processed. For the random seed point $\mathcal{P}_i$ new points $\vec{p}_j$ are added $\mathcal{P}_i = \mathcal{P}_i \cup \vec{p}_j$ if they pass the distance rule,

$$\min_i \text{dist}(\vec{P}_i, \vec{p}_j) \leq \tau_{voxel} \ , \tag{3.5}$$

where $\text{dist}(\cdot,\cdot)$ is the minimal distance between a set and a point, and the distance threshold is set to $\tau_{voxel} = 0.005m$. The procedure of *super-voxelization* is to merge those voxels that are close to each other and share similar orientation. Formally, the proximity between two voxels $\mathcal{P}_i$ and $\mathcal{P}_j$ is defined as

$$\min_{i,j} \text{dist}(\mathcal{P}_i, \mathcal{P}_j) \leq \tau_{sv\_prox} \ , \tag{3.6}$$

which is equivalent to the minimum-link distance rule in agglomerative clustering. The surface orientation is computed using the PCA method in [43] for each voxel and two voxels are combined if their normals are similar:

$$\arccos\left(\text{norm}_{PCA}(\mathcal{P}_i), \text{norm}_{PCA}(\mathcal{P}_j)\right) \leq \tau_{sv\_orient} \ . \tag{3.7}$$

We set the super-voxelization thresholds to $\tau_{svoxel1} = 0.01m$ and $\tau_{svoxel2} = 15$ which produce high quality super-voxels on all our datasets. The two thresholds with intuitive physical interpretation avoid setting the number of clusters that would depend on the point cloud's size.
The automatically generated super-voxels can be classified by computing the popular 3D shape descriptors as features [67, 68], but we found these slow to compute, and due to variance in points density, their robust usage would require re-sampling, which is a slow procedure as well. Instead, motivated by the success of features with true physical meaning in voxelization and super-voxelization, we adopt several fast-to-compute physical measures as features. The selected features are listed in Table 3.1.
The features are fed to the boosted decision tree classifier [158] which is extremely efficient and produces high accuracy for multi-class classification tasks. The boosting is based on minimizing the exponential loss:

$$\sum_{i=1}^{M} \exp(-y_i f_\lambda(\vec{x}_i)) \tag{3.8}$$

where $\vec{x}_i$ are the input features and y_i the ground-truth class labels and $f_\lambda(\cdot)$ is the estimated label constructed from

$$f_\lambda(\vec{x}) = \sum_{j=1}^{N} \lambda_j h_j(\vec{x}) \tag{3.9}$$

where $h_j(\cdot)$ is a weak learner and λ_j its corresponding weight parameter. Selection of the weak learners and optimization of the weights to minimize the loss function can be done

Table 3.1: Geometric and photometric primitives used to classify super-voxels into pre-defined categories [P1], [P2].

Feature	Motivation
Geometric features:	
Area	small vs. large objects
Edge ratio	minimum and maximum edges
Max edge	longest dimension
Covariance	overall shape
Location and orientation features	
Height above road	
Distance to street	Horizontal distance to the car GPS
Normal angle	With respect to the surface orientation
3D and photometric texture features	
Mean intensity	Overall reflection property of the voxel
Density	Density of the points
Planarity	Average distance to the best fitted plane

efficiently by parallel updating which is faster than the sequential-update algorithm [158], but we adopted the sequential version due to its simplicity and wide-spread availability. In the experiments, we used a forest of 10 decision trees with each of them having 6 leaf nodes and this classifier leads to satisfactory classification results for the benchmark datasets used in our work.

Results and discussion

Our methodology is evaluated on two MLS LiDAR datasets: NAVTEQ True [P1] and Paris-Rue-Madame datasets [159]. A description of the datasets can be found in the experiment section of [P1] and [P2].

NAVTEQ True dataset, collected from Chicago, Boston, and Paris cities, covers roughly 2.4 km of the road altogether. Seven semantic object classes are defined to label the scenes: road, tree, building, car, traffic sign, river, and pedestrian. The three cities' point clouds are split into two portions: the training and the testing sets. The 70% of the total street length is selected for training and 30% for testing.

The confusion matrix in Table 3.2 shows that the average accuracy (over all classes) is about 83%, with rule-based classification accuracies above 88%. Relatively low accuracies were reported for certain classes, e.g., river (73%), traffic sign (73%), and pedestrian (78%). These challenging cases are ascribed to the lack of sufficient training samples for each class.

The main contribution of this section is about achieving high performance within reasonable computing time. Considering the large-scale LiDAR datasets, we believe that fully supervised classification methods are computationally too expensive.

To confirm this claim, we conducted an ablation study. We performed super-voxel-based supervised training and classification for all point clouds, e.g., we switched off the rule-based processing stage. The results are reported in Table 3.3. The computing time is measured on Intel (R) Core(TM) i7-4710 MQ 2.5 GHz CPU with 32 GB RAM.

The results show that without the rule-based segmentation step, the supervised classifier

Table 3.2: Confusion matrix of our method for classification of the NAVTEQ True dataset [P4].

	Building	*Road*	*River*	*Car*	*Tree*	*Tr. sign*	*Pedestrian*
Building	0.885	0.083	0.000	0.000	0.115	0.000	0.000
Road	0.041	0.958	0.000	0.003	0.015	0.001	0.000
River	0.000	0.249	0.733	0.000	0.000	0.000	0.000
Car	0.000	0.018	0.000	0.847	0.000	0.007	0.000
Tree	0.004	0.001	0.000	0.000	0.897	0.007	0.000
Tr. sign	0.000	0.002	0.000	0.000	0.113	0.735	0.002
Pedestrian	0.000	0.049	0.000	0.000	0.008	0.000	0.782

must construct and classify 7.5× more voxels, and thus the computation time is 6.3× longer and requires much more memory usage. Moreover, without rule-based processing, the classification accuracy degraded significantly, partially due to the connectedness problem, i.e., buildings and road surfaces are mis-segmented with other objects. In contrast, removing building facades and road surfaces created better-isolated point clouds and improved classification accuracy from 75% to 86%.

Table 3.3: Computing times of our method with and without the rule-based steps. Without the rule-based step all points are classified using the super-voxel and boosted decision tree method [P4].

Our Method	*# of voxels*	*Comp. time (mins)*	*Overall accuracy*
w rule-based	32,891	46	86%
w/o rule-based	246,548	291	75%

Paris-Rue-Madame dataset [159] is used to compare proposed method with other recent works on 3D segmentation and classification. The point-wise classification results for our method and for the two proposed methods by Serna and Marcotegui [160] and Aijazi et al. [161] are presented in Table 3.4. Our method achieved an average accuracy of 94.1% with notable margin of 22.2% and 8.5% with respect to existing methods [160] and [161] respectively.

Table 3.4: Comparison of proposed method to other reported results on 3D point cloud classification with the Paris-Rue-Madame dataset [P4].

Method	*Building*	*Road*	*Tr. sign*	*Car*	**Class AVE ACCY**
Aijazi et al. [161]	0.914	0.901	0.710	0.900	0.856
Serna et al. [160]	0.986	0.940	0.000	0.950	0.719
Our	**0.991**	**0.950**	**0.841**	**0.982**	**0.941**

3.1.1 Semantic Segmentation of TLS LiDAR Point Cloud

It is notable that our algorithm was originally designed to analyze and process MLS LiDAR data [P2]. One of the main advantages of the proposed method is that it easily adapts to different types of LiDAR datasets such as Airborne Laser Scanning (ALS) and Terrestrial Laser Scanning (TLS) data without major modification as long as the point units are in a metric system (thresholds are set in meters).
It also has broader implications for on-board TLS sensors in autonomous driving vehicles and other robotic platforms [P4]. In this configuration (see Figure 2.3), the sensor scans the same space with every rotation. The LiDAR sensor produces a real-time 360-degree point cloud map from a more limited horizontal field of view and very low density. The on-board computer system needs to simultaneously recognize and locate people's positions and other objects around the vehicle.

Results and discussion

We evaluated our method on the TLS Velodyne LiDAR dataset [162]. TLS Velodyne contains 3D point clouds in the local coordinate system of the LiDAR sensor. We compare our method to Lai and Fox [162].
This dataset includes high-quality 3D point clouds from ten different scenes collected by a Velodyne LiDAR system mounted on a car navigating through Boston streets. Due to this dataset's specific nature, we evaluate our proposed method using each LiDAR rotation as a single scene. The average point density is about 12 points/m2, and the total number of points in each scene is approximately 70,000. We picked seven scenes for training and the three remaining scenes for testing similar to [162], and report per class average precision, and F_1 score computed as:

$$F_1 = \frac{2 \times recall \times precision}{recall + precision}. \tag{3.10}$$

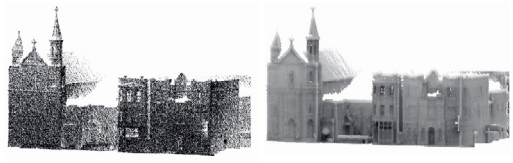
Table 3.5: Comparison of our method to other reported results on 3D point cloud classification with the TLS Velodyne dataset [P4].

Measure	**Method**	*Tree*	*Car*	*Tr. sign*	*Pedestrian*	*Fence*	*Building*	*Class AVE ACCY*
Precision	Lai and Fox [162]	0.83	0.91	**0.80**	0.41	0.61	0.86	0.73
	Our	**0.89**	**0.95**	0.72	**0.88**	**0.85**	**0.95**	**0.87**
F_1	Lai and Fox [162]	0.76	0.79	**0.69**	0.47	0.42	0.91	0.67
	Our	**0.85**	**0.93**	**0.69**	**0.88**	**0.80**	**0.95**	**0.85**

Table 3.5 shows that for five out of six classes, our method is clearly more accurate, and our F_1 score for each class is better than the average F_1 score of Lai and Fox.
In summary, the results in Table 3.5, 3.4 and 3.2 confirm that the proposed geometric and photometric features presented in Table 3.1 can effectively increase the features' discrimination power. The proposed method is efficient by combining fast rule-based processing for street surface and building segmentation and super-voxel-based feature extraction and classification for the remaining scene elements. The results in Table 3.3 exhibit that the rule-based stage makes computing significantly faster than classifier-only and improves the segmentation accuracy.

3.1.2 3D HDM Reconstruction

The research of 3D digital city modeling is becoming popular [4, 163–167]. With MLS data of high point density, not only the different types of objects (e.g., ground, roads, trees, buildings, utilities) can be classified and detected, but also the detailed city model could be 3D regularized and reconstructed geometrically [4]. In this section, we aim to answer the following research question: How to make a 3D map of urban areas using currently available measurement technologies?
Figure 3.2 presents our contributions in publications [P3], a fast rendering-friendly method that reconstructs a 3D HDM model in two main stages.
In the first step, we utilize the enhanced ShadVis algorithm [168] rendering the building facades with high-quality details. The algorithm calculates a point cloud's illumination with the light coming from a theoretical sphere or hemisphere around the object. The graphics hardware rendering pipelines have been designed for polygons, but in our case, it is computationally more attractive to render only the points (Figure 3.2a). Therefore, we adopt the simple but practical algorithm in [158].
The second step is to apply methods to fit pre-designed template models to non-building labeled point clouds (Figure 3.2b). The fitting method is categorized into two types, depending on whether an object's orientation in question plays an essential role in rendering.
The first fittings type is related to the object groups, in which their pre-designed mesh structure orientation is not essential, and their object models will be based on their position and dimension only. The first object type fitting includes pedestrians, trees, and sign symbols. In this case, for each separated labeled point cloud $\mathcal{P}_i$, the center and its 3D bounding boxes (boundaries) will be determined. The best isodiametric meshes to the point cloud are localized, based on the size of the existing pre-designed library meshes. This is a similarity transformation of estimated isotropic scale s and transformation $\vec{t} = (t_x, t_y, t_z)^T$. It is also possible to estimate a similarity transformation where scale is applied to each dimension $\vec{s} = (s_x, s_y, s_z)^T$.



(a) Example of 3D points $\mathcal{P}_{building}$, and results of our point rendering algorithm

(b) Example of a 3D car model fitted and rendered to a point cloud

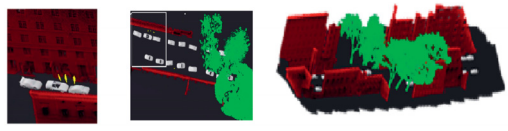


Figure 3.2: Example of 3D HDM [P3].

The second type of object also requires an x-y orientation angle θ and is needed for different vehicle types (bus, bike, car). First, the center of the pre-designed mesh is computed, and the point cloud will be matched, and then the corresponding model will be chosen from the library based on the dimension of the vehicle 3D bounding box. The Iterative Closest Point (ICP) algorithm is automatically applied to refine point clouds' registration with the desired mesh. The ICP algorithm that we apply optimizes the Root Mean Square (RMS), the distance between closest point pairs of the models' vertices to the point cloud:

$$err_{RMS}(\mathcal{P}_{model}, \mathcal{P}_i) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} ||\vec{p}_{n,model} - \vec{p}_{n,\mathcal{P}_i}||^2} \tag{3.11}$$

The scene prior knowledge reduces the number of possible vehicle orientations as the road surface is determined and only rotations around the road's z-axis are considered.

3.1.3 Semantic Segmentation of Street View Images

The 2D-3D correspondences between the image plane and the 3D point cloud are described in Section 2.2.4. Since the point cloud and the image are pre-registered and defined in the same world coordinate system, we could link the 2D-3D correspondences features or labels as illustrated in Figure 3.3. Depending on the way to generate semantic segmentation of images, the methods can be divided into two main categories: direct and indirect approaches.

In our context, the direct approach means all LiDAR points are transformed into each image and mapped to the closest super-pixel [12]. By doing this, those 3D points that are projected within a specific super-pixel are identified. Then image segmentation is processed on the 2D image patches by extracting 3D features (See Table 3.1) for different patches. In our experiment, these 3D features are used to train the boosted decision tree classifier [P1].

In the indirect approach, semantic labels of 3D point clouds that are classified based on 3D voxel features are mapped to the image plane. In this case, input images are over-segmented into super-pixels, and each image plane super-pixel is associated with a collection of labeled 3D points. The majority vote label of 3D points projected to the same super-pixel is selected as the super-pixels label.

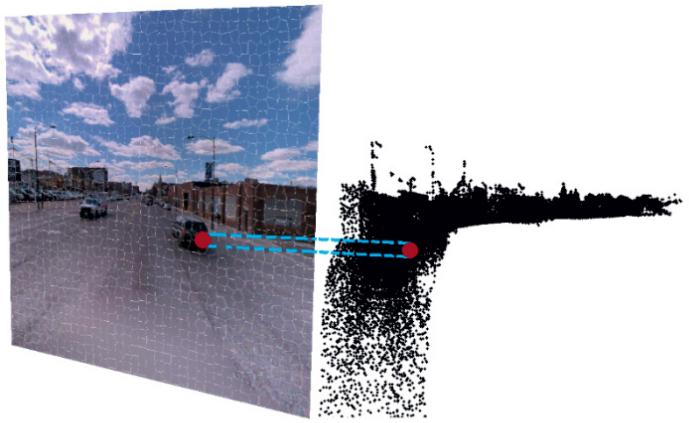


Figure 3.3: 2D-3D association [P3]

Results and discussion

In this section, we conducted experiments to answer the following research questions: How to label street view images using 3D LiDAR point cloud cues? How much LiDAR intensity feature affects the performance of the segmentation model?
We use the NAVTEQ True dataset [P1] to evaluate both direct and indirect image semantic segmentation algorithms' performance. For direct 2D semantic segmentation, we selected 200 images captured from three cities and split them into training (150) and testing (50) sets. 10 semantic object classes are defined to label the images: sky, building, fence, road, side-walk, tree, car, traffic-sign, pedestrian, and river. Some of these classes, e.g., building, road, and, car are common objects in street view images, while others such as river and fence occur less frequently. the statistics of occurrences of each class are presented in [P1]. As can be seen in Table 3.6 our algorithm performs well on most per class accuracies with the highest accuracy 96% achieved for the sky and road, and lowest for traffic-sign (17%). For evaluating indirect 2D semantic segmentation, we generated

Table 3.6: Confusion matrix of pixel-wise accuracies of our method for direct 2D semantic classification of the NAVTEQ True street view images [1].

	Sky	*Building*	*Fence*	*Road*	*sidewalk*	*Tree*	*Car*	*Tr. sign*	*Pedestrian*	*River*
Sky	0.96	0.02	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
Building	0.04	0.90	0.01	0.00	0.02	0.03	0.00	0.00	0.00	0.00
Fence	0.05	0.37	0.49	0.00	0.01	0.03	0.04	0.00	0.00	0.01
Road	0.02	0.00	0.00	0.96	0.01	0.00	0.01	0.00	0.00	0.00
sidewalk	0.02	0.04	0.00	0.12	0.77	0.01	0.04	0.00	0.00	0.00
Tree	0.06	0.17	0.00	0.00	0.03	0.74	0.00	0.00	0.00	0.00
Car	0.05	0.11	0.02	0.35	0.11	0.01	0.35	0.00	0.00	0.00
Tr. sign	0.08	0.02	0.00	0.05	0.60	0.04	0.03	0.17	0.00	0.01
Pedestrian	0.10	0.34	0.00	0.01	0.21	0.03	0.03	0.00	0.22	0.06
River	0.48	0.06	0.01	0.01	0.05	0.05	0.01	0.00	0.00	0.33

2D ground-truth by back projecting the ground-truth 3D labels to the corresponding street view images in 500 randomly selected images in the NAVTEQ True dataset [P1]. The back-projection results were manually verified and corrected. In indirect semantic segmentation, the fence class is redefined as building, and ground surfaces, car paths, and side-walk are merged as road. Moreover, the image super-pixels without any label are labeled as "sky". A description of the datasets can be found in the experiment section of [P1] and [P4].

Table 3.7: Confusion matrix of the pixel-wise accuracies for 2D semantic segmentation of the NAVTEQ True images. The semantic labels of 3D point clouds that are classified based on 3D voxel features are mapped to the image plane [P4].

	Sky	*Building*	*Road*	*Tree*	*Car*	*Tr. sign*	*Pedestrian*	*River*
Sky	0.960	0.020	0.000	0.020	0.000	0.000	0.000	0.000
Building	0.030	0.870	0.024	0.075	0.000	0.000	0.000	0.000
Road	0.000	0.015	0.920	0.000	0.065	0.000	0.000	0.000
Tree	0.000	0.280	0.080	0.640	0.000	0.000	0.000	0.000
Car	0.050	0.000	0.250	0.020	0.680	0.000	0.000	0.000
Tr. sign	0.010	0.280	0.090	0.000	0.000	0.370	0.250	0.000
Pedestrian	0.010	0.340	0.020	0.020	0.000	0.330	0.280	0.000
River	0.000	0.050	0.250	0.050	0.000	0.000	0.000	0.650

Table 3.7 presents pixel-wise segmentation accuracies results. The building, sky, and road sections were accurately labeled in ($\geq 85\%$ accuracy). In contrast, the pedestrians and traffic signs classes were poorly segmented. It could be interpreted that there are not many examples to train the classifier, and consequently, it makes misclassifications to the more frequent classes.
Both direct and indirect approaches have shown good performance when using just geometrical features. However, the indirect approach demonstrated superior computational efficiency.

Impact of LiDAR Intensity Feature in 2D Semantic Segmentation: LiDAR systems provide positioning information and reflectance property, referred to as intensity, of laser-scanned objects. The intensity feature is utilized in our workflow, combined with other features, for semantic scene parsings. The points' median intensity in each batch/super-voxel is used to train the classier. The bar chart in Figure 3.4 indicates that using intensity information from LiDAR data, the direct image segmentation model's robustness is increased for certain object classes. e.g., building, car, and signs-symbol and pedestrian.

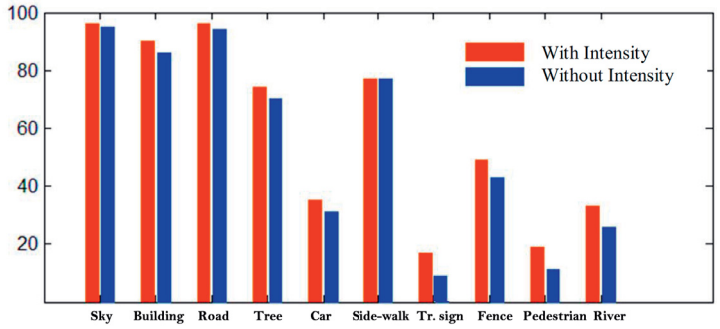


Figure 3.4: Impact of LiDAR intensity feature in 2D semantic segmentation [P1]

3.2 Depth-Augmented Stereo Panorama (DASP) Using Stereo Matching

The general scenario that we target is to format and store the raw data from a 360-degree camera into the DASP representation to support the 6-DoF head-motion parallax. In this method, the left and right-eye equirectangular pixels can be projected into the 3D scene using their corresponding depth values, and a new synthesized stereo view can be rendered for a given head position. Our software pipeline is generic and can handle a variety of camera geometries and configurations.

This section describes our DASP representation pipeline, which takes as input four fish-eye images and produces a stereo pair of stitched equirectangular images along with corresponding depth maps (see Figures 3.5d and 3.5e). The goal of stitching is to blend raw images even though they have been photographed from different perspectives.

As shown in Figure 3.5a, the inputs to our stitching algorithm are calibrated fish-eye images captured by a single OZO camera. A fish-eye lens is a special wide-angle lens that can produce 180-degree or even larger FoV by bending the incident lights.

OZO camera consists of four 2048×2048 ($2K \times 2K$) sensors, and each sensor is paired with a wide-angle 195-degree FoV lens and captures images that overlap adjacent sensors to provide natural stereoscopic views. By design, the VR camera captures stereoscopic 3D imagery through approximately 260 degrees (+/- 130 degrees from center) oriented to the camera front and sides, with the balance of the 360-degree FoV captured with monoscopic imagery. Each lens captures approximately 60 degrees of an overlapping area with the adjacent lenses. This overlap allows for blending the two images together.

The equirectangular projection is the most common format used for processing and storing VR videos. It can be described as unwrapping a sphere on a rectangular plane with dimensions 360×180 degree. Each fish-eye image I_i (where $i = 0..n$, and n being the number of lenses with $n = 4$ for our camera) is assumed to have an aperture of at least 180 degree. Assuming that all the objects in the scene are sufficiently far away, the whole equirectangular projection of I_i images should roughly portrait the same pixels in the overlap areas (see Figure 3.5b).

Given the calibrated camera parameters, our proposed method picks two adjacent lenses and rotates them to face the same direction, and in the process, also undo lens-distortion [P5]. So the resulting two images are partial fish-eye images, which only differ by the origin from which they were captured. The overlap areas between adjacent lenses I_i^j in spherical coordinate are shown in Figure 3.5c, where i denotes the rotated fish-eye image

I_i with respect to the view shared with I_j. This way, we obtain six images I_0^1, I_1^0, I_1^2, I_2^1, I_2^3, and I_3^2 from four input fish-eye images.
Left and right-eye panoramas are stitched from the overlapping portions by assembling the left images, interpolated images, and the right images, where the interpolated images represent camera views from positions between the original views. These interpolated areas are referred to as the "stitched" or "seam" and marked by red rectangles in the stitched panorama in Figure 3.5d and 3.5e. Figure 3.6 shows the blocks for stitching two adjacent lenses using flow-based view interpolation.

The seams can be changed in both their location and width. The seams can range anywhere from around 50 degrees to 2 degrees in width. The seam's middle point can also be adjusted by approximately 50 degrees, depending on the specified width. It may be beneficial to adjust the seam areas based on the image's subject [P5]. Although a wider seam can reduce the visible seams, it requires more computational cost. We chose to set seam locations in degree for left panorama at [-55.0, -30.0], [30.0, 55.0], and [90.0, 115.0] and for right panorama at [-55.0, -30.0], [30.0, 55.0], and [-115.0, -90.0] intervals.

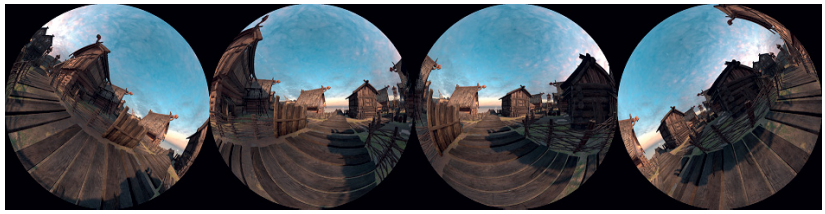
Our DASP technical approach follows the following steps:

1. We first estimate the forward and backward flow fields for each adjacent image pair. We describe our flow estimation in terms of the reference image I_i^j (the image for which flow is computed) and a neighboring image I_j^i to obtain flow maps F_i^j (see Figure 3.7b), though this process is performed 6 times for each single video time-step (3 overlapping image pairs, forward and backward).
   As part of the stitching algorithm, we introduce a CNN-based optical flow and compositing method designed to support high-quality view synthesis in VR. We will explain details about the process of optical flow and the network architecture in Section 3.2.1.

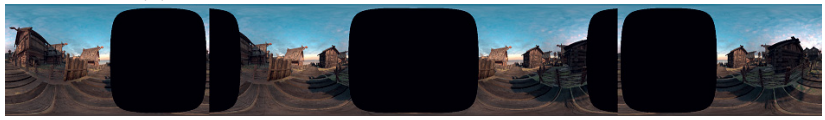
2. The optical flows for a N number of frames are stored, and a temporal filter (mean filter) is applied to the flow maps at every pixel. This method of enforcing temporal consistency by correlating each pixel to its nearby pixels in the video sequence implicitly reasons for object motion by assuming motion is small and temporally smooth for images with the same color, which works well in practice. It avoids the need for estimating temporal flow across adjacent frames, as is often required by other techniques [118]. In a video sequence, this means that the relative ordering of fragments may change from frame to frame, causing temporal flickering.

3. We convert the optical flows to depth maps using camera parameters. Stereo depth is needed over the image's entire size to re-render the scene from any new viewpoint. Actual depth maps are usually obtained from two rectified images taken by stereo cameras. However, they can also be estimated by unrectified stereo cameras as long as they depict the same objects in the scene, and cameras' parameters (both extrinsic and intrinsic) are known. As described in Section 2.2.3, to compute the depth value of a pixel, we first project the corresponding point in the target image to the epipolar line and then triangulate the 3D point to retrieve the depth in the reference images. We denote D_i^j as estimated depth from flow F_i^j.

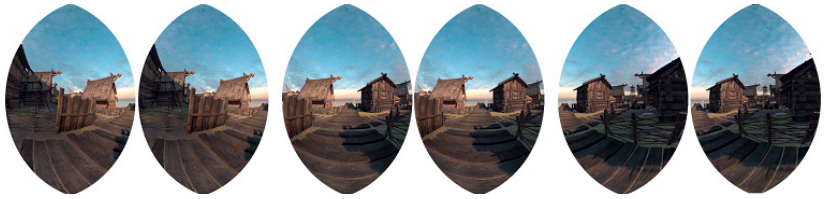
4. From two adjacent lenses, given dense optical flows between them, we can transform both images into an intermediate position and afterward superimpose and blend both warped images to reconstruct a single interpolated viewpoint. A common strategy, which is also the basis for our view interpolation, is to warp the images in 2D space using image correspondences computed as optical flow fields in the



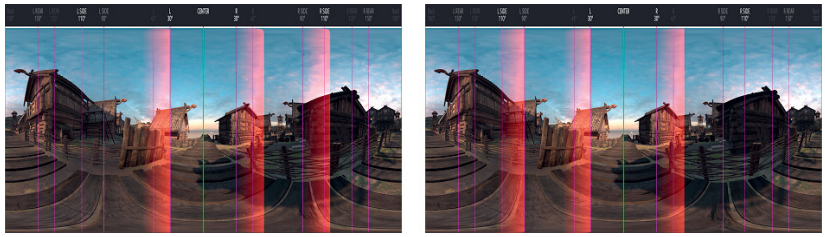
(a) Input fish-eye images, from left to right I_0,I_1,I_2,I_3.



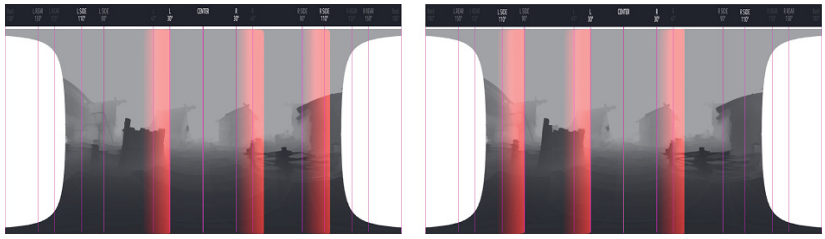
(b) Equirectangular mapping of each lens.



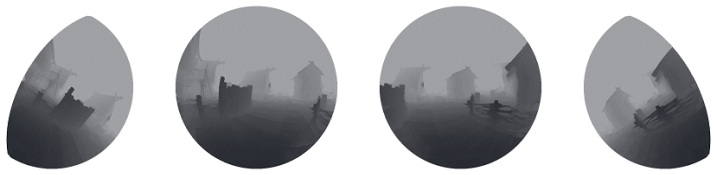
(c) The overlapping portion of adjacent lenses I_i^j used to construct the stereo panorama and to estimate the scene's depth.



(d) Left and right-eye panoramas. Seams are shown in red.



(e) The parallax information from the overlapping area of adjacent lenses is used to derive the 260 degree per-pixel depth maps.



(f) Per-lens depth maps.

Figure 3.5: Illustration of depth-augmented stereo panorama and stitching procedure. The purpose of computing depth is to support high-quality novel view synthesis in VR [P5].

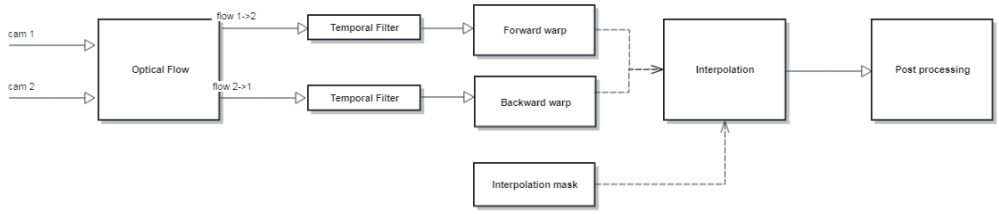


Figure 3.6: Structure of the flow-based interpolation pipeline [P5].

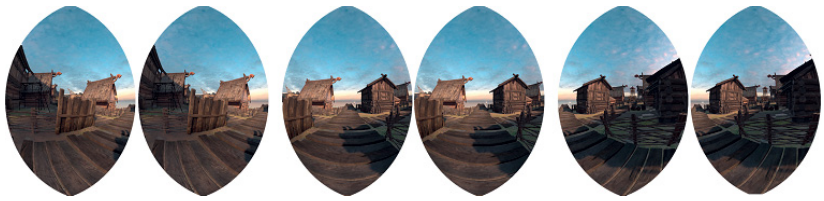
overlapping image regions.
Note that the optical flows calculated in the first step are asymmetric, e.g., the forward flow estimated from I_i^j to I_j^i (i<j) is not the same as the backward flow from I_j^i to I_i^j. The forward flow has a coordinate system corresponding to the pixels in I_i^j, and the backward flow follows the coordinate system given by the pixels in I_j^i. To generate stitched equirectangular images/depths, we need to warp the adjacent images/depths to match at intermediate frames. This mapping into the equirectangular plane can be performed very efficiently by assuming that the angular distance between the intermediate view and real images varies linearly with the motion of a pixel along its flow trajectory. Under this assumption and given the desired seam center location in degree Θ, when interpolating between two images, we need to find the fraction of the way between the images to warp the flows:

$$\widetilde{F_i^j} = \frac{\Theta - c_i}{C_j - C_i} F_i^j \quad (i < j) \qquad forward\,warping \tag{3.12}$$

$$\widetilde{F_j^i} = \frac{c_j - \Theta}{C_j - C_i} F_j^i \quad (i < j) \qquad backward\,warping \tag{3.13}$$

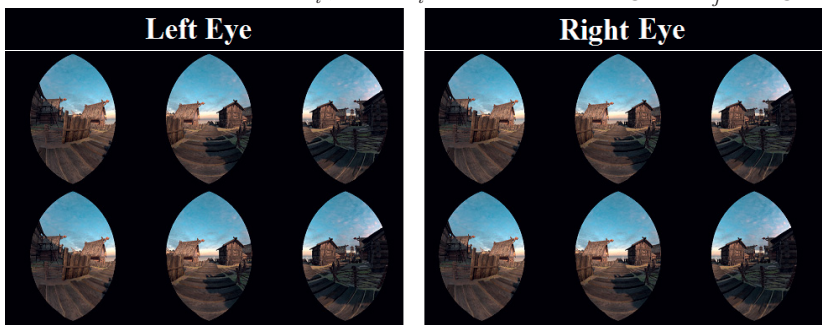
where c_i and c_j are angular positions of adjacent cameras and Θ (c_i<Θ<c_j), is the seam center position in degree. The Equations 3.12 and 3.13 correspond to new optical flow fields, used for warping images and depth to seam areas from the left and right cameras positions, respectively. The first rows in Figure 3.7d and Figure 3.7c demonstrate the warped depths and warped images from left camera using forward mapping F_i^j (i<j). The warped backward images and depths are estimated similarly and shown in second rows. In the warping procedure, multiple flow vectors may hit the same pixel. In that case, at the location where multiple pixels collide, a maximum flow (corresponding to minimum depth) is retained.

5. Finally, we blend the two warped images/depths using an interpolation mask (Figure 3.7e). In the seam areas, this mask has values which are linear ramp from 0 to 1, and for non-seam areas, it contains a fixed value of either 0 or 1.
This ensures a smooth transition between the interpolated images. The image closer to the position is given more weight since there are more artifacts when the images are shifted by a large amount. The image shifted by a smaller amount is estimated better and is given more weight while computing the blended image. Using the forward warp, backward warp, and interpolation mask, the interpolation area will be generated. The final pixel will either be a forward warp pixel, backward pixel, or in case of holes, a blend of both warps.

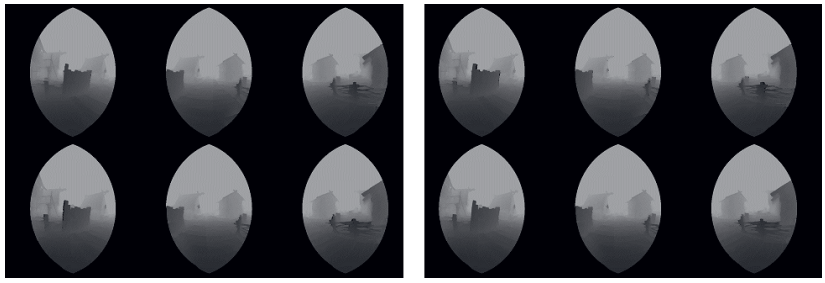


(a) The overlapping portion of adjacent lenses I_i^j

(b) Forward and backward flow field F_i^j, where I_i^j is the reference image and I_j^i is neighbor view.



(c) The top row of six images shows forward warp of left images I_i^j (i<j) using (F_0^1, F_1^2, and F_2^3) flows, and the bottom row corresponds to backward warped images. Note that the first three warped images in each row are used for left-eye panorama and remained three for right-eye panorama.



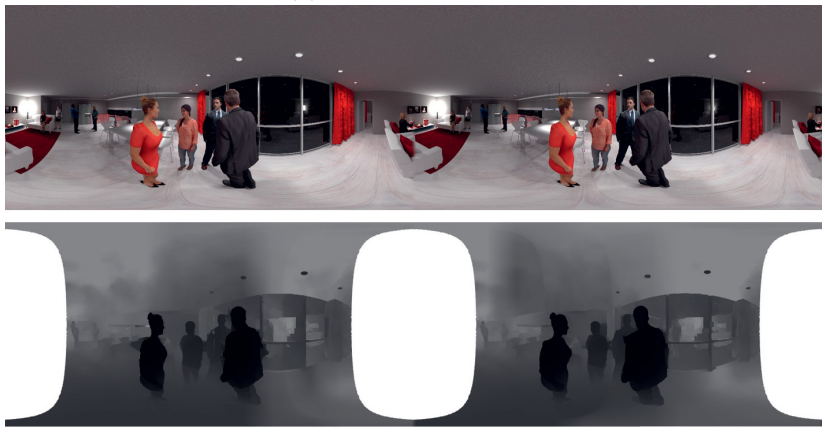
(d) Depth maps derived from optical flow using lenses parameters

(e) Seam stitching and blending for both depth maps (top row) and RGB values (bottom row). The final pixel will either be a forward warp pixel, backward pixel, or a blend of both warps.

Figure 3.7: Flow-based view synthesis and depth reconstruction [P5].

For rendering DASP contents in VR glasses, we assume that the depth of every scene point p is known with respect to the current viewpoint. With the known depth and the ray direction in conjunction with known physical camera geometry and lens characteristics, we can calculate the 3D position of the world point and find projection to any new viewpoint. Figure 3.8c shows an example of DASP content generated by 360-degree camera lenses and corresponding reconstruction of 3D scene.
VR Immersive Rendering (IR) via a HMD is done by converting the depth map into a 3D polygon mesh and applying the associated color image as a texture, then projecting the 3D mesh back into the user's virtual view. This way, as the viewer moves his or her head away from the depth map's origin, the depth map transforms according to the reconstructed 3D information. In a perfect depth map, all objects in the image should transform exactly like their real-world counterparts, while areas containing incorrect depth information will distort noticeably.

(a) input fish-eye images



(b) DASP content and corresponding fish-eye depth maps

(c) Textured point cloud, rendered by DASP content

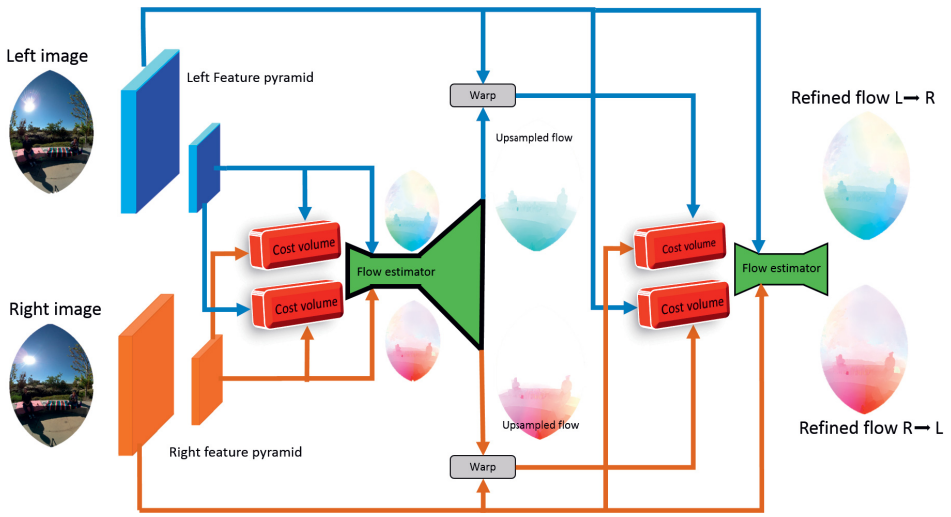
Figure 3.8: Example of depth-augmented stereo panorama visualization. Given the camera model and depth map, it is possible to project the image texture outward to create a colorized point cloud corresponding to the 3D surfaces seen by the sensors [P5].

3.2.1 Bi-Directional Flow Estimation Network

In the proposed DASP pipeline, six forward and backward dense flow fields must be computed to construct depth maps and precisely synthesize the transition regions.
Current state-of-the-art stereo algorithms are often very computationally expensive and have difficulty with textureless areas, reflective surfaces, thin structures, and repetitive patterns. A number of these challenging problems for stereo algorithms would benefit from the knowledge of scene geometry.
Our contributions in publication [P5] covered in this thesis aims to answer the following research questions and meets the set objectives: Can the entire stereo vision problem be formulated with deep learning using our understanding of stereo geometry? Can we build a compact and effective CNN model for optical flow estimation to be used for view compositing? How much the performance of optical flow methods affect DASPs' quality? Can we improve the performance of a CNN model by training it on the synthesized VR dataset?
This section's main contribution is an end-to-end deep learning method to estimate dense forward and backward optical flow from an unrectified image pair. Our intention is not to naively construct a machine learning architecture as a black box to model stereo. Instead, we advocate using insights from many decades of multi-view geometry research [142, 143, 145, 169] to guide model architectural design. Hence, we build our model by developing differentiable layers representing each major component in traditional stereo pipelines.
Our model's key feature is to exploit the symmetry properties that characterize optical flow and occlusions in the stereo images. The first symmetry feature we consider is bi-directional motion consistency, i.e., motions of corresponding pixels visible in both views should be the inverse of one another. The occlusion/dis-occlusion symmetry relationship is the second property we aimed to exploit, i.e., occlusions in the forward direction correspond to dis-occlusions in the backward direction and vice versa.
In this work, we augment CNN input with this forward-backward consistency information and provide both optical flow ground-truths (see Section 3.2.2) to constrain the model in a way that preserves our knowledge of the geometry of stereo vision. Therefore, by design, pixels are forced to be either visible and satisfy the bi-directional flow consistency or are identified as occlusions and provide background flow, based on occlusion/dis-occlusion symmetry.
The components of our bi-directional flow estimation network at two pyramid level is illustrated in Figure 3.9a. Given two input images I_L and I_R with H×W×C dimensions (where C = 3 is the number of input channels), we generate K-level pyramids of feature representations. Input features are extracted by a small encoder architecture made of 12 convolutional layers. To generate feature representation at the lth level F^l (where F^0_L= I_L and F^0_R= I_R), we use layers of convolutional filters to down-sample the features at the (l-1)th pyramid level, F^{l-1} , by stride 2. The left feature extractor tower is shown in Figure 3.9b1. At the lth level we warp the features of the two input images towards each other using up-sampled flows from (l+1)th level ($\widetilde{W}^{l+1}$), and repeat this process at each pyramid except the last one using Equation 3.14 :

$$F^l_{warped} = F^l(x + \widetilde{W}^{l+1}(x)) \tag{3.14}$$

where x is the pixel index. Next, we use the warped features to create two cost volumes, one for left-right matching and the other for right-left matching (red boxes at Figure 3.9a).
The cost volumes, which are further processed by the flow estimator networks, are defined



(a) Our stereo network's architecture (2 levels are shown) estimates bi-directional optical flows using coarse-to-fine pyramidal processing. Note that in practice, we use a 7-level pyramid network.

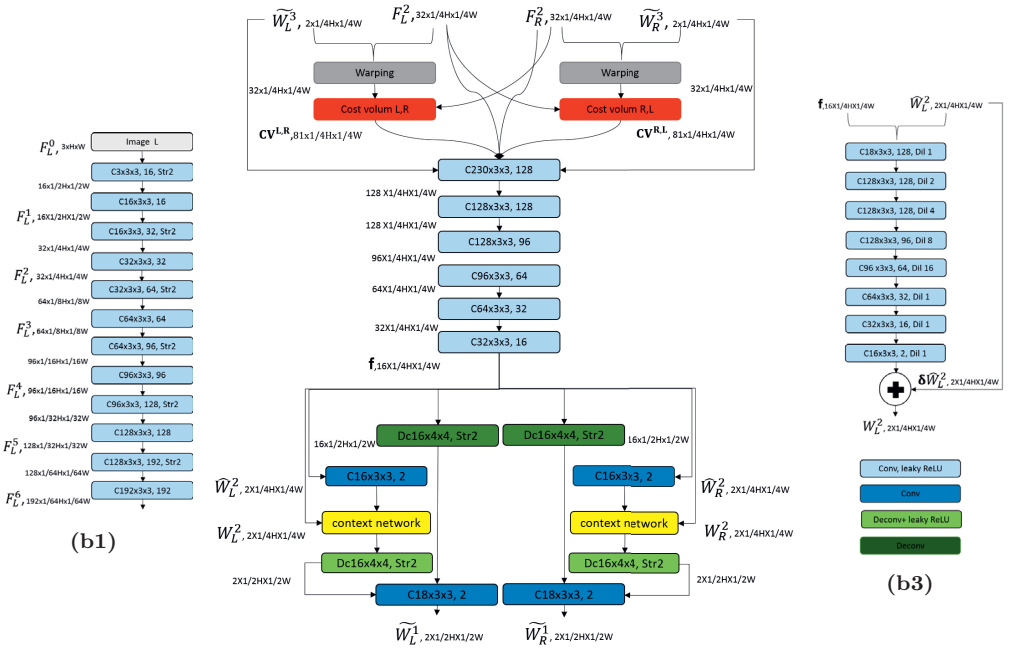


Figure 3.9: Summary of our end-to-end deep stereo regression architecture for bi-directional optical flow estimation. (a) Overview of the network architecture. The network takes an image pair as input and predicts both forward and backward optical flow using a 7 level pyramid setting. (b1) Feature extractor. (b2) Optical flow estimator. The siamese encoder-decoder architecture for predicting optical flows is shown at pyramid level 2. (b3) The context network used as post-processing to refine the optical flows. [P5].

as matching costs based on the correlation between features of one image and warped features of another image. The partial cost volume dimension at each level is $d^2 \times W^l \times H^l$, where d corresponds to a limited pixel motion range. Each level in our pyramid network has a simplified task relative to the full resolution flow estimation problem; it only has to estimate a small motion update to one higher level estimated flow field. Thus at each pyramid level, we assume that the motion is small, and we can set d=9 (search range of 4 pixels).

At each level of the pyramid we estimate the flow W^l using a Siamese encoder-decoder network. Our flow estimator network takes as input the skip connection from features of the left and right images (F_L^l, F_R^l), up-sampled optical flows from higher level (W_L^{l+1}, W_R^{l+1}) and concatenates them with left-right and right-left cost volumes. Based on these inputs it estimates both forward and backward flows (W_L^l, W_R^l) and up-samples them to the $\times 2$ resolution to be used at the next pyramid level ($\widetilde{W_L}^{l-1}$, $\widetilde{W_R}^{l-1}$). This setup at pyramid level 2 is shown in Figure 3.9b2.

The decoder decodes the joint feature representation, up-projects the encoder's output, and generates two up-sampled and refined optical flows. This way, we preserve fine local information provided in lower layer feature maps for each flow.

Finally, we use a CNN context network to exploit contextual information to refine the optical flows at two decoder branches according to [170]. From the optical flow estimator, context network takes the estimated flow ($\widehat{W}^l$), encoder output ($\mathbf{f}$) and generate a refined flow ($\widetilde{W}^l$) at each level. Figure 3.9b3 shows the context network, which is designed based on 7 convolutional layers with dilation.

We use a 7-level pyramid and set ł$_0$ to be 2, i.e., our model outputs a quarter resolution optical flow and uses bilinear interpolation to obtain the full-resolution optical flow. To train our network, we use the average End Point Error (EPE) loss [171] and apply the same multi-scale training loss proposed in [170, 172] to measure the point-wise losses to our outputs. Let ω be the set of all the learnable parameters at different pyramid levels in our model (the warping and cost volume layers have no learnable parameters) and W_ω^l denotes the flow field at the lth pyramid level estimated by the network. The supervised regression loss is defined as L2 norm ($\|.\|_2$) to penalize deviations from the respective corresponding ground-truth W_{GT}^l.

$$\mathcal{L}(\omega) = \sum_{l=l_0}^{L} \alpha_l \sum_x \|W_\omega^l - W_{GT}^l\|_2 + \lambda \|\omega\|_2 \tag{3.15}$$

According to [170, 172] the weights in the training loss (Equation 3.15) are determined empirically to be α_6 =0.32, α_5 =0.08, α_4 =0.02, α_3 = 0.01, and α_2 = 0.005. Note that the calculated loss for a small-sized prediction has more contribution to the total loss rather than the larger one. Thus, the weights in the training loss are set to be $\alpha_6 > \alpha_5 > \alpha_4 > \alpha_3 > \alpha_2$. The second term regularizes parameters of the model (L2 regularization) and the trade-off weight λ is set to be 0.0004.

Results and Discussion

Table 3.8 summarizes the quantitative results of our flow estimation method and state-of-the-art deep learning methods including FlowNet [172], SpyNet [173], FlowNet2 [174] and PWC-Net [170] on the MPI Sintel [170] and KITTI [175] benchmarks.

Most of the current state-of-the-art CNN networks have used FlyingChairs [172], and FlyingThings [176] for training their networks and then fine-tuned on Sintel and KITTI

datasets. Since our network needs both forward and backward ground-truth flow in the training phase and backward ground-truths were not available for these datasets, we trained our network with our dataset (see Section 3.2.2) and report numbers on Sintel and KITTI benchmarks without extra fin-tuning. Although our network predicts both forward and backward flows in one inference during testing, we report numbers from forward flow at the test split of benchmarks.

Table 3.8: Quantitative evaluation on public optical flow benchmarks. We report the average EPE for all benchmarks, except KITTI, where F1-all and Out-noc are used to benchmark KITTI 2015 and 2012, respectively. Out-noc presents the percentage of outliers with errors more than 3 pixels in non-occluded regions, whereas F1-all presents the percentage in all regions. For all measures, lower is better. Results are divided into methods trained with (−ft) and without fine-tuning. Entries in parentheses indicate methods that were fine-tuned on the evaluated datasets. *Our network uses different training data. Since Sintel and Kitti do not provide the ground-truth backward flow, we report numbers on the test sets and from the forward flow. The run times are measured based on one forward inference on the MPI-Sintel benchmark's final pass [P5].

Methods	**Sintel**				**KITTI**				**Run time (s)**
	(Clean)		(Final)		(2012)		(2015)		
	AEE		AEE		AEE	OUT-noc	AEE	F1-all	
	train	*test*	*train*	*test*	*train*	*test*	*train*	*test*	
PWC-Net [170]	2.55	-	3.93	-	4.14	-	10.35	33.67%	0.030
PWC-Net-ft	(2.02)	4.39	(2.08)	5.04	(1.45)	4.22%	(2.16)	**9.60%**	0.030
FlowNet2 [174]	**2.02**	**3.96**	**3.14**	6.02	**4.09**	-	**10.06**	-	0.123
FlowNet2-ft	(1.45)	4.16	(2.01)	5.74	(1.28)	4.82%	(2.30)	11.48%	0.123
FlowNetS [172]	4.50	7.42	5.45	8.43	8.26	-	-	-	0.186
FlowNetS-ft	(3.66)	6.96	(4.44)	7.76	(7.52)	-	-	-	0.186
SpyNet [173]	4.12	6.69	5.57	8.43	9.12	-	-	-	0.016
SpyNet-ft	(3.17)	6.64	(4.32)	8.36	(4.13)	12.31%	-	35.07%	**0.016**
Ours*	-	4.14	-	**4.76**	-	**3.16%**	-	9.74%	0.033

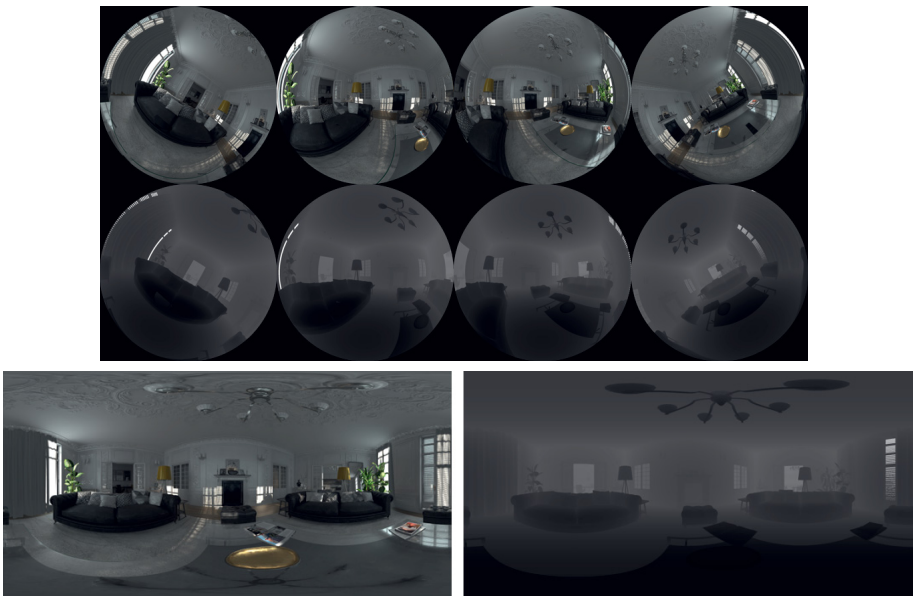
The timings have been measured on the same machine equipped with an NVIDIA Pascal TitanX GPU and measured on one forward inference of Sintel datasets (with 768 × 384 image size) excluding data loading/writing time.
The quantitative results in Table 3.8 show our method outperforms existing CNN learning methods and obtains the backward flow as an additional output at no cost. The results confirm that our method is also notably faster than most competing approaches. These findings satisfactorily answer the research questions and hence meet the objectives.
The closest method to our architecture is PWC-Net [170] an end-to-end regression. However, our method outperforms this architecture on KITTI 2012 and Sintel.
The proposed network's advantages become even more prominent as we look at the accuracy-runtime trade-off in our DASP pipeline. e.g., compared to PWC-Net, it is twice as fast. Our proposed network would be the best choice because it lets the network estimate forward and backward flows on one forward inference and achieve the best balance between accuracy and running time.
Our architecture more explicitly leverages geometry by getting both optical flow ground-truths. By building architectures that use this knowledge, we could simplify the learning problem.

3.2.2 Datasets:Unity Synthesized Scene

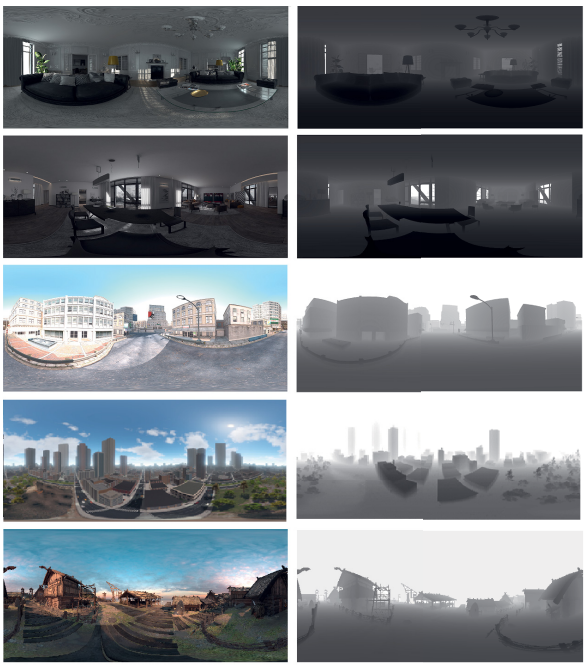
Numerous datasets [142, 177 – 179] exist with flow ground-truth, but none of them propose both bi-directional flow maps for supervision of our bi-directional flow model. Moreover, they often feature unrealistic and cartoon-like images because they are mostly generated by a mixture of real scenes and rendered scenes.
Recently Zioulis et al. [180] provide a dataset consisting of 360-degree images paired with ground-truth depth maps in equirectangular format; however, their datasets do not contain corresponding fish-eye images and only cover indoor scenes with constant lighting. To address the absence of a suitable dataset, we utilize computer graphics and virtual reality technologies to create a realistic and large-scale synthesized dataset, in which the fidelity and geographic information can match the real world well. Our implementation may help in the research on VR system calibration, depth reconstruction, transmission, rendering, and quality assessment of 360-degree content (see Section 3.2.3).
We use the Unity3D development platform to render the RGB images with specific camera properties from virtual scenes and automatically annotate depth labels at the pixel level. Figure 3.10a shows examples of the dataset in different camera format rendered from virtual scenes.
We modified the Unity3D main camera (see Figure 3.10b2) which rendered images in perspective to capture desired contents in fish-eye and equirectangular formats and name it as Scene Recorder (SR).
Given stereo images, we generate the bi-directional optical flow ground-truths based on the depth labels and known camera parameters as post-processing, which are required by our flow estimation network in training and validation phases.
We selected five indoor and outdoor 3D scenes from Unity assets built with realistic details such as lighting and reflection [181 – 186]. Figure 3.10b1 shows one sample of each artificial 3D scenes rendered in equirectangular format. Each artificial scene is imported into Unity3D and gets rendered by using the script and the Shader. Accurate ground-truth depths and images are generated automatically and quite fast (less than 1.5 second per four fish-eye images, stereo panorama, and corresponding depths), and environmental conditions and parameters (such as weather, illumination and objects number, size, position, material) can be controlled completely and flexibly.
We have collected datasets from a total of 92460 time-steps, and at each time step, four fish-eye and two equirectangular images and corresponding depth maps are rendered. Note that the flow information is complete even in occluded regions since the render engine always has full knowledge of all visible and invisible scene objects.

3.2.3 Perceptual Quality Test Paradigm for DASP with 6-DoF

With so many quality assessment algorithms proposed, the question of their relative merits and demerits naturally arises. DASP quality assessment research aims to design quality metrics that can predict perceived VR quality automatically. The objective of the thesis is to answer the following questions: How can we realistically measure DASP quality? How well does the converted optical flow to depth map perform in actual user 3D sensing experiences? How much DASP model solutions support the head motion?
Currently, objective quality in 360-degree video is measured in the planar projection through Structural Similarity Index Measure (SSIM) [187] and standard Peak-Signal-to Noise Ratio (PSNR) [188]. However, they give equal importance to all parts of the spherical image, even though different parts have different viewing probabilities and thus different importance.

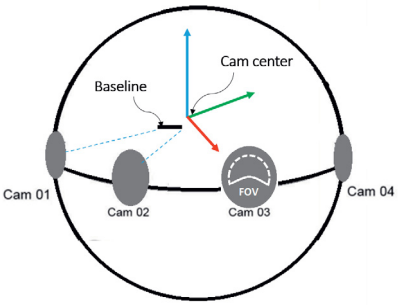


(a) Examples of our proposed dataset. At each time step, four fish-eye and two equirectangular images and corresponding depth maps are rendered .



(b1) Illustration of the diversity in artificial scenes.

(b2) Unity "main camera" renders images in perspective projection. We configured virtual cameras to capture images in fish-eye and equirectangular formats



(b3) SR lenses configuration.

Figure 3.10: Our Unity Synthesized Scene (USS) dataset was built for evaluation and training. a) The artificial scene is imported into Unity3D and gets rendered using the script and the Shader b) The scenes include two indoor and three outdoor scenarios. We modified the "main camera" in Unity to generate our desired dataset. We can flexibly control and move the cameras rig and render our data by synchronized virtual lenses [P5].

Additionally, they still do not give a good representation of subjective quality. Viewport-based PSNR or SSIM metrics could be a solution closer to what the users perceive. However, all objective metrics still fail to consider perceptual artifacts such as visible seams.

While viewing a DASP video on HMD with 6-DoF, immersive image quality strongly relies on the depth map. In this thesis, we propose such a perceptual quality test paradigm that is designed to mitigate some shortcomings of objective and subjective methodologies and, in particular, is suitable for 3D content rendering on the head-mounted displays.

This evaluation also allows researchers to tune the parameters of an algorithm within a particular context, determine the impact of specific components and procedures, and decide.

To this end, we chose geometric and texture features stability as our focus in the evaluation system because they are the fundamental components of 3D scene reconstruction, and their simplification can have a significant impact on human visual perception.

The diagram of the proposed DASP quality assessment model is shown in Figure 3.11a. Assuming the original content was created in one view (reference view), parts of objects visible in the other views (target views), regardless of the occlusion/dis-occlusion artifacts, should preserve their relative geometric shapes based on translated viewpoints. Referring to Figure 3.11a, firstly we adapt artificial environments and Unity SR to render fish-eye and equirectangular synthetic data (RGB+Depth) at reference viewpoint, as well as equirectangular synthetic data (RGB+Depth) at translated head positions as target views ground-truth. Next, according to Section 3.2 equirectangular images and depth maps are generated from fish-eye images at reference view. The resulting content's effectiveness is evaluated quantitatively and qualitatively with synthetic data (at the target view) to validate its functionality.

Many operators in our proposed DASP pipeline (see Section 3.2) have adjustable parameters, and the model performance is heavily dependent on the value of these parameters. How to determine optimal parameters' value is a difficult problem. Therefore, since the main focus is to design a DASP representation system with 6-DoF, and our system's result depends strongly upon the optical flow estimation step, we evaluate our method's performance based on different flow estimation functions.

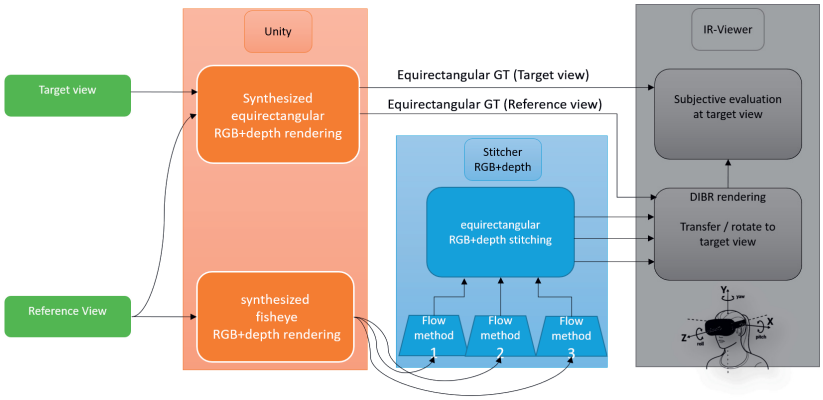
Towards this goal, along with our proposed flow estimator (described in Section 3.2.1), we selected two other state-of the-art optical flow methods FlowNet2 [174], and FlownetS [172] and created DASP contents while other platform parameters are fixed.

The important question is that, as the viewer moves his or her head away from the origin (reference view), which of these three translated contents shows higher similarity according to the reconstructed 3D information compared with rendered ground-truth at target views.

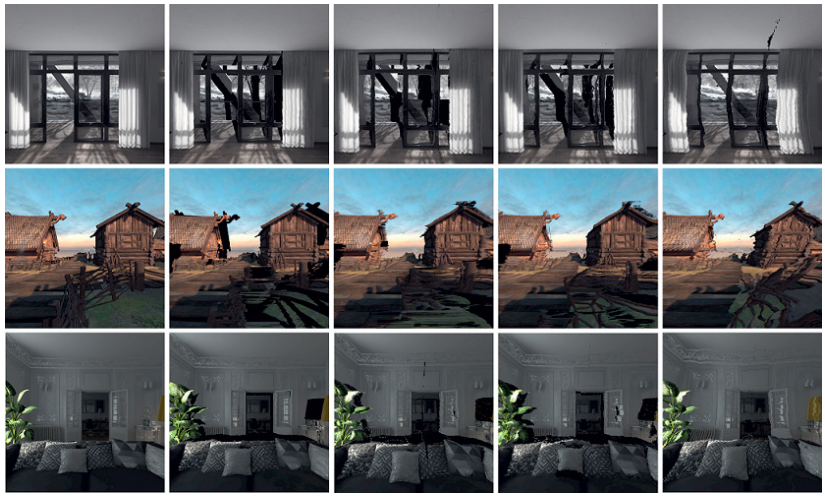
Given a translated eye-position and the viewing direction, the following viewports are generated: (1) Desired target view. (2) Translated synthesized viewport generated at reference view and rendered in the target view. (3) DASP model output viewports, which are generated at the reference view and rendered in the target view. Note that, DASP contents are generated using three different flow models. Examples of generated viewports for these five groups rendered on HMD are shown in Figure 3.11b.

With the accurate alignment of the first two groups, occluded/dis-occluded regions could be easily detected and removed. The information contained in the remaining pixels serves as effective ground-truth samples for objective immersive quality assessment. The objective quality is measured through SSIM. We use five different scenes with different characteristics for the experiments, as shown in Figure 3.10b1.

The basis of the subjective test protocol is inherited from the International Telecommu-



(a) The performance of generated DASP contents by three flow estimation functions (as adjustable parameters) are evaluated.



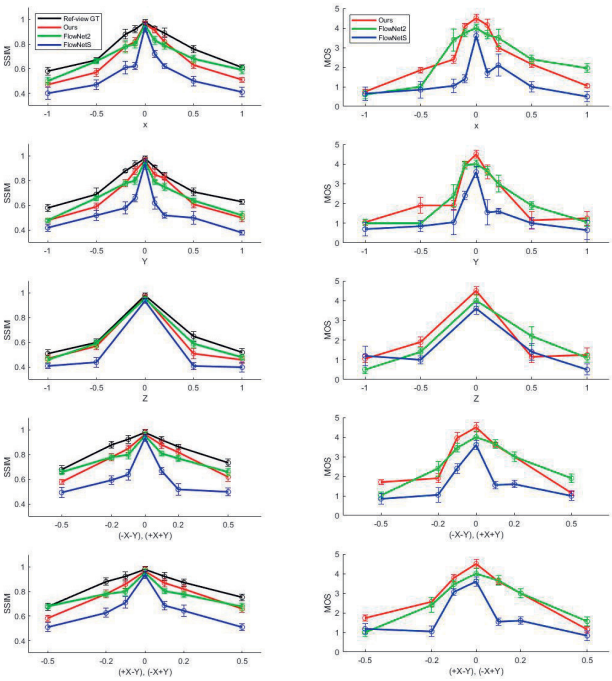
(b) The first from left column shows the synthesized viewports generated from target views; the second column corresponds to translated synthesized viewports generated at reference views and rendered in the target views. Three last columns illustrate viewports generated by our proposed DASP stitching method using three different flow models (Bi-directional flow estimator described in Section 3.2.1), FlowNet2 [174] and FlowNetS [172]) which are shifted from reference views to target views. From the first to third rows, The viewports correspond to direct viewing angle (no head rotation) and 50 cm horizontal shift to left, 100 cm horizontal shift to the right, and 50 cm to the front from the reference, respectively.

Figure 3.11: Perceptual quality test paradigm for DASP with 6-DoF. Our proposed method takes into account the actual user sensing experiences. Stitching and depth estimation methods are evaluated quantitatively and qualitatively with synthetic data to validate their functionality by comparing generated contents rendered in different viewports with those viewports created using known 3D geometry [P5].

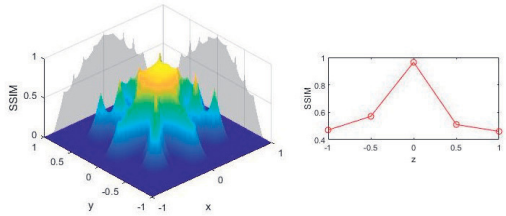
nications Union (ITU) standards for subjective quality assessment of 2D, and 3D video [189 – 191]. However, these specifications are not directly suited for evaluating 6-DoF VR content. Some extensions and adaptation thus are required. The viewports corresponding to the target view vary from 0 to 5 m away from the reference view. The detail of the subjective test protocol and selection of the target viewports position and orientation are presented in detail at [P5].
Using VR headsets instead of flat screens to display VR content is beneficial for immersion. Therefore, our subjective test approach is proposed to be done in a VR environment (such as a Oculus Rift, Samsung Gear VR, or HTC Vive), and the VR glasses should be calibrated with respect to the sensor positions.
The number of contents assessed by each test subject was 660. e.g., 33 target view positions from 5 different scenes to assess 4 different DASP flow-based methods. Commonly, a 10 seconds viewing duration has been used in conventional image quality assessment studies, but we had observed sometimes (especially when scenes changed) that this is insufficient when viewing immersive images. Moreover, It was discovered in the begging of our experiments that several users experienced issues such as discomfort and visual fatigue. The proposed framework uses hand controllers (e.g., Oculus Touch) to address these issues, which enabled the subject to switch and pause contents display manually, allowing the user to stabilize the experience and deliver more reliable opinion scores.
To extract valuable conclusions from the subjective test, it is essential to analyze the evaluations provided by the observers statistically. The observers score five groups of viewports at each step. Thus, the rating scores ranged from 0 (Bad) to 5 (Excellent). This technique is called Mean Opinion Score (MOS) [192] and is computed by averaging the scores provided by the observers.

Results and Discussion

This section presents the overall perceptual quality of our DASP method and its components and compares contents generated using the proposed flow method in Section 3.2.1 against ones created by Flownet2 and FlownetS. Three methods produce DASP close to each other, and at first glance, results appear plausible. Visual comparison between DASPs generated using three methods can be found in Figures 7 and 9 and in [P5]. It is difficult to tell from the 2D Panorama how much the results differ. Figure 3.12 summarizes the three methods' SSIM and MOS scores using the proposed DASP quality assessment paradigm.
The experimental results demonstrate our model's high performance using our proposed flow model around the reference position. The mean SSIM and MOS is high until a certain critical viewing radius and thereafter drops sharply compared to Flownet2. This is expected since Flownet2 provides smoother depth around objects. Although the objective evaluation results are not exactly the same as the subjective evaluation results, they are highly consistent.



(a) The SSIM and MOS scores averaged over all scenes (and all participants for the subjective test) for each optical flow algorithm. The vertical axis gives the score, while the horizontal axis represents the head translation direction.



(b) SSIM averaged over all scenes for our proposed flow method illustrated within 6-DOF motion directions (left image for XY-plane and right image across Z-axis)

Figure 3.12: Qualitative and quantitative evaluation of 6-DOF contents [P5].

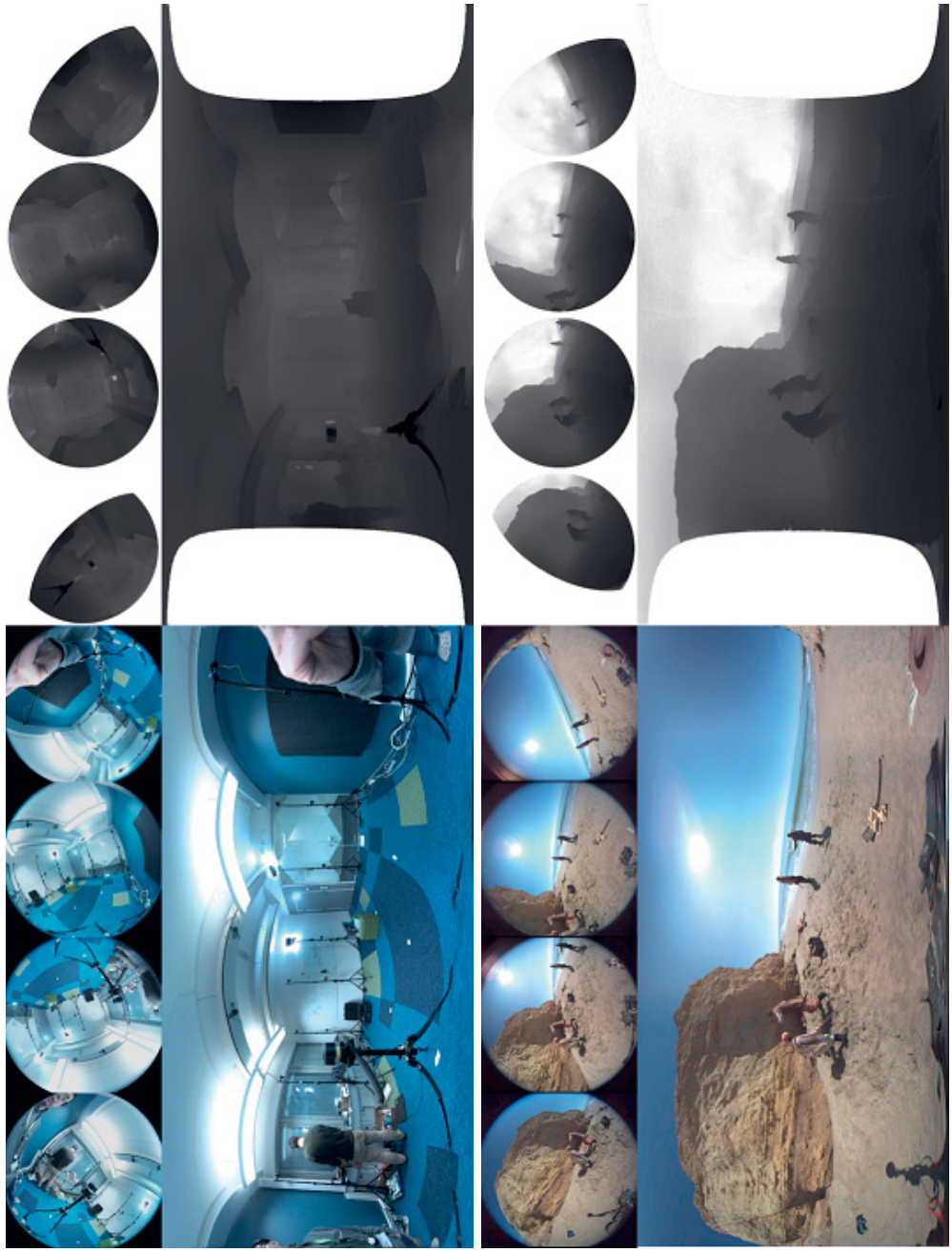


Figure 3.13: Qualitative results from challenging indoor and outdoor scenes. Left-eye equirectangular contents are shown for each scene because the corresponding right-eye is almost indistinguishable from the left-eye when seen in 2D. Our method achieves very accurate and reliable results in fast run-time inherited from the bi-directional CNN flow network. We can get the feeling of immersion when watching them in HMD [P5].

4 Conclusion

Understanding the scene's geometry is a fundamental task for computer vision applications. This provides the necessary information to navigate, avoid obstacles, and interact intelligently with the surroundings. Accurate and efficient scene perception of the environment is critical for different applications, including high definition mapping, autonomous driving, 3D model reconstruction, and smart city. Mobile mapping, especially based on LiDAR and cameras, is an emerging technology for efficiently collecting object surfaces' 3D spatial data. However, a large amount of data, heavy redundancy and irregular distributions in raw scanned data make data processing a complex task. Another important application domain of computer vision is virtual reality, where input images are acquired, processed, analyzed, and understood similar to humans, and meaningful data is extracted to provide a 360-degree visual experience. To create VR content from multi-lens cameras, we need a projection of a 3D scene, which requires finding and matching feature points between frames. Some of the challenging problems that need to be solved include correspondence matching, which can be difficult for reasons such as texture, occlusion, non-Lambertian surfaces, and resolving ambiguous solutions.

Geometric computer vision and its fundamental techniques were reviewed in Chapter 2. Different depth sensors, how they work, and where they are being used in 3D vision systems are discussed. In particular, various types of LiDAR sensors as active depth-sensing technology and their applications are reviewed. This is followed by photogrammetry imaging technology concepts that utilize images' features to recover information about the 3D scene's structure. The stereo-photogrammetric and LiDAR-based scene reconstruction methods were also briefly compared. The literature on geometric computer vision applications, particularly in the semantic labeling of LiDAR point clouds and street view images, are summarized. Then, relevant methods in depth-augmented stereo panorama for 6-DoF VR systems are discussed.

The methods developed throughout the thesis were presented in Chapter 3. Firstly, a complete urban map data processing pipeline, which annotates 3D MLS LiDAR points with semantic labels, was introduced [P2]. The method is made efficient by combining fast rule-based processing for building and street surface segmentation and super-voxel-based feature extraction and classification for the remaining map elements (cars, pedestrians, trees, and traffic signs). Its extension to TLS LiDAR point cloud classification is subsequently described [P4]. Two back ends for semantically labeled 3D point clouds are introduced as reconstruction and visualization of 3D HDM and indirect semantic segmentation of 2D street view images [P3]. In the indirect approach, semantic labels of 3D point clouds classified based on 3D voxel features are mapped to the image plane. Additionally, a direct image segmentation approach was introduced where all LiDAR points are transformed into each image and mapped to the closest super-pixel. Then image segmentation is processed on the 2D image patches by extracting 3D features [P1]. Secondly, a practical, fast, and robust method to create high-resolution depth-augmented

stereo panoramas from a 360-degree camera is formulated [P5]. Using rendering DASP contents in VR glasses, by converting the depth map into a 3D polygon mesh and applying the associated color image as a texture, the user is free to look around the entire scene and experiences the feeling of actually "being there". The proposed algorithm includes a practical end-to-end deep learning model for optical flow estimation. The CNN model architecture more explicitly leverages geometry by getting both optical flow ground-truths. Moreover, a 6-DoF testbed for immersive content quality assessment is developed. The geometric and texture features stability is chosen as the focus in the objective and subjective evaluation system because they are the fundamental components of 3D scene reconstruction, and their simplification can have a significant impact on human visual perception. Finally, due to the unavailability of synthesized datasets for training the bi-directional flow estimator model and assessing VR quality contents, a rich dataset, including fish-eye and equirectangular images with corresponding depth maps, is generated.

The thesis is set to answer two important sets of research questions: the first set includes these questions: what is the best way to infer information from large 3D point clouds? How to recognize objects and semantics in 3D scenes? How to obtain robust features from the 3D point cloud? How well does the geometric information perform in semantic segmentation? Is it possible to use a machine learning model with classical computer vision algorithms to enhance semantic segmentation model performance? How rule-based segmentation is applied, and what is the effect of this stage on the overall pipeline? How to make a 3D map of urban areas using currently available measurement technologies and represent the information on this map? The second set include these questions: can the entire stereo vision problem be formulated with deep learning using our understanding of stereo geometry? Can we build a compact and effective CNN model for optical flow estimation to be used for view compositing? How much the performance of optical flow methods affect Depth Augmented Stereo Panorama's (DASP) quality? Can we improve the performance of a CNN model by training it on the synthesized VR dataset? How well does the converted optical flow to depth map perform in actual user 3D sensing experiences? How much DASP model solutions support the head motion? How can we realistically measure DASP quality?

Based on the results summarized above and included in the original publications, we find that using scene geometry indeed improves the representational power of computer vision models, allowing them to learn relationships in the 3D scenes and improving their performance by simplifying the learning task. Consequently, many complex relationships in the scene, such as objects' shape, size, and depth, do not need to be learned from scratch with machine learning. As a conclusion, the novel contributions in the thesis address several core 3D vision problems and provide efficient solutions. Furthermore, the proposed architectures are practical, real-time systems that impact the development of today's technologies, including autonomous vehicles, robots, smart-city infrastructure, virtual reality, and augmented reality. These findings satisfactorily answer the main research questions and hence meet the objectives set in the thesis.

Future research should further explore the reduction of the computational complexity of the proposed deep learning models using pruning and quantization techniques. Moreover, combining handcrafted features and row-data based deep learning representations should be studied for point cloud classification. The fusion of LiDAR data and VR imagery can be explored for a more robust scene rendering.

Bibliography

[1] F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin, "Applications of 3D city models: State of the art review," *ISPRS International Journal of Geo-Information*, vol. 4, no. 4, pp. 2842–2889, 2015.

[2] J. Pirker, I. Lesjak, J. Kopf, A. Kainz, and A. Dini, "Immersive learning in real VR," in *Real VR – Immersive Digital Reality*. Springer, 2020, pp. 321–336.

[3] A. D. Kumar, R. Karthika, and K. Soman, "Stereo camera and LiDAR sensor fusion-based collision warning system for autonomous vehicles," in *Advances in Computational Intelligence Techniques*. Springer, 2020, pp. 239–252.

[4] L. Fan and P. Babahajiani, "Objection recognition in a 3D scene," Dec. 19 2017, US Patent 9,846,946.

[5] L. Nalpantidis and A. Gasteratos, "Stereo vision for robotic applications in the presence of non-ideal lighting conditions," *Image and Vision Computing*, vol. 28, no. 6, pp. 940–951, 2010.

[6] Y. Song, X. Chen, X. Wang, Y. Zhang, and J. Li, "6-DoF image localization from massive geo-tagged reference images," *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1542–1554, Aug 2016.

[7] B. K. Cabral, "Introducing facebook surround 360: An open, high-quality 3D-360 video capture system," *Retrieved March*, vol. 25, p. 2017, 2016.

[8] S. S. Mate, J. Leppanen, P. Babahajiani, and J. Fu, "Methods and apparatuses for directional view in panoramic content," Jun. 30 2016, US patent App. 14/586,286.

[9] T. T. Nguyen, D. C. Slaughter, N. Max, J. N. Maloof, and N. Sinha, "Structured light-based 3D reconstruction system for plants," *Sensors*, vol. 15, no. 8, pp. 18 587–18 612, 2015.

[10] A. Maimone and H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," in *2012 IEEE Virtual Reality Workshops (VRW)*. IEEE, 2012, pp. 51–54.

[11] W. Wiley and I. H. McLaren, "Time-of-flight mass spectrometer with improved resolution," *Review of scientific instruments*, vol. 26, no. 12, pp. 1150–1157, 1955.

[12] L. Fan and P. Babahajiani, "Automatic scene parsing," May 21 2015, US patent App. 14/534,124.

[13] S. Katz and A. Adler, "Depth camera based on structured light and stereo vision," Mar. 8 2012, US patent App. 12/877,595.

[14] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus time-of-flight kinect," *Computer vision and image understanding*, vol. 139, pp. 1–20, 2015.

[15] M. Lemmens, "A survey on stereo matching techniques," *International Archives of Photogrammetry and Remote Sensing*, vol. 27, no. B8, pp. 11–23, 1988.

[16] A. Lucieer, S. M. d. Jong, and D. Turner, "Mapping landslide displacements using structure from motion (SfM) and image correlation of multi-temporal UAV photography," *Progress in Physical Geography*, vol. 38, no. 1, pp. 97–116, 2014.

[17] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "SfM-net: Learning of structure and motion from video," *arXiv preprint arXiv:1704.07804*, 2017.

[18] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape-from-shading: a survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 8, pp. 690–706, 1999.

[19] Q.-T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *International journal of computer vision*, vol. 17, no. 1, pp. 43–75, 1996.

[20] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[21] R. Wang, J. Bach, J. Macfarlane, and F. P. Ferrie, "A new upsampling method for mobile LiDAR data," in *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*. IEEE, 2012, pp. 17–24.

[22] E. P. Baltsavias, "A comparison between photogrammetry and laser scanning," *ISPRS Journal of photogrammetry and Remote Sensing*, vol. 54, no. 2-3, pp. 83–94, 1999.

[23] G. Bitelli, M. Dubbini, and A. Zanutta, "Terrestrial laser scanning and digital photogrammetry techniques to monitor landslide bodies," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, no. Part B5, pp. 246–251, 2004.

[24] D. Veneziano, S. Hallmark, and R. Souleyrette, "Comparison of LiDAR and conventional mapping methods for highway corridor studies," Tech. Rep., 2002.

[25] J. C. White, P. Tompalski, N. C. Coops, and M. A. Wulder, "Comparison of airborne laser scanning and digital stereo imagery for characterizing forest canopy gaps in coastal temperate rainforests," *Remote sensing of environment*, vol. 208, pp. 1–14, 2018.

[26] X. Tong, X. Liu, P. Chen, S. Liu, K. Luan, L. Li, S. Liu, X. Liu, H. Xie, and Y. Jin, "Integration of UAV-based photogrammetry and terrestrial laser scanning for the three-dimensional mapping and monitoring of open-pit mine areas," *Remote Sensing*, vol. 7, no. 6, pp. 6635–6662, 2015.

[27] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, "Deep learning sensor fusion for autonomous vehicle perception and localization: A review," *Sensors*, vol. 20, no. 15, p. 4220, 2020.

[28] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3D LiDAR and dense stereo," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2181–2188.

[29] K. Park, S. Kim, and K. Sohn, "High-precision depth estimation with the 3D LiDAR and stereo fusion," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2156–2163.

[30] E. Che, J. Jung, and M. J. Olsen, "Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review," *Sensors*, vol. 19, no. 4, p. 810, 2019.

[31] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer, "A survey of urban reconstruction," in *Computer graphics forum*, vol. 32, no. 6. Wiley Online Library, 2013, pp. 146–177.

[32] A. Nguyen and B. Le, "3D point cloud segmentation: A survey," in *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE, 2013, pp. 225–230.

[33] P. Theologou, I. Pratikakis, and T. Theoharis, "A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation," *Computer Vision and Image Understanding*, vol. 135, pp. 49–82, 2015.

[34] J. Li, X. Li, B. Yang, and X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 507–518, 2015.

[35] Y. Zheng, B. Jeon, D. Xu, Q. Wu, and H. Zhang, "Image segmentation by generalized hierarchical fuzzy c-means algorithm," *Journal of Intelligent & Fuzzy Systems*, vol. 28, no. 2, pp. 961–973, 2015.

[36] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[37] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *null*. IEEE, 2003, p. 1470.

[38] Z. Zhou, Y. Wang, Q. M. J. Wu, C. Yang, and X. Sun, "Effective and efficient global context verification for image copy detection," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 48–63, Jan 2017.

[39] 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*, vol. 32, no. 9, pp. 1627–1645, 2010.

[40] C. Zhang, L. Wang, and R. Yang, "Semantic segmentation of urban scenes using dense depth maps," in *European Conference on Computer Vision*. Springer, 2010, pp. 708–721.

[41] P. Sturgess, K. Alahari, L. Ladicky, and P. H. Torr, "Combining appearance and structure from motion features for road scene understanding," 2009.

[42] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *European conference on computer vision*. Springer, 2008, pp. 44–57.

[43] J. Xiao and L. Quan, "Multiple view semantic segmentation for street view images," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 686–693.

[44] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 8, pp. 1362–1376, 2009.

[45] F. Nex and F. Remondino, "UAV for 3D mapping applications: A review. applied geomatics, 6 (1), 1–15," 2014.

[46] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, and J. M. Reynolds, "'structure-from-motion'photogrammetry: A low-cost, effective tool for geoscience applications," *Geomorphology*, vol. 179, pp. 300–314, 2012.

[47] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 807–814.

[48] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[49] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International journal of computer vision*, vol. 80, no. 2, pp. 189–210, 2008.

[50] G. E. Hinton, A. Krizhevsky, and I. Sutskever, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1106–1114, 2012.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[52] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3D object detection in the wild," in *IEEE winter conference on applications of computer vision*. IEEE, 2014, pp. 75–82.

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[54] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *International archives of photogrammetry, remote sensing and spatial information sciences*, vol. 36, no. 5, pp. 248–253, 2006.

[55] X. Y. Jiang, U. Meier, and H. Bunke, "Fast range image segmentation using high-level segmentation primitives," in *Proceedings Third IEEE Workshop on Applications of Computer Vision. WACV'96*. IEEE, 1996, pp. 83–88.

[56] A. D. Sappa and M. Devy, "Fast range image segmentation by an edge detection strategy," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 292–299.

[57] M. A. Wani and H. R. Arabnia, "Parallel edge-region-based segmentation algorithm targeted at reconfigurable multiring network," *The Journal of Supercomputing*, vol. 25, no. 1, pp. 43–62, 2003.

[58] B. Bhanu, S. Lee, C.-C. Ho, and T. Henderson, "Range data processing: Representation of surfaces by edges," in *Proceedings of the eighth international conference on pattern recognition*. IEEE Computer Society Press, 1986, pp. 236–238.

[59] E. Grilli, F. Menna, and F. Remondino, "A review of point clouds segmentation and classification algorithms," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 339, 2017.

[60] E. Castillo, J. Liang, and H. Zhao, "Point cloud segmentation and denoising via constrained nonlinear least squares normal estimates," in *Innovations for shape analysis*. Springer, 2013, pp. 283–299.

[61] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, "Octree-based region growing for point cloud segmentation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 104, pp. 88–100, 2015.

[62] A. Nurunnabi, D. Belton, and G. West, "Robust segmentation in laser scanning 3D point cloud data," in *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)*. IEEE, 2012, pp. 1–8.

[63] Z. Dong, B. Yang, P. Hu, and S. Scherer, "An efficient global energy optimization approach for robust 3D plane segmentation of point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 137, pp. 112–133, 2018.

[64] X. Ning, X. Zhang, Y. Wang, and M. Jaeger, "Segmentation of architecture shape information from 3D point cloud," in *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, 2009, pp. 127–132.

[65] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3D surf for robust three dimensional classification," in *European Conference on Computer Vision*. Springer, 2010, pp. 589–602.

[66] A. Zaharescu, E. Boyer, and R. Horaud, "Keypoints and local descriptors of scalar functions on 2D manifolds," *International Journal of Computer Vision*, vol. 100, no. 1, pp. 78–98, 2012.

[67] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3D object recognition in cluttered scenes with local surface features: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2270–2287, 2014.

[68] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016.

[69] C. Papazov and D. Burschka, "An efficient ransac for 3D object recognition in noisy and occluded scenes," in *Asian Conference on Computer Vision*. Springer, 2010, pp. 135–148.

[70] A. Glent Buch, Y. Yang, N. Kruger, and H. Gordon Petersen, "In search of inliers: 3D correspondence by local and global voting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2067–2074.

[71] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3D local surface description and object recognition," *International journal of computer vision*, vol. 105, no. 1, pp. 63–86, 2013.

[72] A. Velizhev, R. Shapovalov, and K. Schindler, "Implicit shape models for object detection in 3D point clouds," in *International Society of Photogrammetry and Remote Sensing Congress*, vol. 2, 2012.

[73] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[74] P. V. Hough, "Method and means for recognizing complex patterns," Dec. 18 1962, US Patent 3,069,654.

[75] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized hough transform (rht)," *Pattern recognition letters*, vol. 11, no. 5, pp. 331–338, 1990.

[76] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. Torr, "Urban 3D semantic modelling using stereo vision," in *2013 IEEE International Conference on robotics and Automation*. IEEE, 2013, pp. 580–585.

[77] A. Sampath and J. Shan, "Clustering based planar roof extraction from LiDAR data," in *American Society for Photogrammetry and Remote Sensing Annual Conference, Reno, Nevada, May*, 2006, pp. 1–6.

[78] A. Sampath and J. Shan, "Segmentation and reconstruction of polyhedral building roofs from aerial LiDAR point clouds," *IEEE Transactions on geoscience and remote sensing*, vol. 48, no. 3, pp. 1554–1567, 2009.

[79] J. Yan, J. Shan, and W. Jiang, "A global optimization approach to roof segmentation from airborne LiDAR point clouds," *ISPRS journal of photogrammetry and remote sensing*, vol. 94, pp. 183–193, 2014.

[80] M. Shahzad, X. X. Zhu, and R. Bamler, "Facade structure reconstruction using spaceborne tomosar point clouds," in *2012 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2012, pp. 467–470.

[81] M. Shahzad and X. X. Zhu, "Robust reconstruction of building facades for large areas using spaceborne tomosar point clouds," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 2, pp. 752–769, 2014.

[82] J. M. Biosca and J. L. Lerma, "Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 63, no. 1, pp. 84–98, 2008.

[83] M. Carlberg, P. Gao, G. Chen, and A. Zakhor, "Classifying urban landscape in aerial LiDAR using 3D shape analysis," in *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2009, pp. 1701–1704.

[84] J. Zhang, X. Lin, and X. Ning, "SVM-based classification of segmented airborne LiDAR point clouds in urban areas," *Remote Sensing*, vol. 5, no. 8, pp. 3749–3775, 2013.

[85] Z. Li, L. Zhang, X. Tong, B. Du, Y. Wang, L. Zhang, Z. Zhang, H. Liu, J. Mei, and X. Xing, "A three-step approach for tls point cloud classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 9, pp. 5412–5424, 2016.

[86] Z. Wang, L. Zhang, T. Fang, P. T. Mathiopoulos, X. Tong, H. Qu, Z. Xiao, F. Li, and D. Chen, "A multiscale and hierarchical feature extraction method for terrestrial laser scanning point cloud classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 5, pp. 2409–2425, 2014.

[87] S. K. Lodha, D. M. Fitzpatrick, and D. P. Helmbold, "Aerial LiDAR data classification using adaboost," in *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*. IEEE, 2007, pp. 435–442.

[88] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[89] N. Chehata, L. Guo, and C. Mallet, "Airborne LiDAR feature selection for urban classification using random forests," 2009.

[90] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[91] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[92] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, pp. 820–830, 2018.

[93] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "Pointsift: A sift-like network module for 3d point cloud semantic segmentation," *arXiv preprint arXiv:1807.00652*, 2018.

[94] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6411–6420.

[95] H. A. Arief, U. G. Indahl, G.-H. Strand, and H. Tveite, "Addressing overfitting on point cloud classification using atrous xcrf," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 155, pp. 90–101, 2019.

[96] W. Li, F.-D. Wang, and G.-S. Xia, "A geometry-attentional network for als point cloud classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 164, pp. 26–40, 2020.

[97] G. Vosselman, "Building reconstruction using planar faces in very high density height data," *International Archives of Photogrammetry and Remote Sensing*, vol. 32, no. 3; SECT 2W5, pp. 87–94, 1999.

[98] B. Yang, Y. Zang, Z. Dong, and R. Huang, "An automated method to register airborne and terrestrial laser scanning point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 109, pp. 62–76, 2015.

[99] J. Heo, S. Jeong, H.-K. Park, J. Jung, S. Han, S. Hong, and H.-G. Sohn, "Productive high-complexity 3D city modeling with point clouds collected from terrestrial LiDAR," *Computers, Environment and Urban Systems*, vol. 41, pp. 26–38, 2013.

[100] M. Boussaha, E. Fernandez-Moral, B. Vallet, and P. Rives, "On the production of semantic and textured 3D meshes of large scale urban environments from mobile mapping images and LiDAR scans," in *CFPT 2018, Conférence Française de Photogrammétrie et de Télédétection*, 2018.

[101] R. Kurazume, S. Oshima, S. Nagakura, Y. Jeong, and Y. Iwashita, "Automatic large-scale three dimensional modeling using cooperative multiple robots," *Computer Vision and Image Understanding*, vol. 157, pp. 25–42, 2017.

[102] R. Qin and A. Gruen, "3D change detection at street level using mobile laser scanning point clouds and terrestrial images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 90, pp. 23–35, 2014.

[103] C. Thomson and J. Boehm, "Automatic geometry generation from point clouds for bim," *Remote Sensing*, vol. 7, no. 9, pp. 11 753–11 775, 2015.

[104] J. Wang and K. Xu, "Shape detection from raw LiDAR data with subspace modeling," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 9, pp. 2137–2150, 2016.

[105] D. Gledhill, G. Y. Tian, D. Taylor, and D. Clarke, "Panoramic imaging—a review," *Computers & Graphics*, vol. 27, no. 3, pp. 435–445, 2003.

[106] B. J. Concannon, S. Esmail, and M. Roduta Roberts, "Head-mounted display virtual reality in post-secondary education and skill training: A systematic review," in *Frontiers in Education*, vol. 4. Frontiers, 2019, p. 80.

[107] W.-T. Lee, H.-I. Chen, M.-S. Chen, I.-C. Shen, and B.-Y. Chen, "High-resolution 360 video foveated stitching for real-time VR," in *Computer Graphics Forum*, vol. 36, no. 7. Wiley Online Library, 2017, pp. 115–123.

[108] L. E. Gurrieri and E. Dubois, "Acquisition of omnidirectional stereoscopic images and videos of dynamic scenes: a review," *Journal of Electronic Imaging*, vol. 22, no. 3, p. 030902, 2013.

[109] Y. Yagi, "Omnidirectional sensing and its applications," *IEICE Transactions on Information and Systems*, vol. 82, no. 3, pp. 568–579, 1999.

[110] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski, "Low-cost 360 stereo photography and video capture," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 148, 2017.

[111] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross, "Panoramic video from unstructured camera arrays," in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 57–68.

[112] H. Guo, S. Liu, T. He, S. Zhu, B. Zeng, and M. Gabbouj, "Joint video stitching and stabilization from moving cameras," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5491–5503, 2016.

[113] M. Wang, J.-B. Liang, S.-H. Zhang, S.-P. Lu, A. Shamir, and S.-M. Hu, "Hyperlapse from multiple spatially-overlapping videos," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1735–1747, 2018.

[114] H.-C. Huang and Y.-P. Hung, "Panoramic stereo imaging system with automatic disparity warping and seaming," *Graphical Models and Image Processing*, vol. 60, no. 3, pp. 196–208, 1998.

[115] Y. Pritch, M. Ben-Ezra, and S. Peleg, "Optics for omnistereo imaging," in *Foundations of Image Understanding*. Springer, 2001, pp. 447–467.

[116] C. Weissig, O. Schreer, P. Eisert, and P. Kauff, "The ultimate immersive experience: panoramic 3D video acquisition," in *International Conference on Multimedia Modeling*. Springer, 2012, pp. 671–681.

[117] R. Schäfer, P. Kauff, and C. Weissig, "Ultra high resolution video production and display as basis of a format agnostic production system," in *Proceedings of International Broadcast Conference (IBC 2010)*, vol. 1, 2010.

[118] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz, "Jump: virtual reality video," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 198, 2016.

[119] J. Studios, "The cinematic VR field guide," *A Guide to Best Practices for Shooting*, vol. 360, 2017.

[120] R. I. Change, "New RICOH THETA Model, capturing 360-degree images in one shot, is on sale soon—spherical video function, api and sdk (beta version)," *News Release, Oct*, vol. 28, p. 3, 2014.

[121] T. Ho and M. Budagavi, "Dual-fisheye lens stitching for 360-degree imaging," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2172–2176.

[122] H.-Y. Shum and L.-W. He, "Rendering with concentric mosaics," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 299–306.

[123] R. Szeliski, "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2007.

[124] R. Szeliski, H.-Y. Shum, H.-Y. Shum, and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 251–258.

[125] H. Dersch, "Panorama tools," *World Wide Web*, 2005.

[126] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 600–608, 2004.

[127] F. Huang and R. Klette, "Stereo panorama acquisition and automatic image disparity adjustment for stereoscopic visualization," *Multimedia Tools and Applications*, vol. 47, no. 3, pp. 353–377, 2010.

[128] J. Thatte, J.-B. Boin, H. Lakshman, G. Wetzstein, and B. Girod, "Depth augmented stereo panorama for cinematic virtual reality with focus cues," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1569–1573.

[129] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec, "A system for acquiring, processing, and rendering panoramic light field stills for virtual reality," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–15, 2018.

[130] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision*. Springer, 2016, pp. 501–518.

[131] T. Bertel, N. D. Campbell, and C. Richardt, "Megaparallax: Casual 360 panoramas with motion parallax," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 5, pp. 1828–1835, 2019.

[132] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–13, 2015.

[133] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Techniques and Applications of Image Understanding*, vol. 281. International Society for Optics and Photonics, 1981, pp. 319–331.

[134] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981.

[135] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[136] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[137] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[138] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[139] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1164–1172.

[140] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "Demon: Depth and motion network for learning monocular stereo," in *IEEE Conference on computer vision and pattern recognition (CVPR)*, vol. 5, 2017, p. 6.

[141] L. Yu, M. Hannuksela, T. Tillo, C. Lin, and M. Gabbouj, "Is the transmission of depth data always necessary for 3d video streaming?" in *2018 Eighth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE, 2018, pp. 1–5.

[142] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[143] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 2, 2017, p. 6.

[144] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," *CoRR, vol. abs/1703.04309*, 2017.

[145] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934–8943.

[146] A. Ahmadi and I. Patras, "Unsupervised convolutional neural networks for motion estimation," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1629–1633.

[147] J. Y. Jason, A. W. Harley, and K. G. Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," in *European Conference on Computer Vision*. Springer, 2016, pp. 3–10.

[148] C. Ozcinar, A. De Abreu, and A. Smolic, "Viewport-aware adaptive 360 video streaming using tiles for virtual reality," in *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2174–2178.

[149] R. Schatz, A. Sackl, C. Timmerer, and B. Gardlo, "Towards subjective quality of experience assessment for omnidirectional video streaming," in *Proc. 9th Int. Conf. Qual. Multimedia Exp.(QoMEX)*, 2017, pp. 1–6.

[150] E. Upenik, M. Rerabek, and T. Ebrahimi, "On the performance of objective metrics for omnidirectional visual content," in *Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on*. IEEE, 2017, pp. 1–6.

[151] P. Paalanen, J.-K. Kämäräinen, and H. Kälviäinen, "Image based quantitative mosaic evaluation with artificial video," in *Scandinavian Conference on Image Analysis*. Springer, 2009, pp. 470–479.

[152] Y. Qian, D. Liao, and J. Zhou, "Manifold alignment based color transfer for multiview image stitching," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 2013, pp. 1341–1345.

[153] W. Xu and J. Mulligan, "Performance evaluation of color correction approaches for automatic multi-view image and video stitching," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 263–270.

[154] S. Leorin, L. Lucchese, and R. G. Cutler, "Quality assessment of panorama video for video conferencing applications," in *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*. IEEE, 2005, pp. 1–4.

[155] M. Yu, H. Lakshman, and B. Girod, "A framework to evaluate omnidirectional video coding schemes," in *2015 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2015, pp. 31–36.

[156] E. Upenik, M. Rerabek, and T. Ebrahimi, "Testbed for subjective evaluation of omnidirectional visual content," in *32nd Picture Coding Symposium (PCS), Nuremberg, Germany, December 4-7, 2016.*, no. EPFL-CONF-221560. IEEE, 2016.

[157] M. Huang, Q. Shen, Z. Ma, A. C. Bovik, P. Gupta, R. Zhou, and X. Cao, "Modeling the perceptual quality of immersive images rendered on head mounted displays: Resolution and compression," *IEEE Transactions on Image Processing*, vol. 27, no. 12, pp. 6039–6050, 2018.

[158] M. Collins, R. E. Schapire, and Y. Singer, "Logistic regression, adaboost and bregman distances," *Machine Learning*, vol. 48, no. 1-3, pp. 253–285, 2002.

[159] A. Serna, B. Marcotegui, F. Goulette, and J.-E. Deschaud, "Paris-rue-madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods," in *4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014*, 2014.

[160] A. Serna and B. Marcotegui, "Detection, segmentation and classification of 3D urban objects using mathematical morphology and supervised learning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 93, pp. 243–255, 2014.

[161] A. Aijazi, P. Checchin, and L. Trassoudaine, "Segmentation based classification of 3D urban point clouds: A super-voxel based approach with evaluation," *Remote Sensing*, vol. 5, no. 4, pp. 1624–1650, 2013.

[162] K. Lai and D. Fox, "Object recognition in 3D point clouds using web data and domain adaptation," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1019–1037, 2010.

[163] F. Okura, M. Kanbara, and N. Yokoya, "Mixed-reality world exploration using image-based rendering," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 8, no. 2, p. 9, 2015.

[164] W. Piekarski and B. H. Thomas, "Tinmith-metro: New outdoor techniques for creating city models with an augmented reality wearable computer," in *Proceedings Fifth International Symposium on Wearable Computers*. IEEE, 2001, pp. 31–38.

[165] D.-I. L. Ross, "Virtual 3D city models in urban land management," *Technischen Universitat, Berlin*, 2010.

[166] A. Oko, T. Sato, H. Kume, T. Machida, and N. Yokoya, "Evaluation of image processing algorithms on vehicle safety system based on free-viewpoint image rendering," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 706 – 711.

[167] D. Wolf, A. Howard, and G. S. Sukhatme, "Towards geometric 3D mapping of outdoor environments using mobile robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1507 – 1512.

[168] M. Tarini, P. Cignoni, and R. Scopigno, "Visibility based methods and assessment for detail-recovery," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. IEEE Computer Society, 2003, p. 60.

[169] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," *CoRR, vol. abs/1703.04309*, 2017.

[170] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8934 – 8943.

[171] M. Otte and H.-H. Nagel, "Optical flow estimation: advances and comparisons," in *European conference on computer vision*. Springer, 1994, pp. 49 – 60.

[172] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758 – 2766.

[173] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161 – 4170.

[174] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462 – 2470.

[175] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231 – 1237, 2013.

[176] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040 – 4048.

[177] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231 – 1237, 2013.

[178] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040 – 4048.

[179] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision*. Springer, 2012, pp. 611–625.

[180] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras, "Omnidepth: Dense depth estimation for indoors spherical panoramas," *arXiv preprint arXiv:1807.09620*, 2018.

[181] "Archvizpro interior vol.5 scene," https://assetstore.unity.com/packages/3d/environments/archvizpro-interior-vol-5-93317, accessed: 2018-12-1.

[182] "Archvizpro interior vol.3 scene," https://assetstore.unity.com/packages/3d/environments/archvizpro-interior-vol-3-623377, accessed: 2018-12-1.

[183] "Viking village scene," https://assetstore.unity.com/packages/essentials/tutorial-projects/viking-village-29140, accessed: 2018-12-1.

[184] "Urban construction pack scene," https://assetstore.unity.com/packages/3d/environments/urban/urban-construction-pack-8081, accessed: 2018-12-1.

[185] "Megacity construction scene," https://assetstore.unity.com/packages/3d/environments/urban/megacity-construction-kit-19469, accessed: 2018-12-1.

[186] "Corridor lighting scene," https://assetstore.unity.com/packages/essentials/tutorial-projects/corridor-lighting-example-33630, accessed: 2018-12-1.

[187] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[188] A. Tanchenko, "Visual-PSNR measure of image quality," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 874–878, 2014.

[189] "Methodology for the subjective assessment of the quality of television pictures," in *International Telecommunications Union – Standardization (ITU-R) Recommendation BT.500-13,*, 01/2012.

[190] "Methods for the subjective assessment of video quality, audio quality and audiovisual quality of internet video and distribution quality television in any environment," in *International Telecommunications Union – Standardization (ITU-T), Recommendation P.915*, 03/2016.

[191] "Subjective video quality assessment methods for multimedia applications," in *International Telecommunications Union – Standardization (ITU-T), Recommendation P.910*, 04/2008.

[192] J. Li, K. Yu, Y. Zhao, Y. Zhang, and L. Xu, "Cross-reference stitching quality assessment for 360 omnidirectional images," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2360–2368.

PUBLICATION
I

Semantic parsing of street scene images using 3D LiDAR point cloud

P.Babahajiani, L.Fan, and M.Gabbouj

IEEE International Conference on Computer Vision (ICCV) Workshops, 2013, pp. 714-721

Semantic Parsing of Street Scene Images Using 3D LiDAR Point Cloud

Pouria Babahajiani
Tampere University of Technology
Tampere, Finland
pouria.babahajiani@tut.fi

Lixin Fan
Nokia Research Center
Tampere, Finland
fanlixin@ieee.org

Moncef Gabbouj
Tampere University of Technology
Tampere, Finland
moncef.gabbouj@tut.fi

Abstract

In this paper we propose a novel street scene semantic parsing framework, which takes advantage of 3D point clouds captured by a high-definition LiDAR laser scanner. Local 3D geometrical features extracted from subsets of point clouds are classified by trained boosted decision trees and then corresponding image segments are labeled with semantic classes e.g. buildings, road, sky etc. In contrast to existing image-based scene parsing approaches, the proposed 3D LiDAR point cloud based approach is robust to varying imaging conditions such as lighting and urban structures. The proposed method is evaluated both quantitatively and qualitatively on three challenging NAVTEQ True databases and robust scene parsing results are reported.

1. Introduction

Automatic urban scene parsing refers to the process of segmentation and classifying of objects of interest into predefined semantic labels such as "building", "tree" or "road" etc. This task is often done with a fixed number of object categories, each of which requires a training model for classifying image segments (i.e. superpixels). While many techniques for 2D object recognition have been proposed, the accuracy of these systems is to some extent unsatisfactory because 2D image cues are sensitive to varying imaging conditions such as lighting, shadow etc.

In this work, we propose a novel automatic scene parsing approach which takes advantage of 3D geometrical features derived from Light Detection And Ranging (LiDAR) point clouds. Mobile Terrestrial LiDAR (MTL) provides accurate, high-resolution 3D information (e.g. longitude, latitude, altitude) as well as reflectance properties of urban environment (see such an example mobile mapping system in figure 2). Since such 3D information is invariant to lighting and shadow, as a result, significantly more accurate parsing results are achieved by using 3D cues.

1.1. Related Work

Automatic scene parsing is a traditional computer vision problem. Many successful techniques have used single 2D image appearance information such as color, texture and shape [10,12,13]. By using just spatial cues such as surface orientation and vanishing points extracted from single images, Hoiem et al. [7] achieved considerably more robust results. A drawback of single image feature extraction techniques is that they are sensitive to different image capturing conditions such as lighting, camera viewpoint and scene structure. Recently, many efforts have been made to employ 3D scene features derived from single 2D images to achieve more accurate object recognition [2,5,16,17]. Especially, when the input data is a video sequence, 3D cues can be extracted using Structure From Motion (SFM) techniques [18]. Brostow et al. [2], proposed to employ sparse structure from motion point clouds to recover 3D depth information. And in [17] the authors used dense (per-pixel) depth map information recovered in a camera pose independent manner. The SFM technique adopted in these system [2,17], however, is known to be fragile in outdoor environment because of the difficulty in obtaining correct correspondence in cases of sparse texture or occlusion in the images.

With the advancement of LiDAR sensors and Global Positioning Systems (GPS), large-scale, accurate and dense point cloud can be created and used for 3D scene parsing purpose. There has been a considerable amount of research in registering 2D images with 3D point clouds [8,14,15]. Furthermore, there are methods designed for registering point cloud to image using LiDAR *intensity* [1].

1.2. Overview of the Proposed Framework

Figure 1 shows the overview of the proposed scene parsing framework, in which images and LiDAR Point Cloud (PC) are the inputs of the processing pipeline and parsing results are image segments assigned with different class labels. The proposed parsing pipeline starts from aligning 3D LiDAR point cloud with 2D images. Input images are segmented into superpixels to reduce

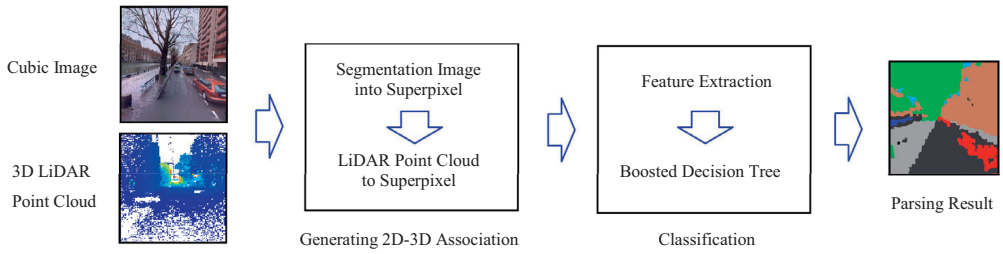


Figure 1: Overview of the proposed framework

computational complexity and to maintain sharp class boundaries. Each SuperPixel (SP) in 2D image is associated with a collection of LiDAR points, which is assumed to form a planar patch in 3D world. 3D features such as *height above camera*, *surface planarity* and *reflectance strength* are extracted for each patch. Using a trained classifier based on boosted decision trees, each 3D feature vector is then assigned with a semantic label such as "sky", "road", "building'' etc. The offline training of the classifier is based on a set of 3D features, which are associated with manually labeled SPs in training images. Main contributions of this work are the following:

1) We demonstrate that using 3D LiDAR point clouds for street view scene parsing improves parsing accuracies under challenging conditions such as varying lighting and urban structures. The improvement is achieved by circumventing error-prone 2D feature extraction and matching steps.

2) We propose a novel method to register 3D point cloud to 2D image plane, and by doing so, occluded points from behind the buildings are removed in an efficient manner (see Section 2 for details).

3) We propose to use a novel LiDAR point reflectance property, i.e. *intensity* feature for semantic scene parsing, and demonstrate that combining both LiDAR intensity feature and geometric features leads to more robust classification results. Consequently, classifiers trained in one type of city and weather condition is now possible to be applied to a different scene structure with high accuracy (See section 4).

2. Generating 2D-3D Association

Given a 3D points cloud and one 2D image with known viewing camera pose, the association module described in this section aims to establish correspondences between collections of 3D points and groups of 2D image pixels. In particular, every collection of 3D points is assumed to be sampled from a visible planar 3D object i.e. patch and corresponding 2D projections are confined within a homogenous region i.e. superpixels (SPs) of the image. While the 3D-2D projection between patches and SPs is straightforward for known geometrical configurations, it still remains a challenging task to deal with outlier 3D points in a computationally efficient manner. We will illustrate in Section 2.2 a novel and simple outlier removal method, but let us first briefly review in Section 2.1 the SP segmentation technique adopted in our processing pipeline.

2.1. Segmenting Images into Superpixels

Without any prior knowledge about how image pixels should be grouped into semantic regions, one commonly used data driven approach segments the input image into homogeneous regions i.e. superpixels based on simple cues such as pixel colors and/or filter responses. The use of SPs improves the computational efficiency and increases the chance to preserve sharp boundaries between different segments.

In our implementation, we adopt the geometric-flow based technique of Levinshtein [9] to segment images into SPs with roughly the same size. Sharp image edges are also well preserved by this method. For input images with dimensionality of 2032×2032 pixels, we set the initial number of SPs as 2500 for each image. See the first image in figure 3 as the example of SP segmentation results

Figure 2: Data Collection vehicle 'NAVTEQ True' [14]

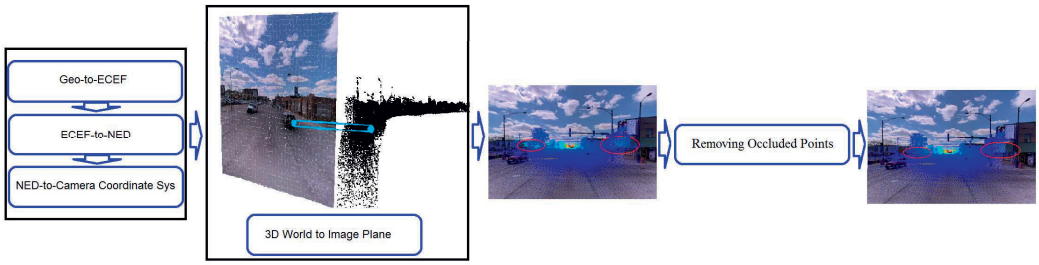


Figure 3: 2D-3D association overview.

.

2.2. LiDAR Point cloud to Superpixel

We first review how to project a 3D point on 2D image plane with known viewing camera pose, and then illustrate a method that associates a collection of 3D points with any given SP on 2D image.

Given a viewing camera pose i.e. position and orientation, represented, respectively, by T a 3×1 translation vector and R a 3×3 rotation matrix, and a 3D point M=[X,Y,Z]^t, expressed in a Euclidean world coordinate system, then the 2D image projection $m_{p=}$ [u, v]^t of the point M is given by

$$\widetilde{m_p} = K\,[R|\,T]\;\widetilde{M} = C\,\widetilde{M} \qquad (1)$$

Where K is an upper triangular 3×3 matrix

$$K=\begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

where f_x and f_y are the focal length in the x and y directions respectively, x_0 and y_0 are the offsets with respect to the image axes, and $\widetilde{m_p}$= [u,v,1]^t and $\widetilde{M}$ = [X,Y,Z,1]^t are the homogeneous coordinates of m_p and M.

3D Light Detection And Ranging (LiDAR) point clouds are often measured in a *geographic coordinate system* (i.e. *longitude, latitude, altitude),* therefore, projecting a 3D LiDAR point on 2D image plane involves two more transformation steps, namely Geo-to-ECEF and ECEF-to-NED. After these two transformations, 3D point in NED coordinate aligns to image plane by equation (2). Figure 3 illustrates an overview of these transformations.

Using the projection step in equation (2) and necessary transformation steps, we are able to identify those 3D points that are projected within a specific SP. Since we assume there is only one dominant 3D patch that associates with the given SP, so outlier 3D points that are far from the patch should be removed. In order to successfully remove outlier points, a plane-fitting method can be used e.g. as in [15]. However, such an outlier removal methods have to be repeatedly applied to every SP and turns out to be too computationally demanding for our application. In this paper, we instead propose a novel and simple method to remove outlier points for all SPs in one pass. The proposed method takes advantage of priori knowledge about urban scene environment and assumes that there are building facades along both sides of the street. While this assumption appears to be oversimplified, the method actually performs quite well with urban scenes in three different US and European cities as demonstrated in the experimental results.

The essence of the method is to fit two hyperbolic curves to 3D points represented in a camera centered two-dimensional Z-u plane (see Figure 4 top image). 3D points that are far from camera center and behind these two hyperbolic curves are deeded outliers and are removed. However, points with depth less than 50 meters (see red line) are kept because they play important roles to label road or other near objects.

The derivation of hyperbolic curves in this Z-u plane is due to the normalization of homogeneous coordinates or simply:

$$v = \frac{f_y\,Y}{Z} + y_0 \qquad u = \frac{f_x\,X}{Z} + x_0$$

In this case the street width X is assumed constant, u is inversely related to the depth Z, and the collection of aligned points in the 3D world lies between two hyperbolic lines (black lines in figure 4)

3. Semantic Parsing of Street Scene Images

After the associating of the LiDAR point cloud with the image SPs, the processing pipeline proceeds to extracting 3D features for different patches. At the offline training phase, these 3D features are used to train a boosted decision tree classifier. Detailed processing steps are elaborated below and the results are presented in section 4.

3.1. Feature Extraction

For each 3D patch, seven features are extracted to train the classifier. Among these seven features, *Height above ground, surface normal, planarity, density* and *intensity* are five camera pose independent features whereas *horizontal distance to camera path* and *depth to camera center* are two features which are defined according to the camera location.

Height above ground: Given a collection of 3D points with known geographic coordinates, the median height of all points is considered as the height feature of the patch. The height information is independent of camera pose and is calculated by measuring the distance between points and the road ground. In contrast to 3D point clouds reconstructed with SFM technique, the advantage of LiDAR point cloud is that we know the exact measure of points height and it is not necessary to use RANSAC method to estimate the ground plane, e.g. such as in [17].

Horizontal distance to camera: Following [17], we compute the horizontal distance of the each patch to the camera as second geographical feature.

Depth to camera: Depth information helps to distinguish objects, because in that case we can estimate the 3D spatial location of each patch. Although these three features represent geographical cues of each patch, they are not completely independent from vehicle location and as seen later when we change scene structure (training in one city and testing in another), the classifier performance is lower than when training and testing are performed on the same scene.

Surface normal: Similar to [17], we also extract surface normal for each patch. But following [11], we adopt a more accurate method to compute the surface normal by fitting a plane to the 3D points in each patch. In our implementation we used RANSAC algorithm to remove outliers which may correspond to very "close" objects such as a pedestrian or vehicle [6].

Planarity: Patch planarity is defined as the average square distance of all 3D points from the best fitted plane computed by RANSAC algorithm. This feature is useful for distinguishing planar objects such as buildings form non planar ones such as trees.

Density: Some objects such as road and sky, have lower density of point cloud as compared to others such as trees and vegetation. Therefore, the number of 3D points in a patch is used as a strong cue to distinguish different classes.

Intensity: LiDAR systems provide not only positioning information but also reflectance property, referred to as intensity, of laser scanned objects. This intensity feature is used in our system, in combination with other features, to classify 3D points. More specifically, the Median intensity of points in each patch is used to train the classifier.

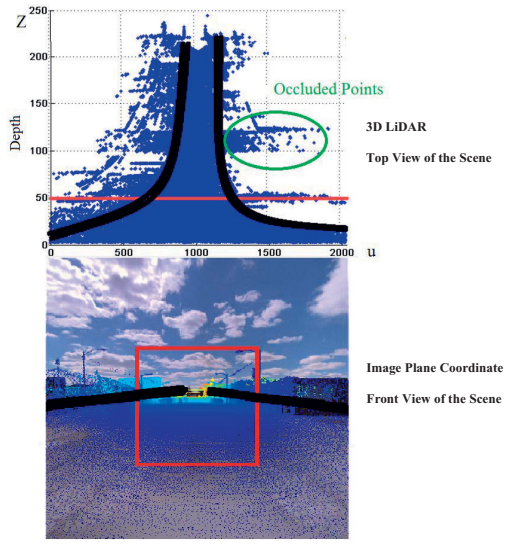


Figure 4: Removing occluded points. The top image shows 3D LiDAR point cloud in NED system. The occluded points in the one bystreet are shown in a green circle. The Bottom image illustrates camera view of scene, occluded points in the bystreet located in the red square (which corresponding to red line in top image) will be deleted.

3.2. Classifier

The Boosted decision tree [3] has demonstrated superior classification accuracy and robustness in many multi-class classification tasks. Acting as weaker learners, decision trees automatically select features that are relevant to the given classification problem. Given different weights of training samples, multiple trees are trained to minimize average classification errors. Subsequently, boosting is done by logistic regression version of Adaboost to achieve higher accuracy with multiple trees combined together.

In our experiments, we boost 20 decision trees each of which has 10 leaf nodes. This parameter setting is similar to those in [7], but with slightly more leaf nodes since we have more classes to label. The number of training samples depends on different experimental settings, which are elaborated in Section 4.

4. Experimental Results

Extensive classification experiments have been performed using point clouds generated with stereo vision techniques [2,17]. Since no labeled dataset consisting of corresponding LiDAR point cloud was available, we

Drive Country (City)	Finland (Helsinki)	United States (Chicago)	France (Paris)
Approx. Lat, Long	60.1°, 24.9°	41.9°, -87.6°	48.8°, 2.4°
Size of Data (GB)	2.8	4.1	3.4
Number of Images	50	110	40
Rate (frame/meter)	1/10	1/15	1/10
Temperature	18.5°c	34°c	5°c
Weather Condition	Sunny	Partly Cloudy	Rainy

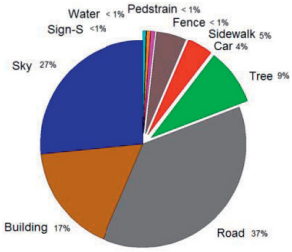


Figure 5: Dataset properties

created and used labeled dataset of driving sequences from 'NAVTEQ True' for all experiments presented in this paper. The dataset includes 200 high quality cubic images and corresponding accurate LiDAR point clouds collected from three different US and European cities. We selected challenging NAVTEQ drives (see figure 5) to evaluate our algorithm in different weather conditions (cloudiness, temperature and daytime) and city landscapes (shape of the buildings vegetation and vehicles). 10 semantic object classes are defined to label the database: *building, tree, sky, car, sign-symbol, pedestrian, road, fence, sidewalk* and *water*.

Note that some of these classes e.g. building, road and sky are common objects in street view images while others such as water, fences etc. occur less frequently. The statistics of occurrences of each class are summarized in Figure 5 as well.

4.1. Data Acquisition

LiDAR data is collected by NAVTEQ True using mobile mapping system consistsing of [14]: a 360 degree LiDAR sensor (Velodyne HDL-64E), a panoramic camera (Ladybug 3), six high-resolution cameras, Inertial Measurement Unit (IMU), GPS, and Distance Measurement Instrument (DMI) (see Figure 2). The LiDAR sensor consists of 64 lasers mounted on upper and lower blocks with 32 lasers in each side and the entire unit spins, and generates over 1.5 million points per second. The panoramic camera covers more than 80 percent of a full sphere with six high quality 1600×1200 Sony CCD sensors, and provides up to 12 MP images at 15 fps. The vehicle is driven at the posted speed limit and the sensors are calibrated and synchronized to produce a coupled collection of high quality geo-referenced (i.e. latitude, longitude and altitude) data. The perspective camera image is generated by rendering the spherical panorama with a view port of 2032×2032 pixels.

4.2. Evaluation with NAVTEQ True Database

We train boosted decision tree classifiers with sample 3D features extracted from the training images. Subsequently, we test the performance of the trained classifier using separated test images. The same tests are applied to three different urban areas. The accuracy of each test is computed by comparing the ground truth with the scene paring results. We report *global* accuracy as the percentage of superpixels correctly classified, *per-class* accuracy (the normalized diagonal of the confusion matrix) and *class average* which represents the average value of per-class accuracies. Since in each experiment, dataset randomly have been divided to two groups of training and testing categories we repeated each experiment five times and the average of resulted experiment represented as the final accuracy.

Direct training and testing: We randomly split each city dataset into two groups in such a way that 70 percent of the images are used for training the classifier and the remaining 30 percent for testing. Table 1 shows the confusion matrixes for different experiments in three cities. As can be seen, some classes in Chicago and Helsinki experiments have not been labeled because there are no sufficient samples for those classes. Our algorithm performs well on most per class accuracies, with the highest accuracy 99% achieved for the sky in Chicago and the lowest 32% for sign-symbol in Paris. The average of the global accuracy for three direct experiments is about 88 %.

Mixed training and testing: The whole 200 images collected from three cities are randomly mixed and then split into 150 for training and 50 for testing. The mixed classification confusion matrix is shown in table 2. It should be noted here that some of the classes have a distinctive geometry and can be classified more easily (e.g., sky and road) whereas others have similar geometrical features (e.g., Fence and building).

Mixing images from different cities poses serious challenges to the parsing pipeline, which is reflected by the decrease in the class average accuracy (down to 59%).

Nevertheless, it seems our system generalizes well to different city scenes and the comparable global accuracy 88% is still maintained.

Cross training and testing: The idea of cross training and testing is to challenge the system with training and testing images taken from different cities. As expected, our method works well when training in Helsinki and

Paris	Sky	Building	Road	Tree	Car	Sidewalk	Sign- S	Pedestrian	Water
Sky	75	4	21	0	0	0	0	0	0
Building	5	90	0	3	0	1	0	0	1
Road	1	0	91	0	2	6	0	0	0
Tree	5	2	0	88	0	5	0	0	0
Car	2	3	55	0	33	7	0	0	0
Sidewalk	2	1	3	1	1	91	0	0	1
Sign- S	5	18	14	10	0	25	32	0	6
Pedestrian	16	24	0	4	0	0	0	47	9
Water	48	5	0	3	0	3	0	0	41

Chicago	Sky	Building	Road	Tree	Car	Sidewalk	Sign- S	Fence
Sky	99	1	0	0	0	0	0	0
Building	12	84	0	2	0	1	0	1
Road	1	0	97	0	0	1	0	0
Tree	10	32	0	57	0	0	0	1
Car	5	10	24	0	46	13	0	2
Sidewalk	3	13	7	0	10	67	0	0
Sign- S	5	14	0	6	0	34	41	0
Fence	7	40	0	1	4	1	0	47

Helsinki	Sky	Building	Road	Tree	Car	Sidewalk
Sky	95	4	0	1	0	0
Building	4	88	0	7	0	1
Road	1	0	96	0	2	1
Tree	1	25	0	74	0	0
Car	10	4	10	0	64	12
Sidewalk	2	15	0	0	26	58

Table 1: Confusion matrices for direct classification in Paris, Chicago and Helsinki

mixed	Sky	Building	Road	Tree	Car	Sidewalk	Sign- S	Fence	Pedestrian	Water
Sky	96	2	0	2	0	0	0	0	0	0
Building	4	90	0	3	0	2	0	1	0	0
Road	2	0	96	0	1	1	0	0	0	0
Tree	6	17	0	74	0	3	0	0	0	0
Car	5	11	35	1	35	11	0	2	0	0
Sidewalk	2	4	12	1	4	77	0	0	0	0
Sign- S	8	2	5	4	3	60	17	0	0	1
Fence	5	37	0	3	4	1	0	49	0	1
Pedestrian	10	34	1	3	3	21	0	0	22	6
Water	48	6	1	5	1	5	0	1	0	33

Experiments\Results	Global Accuracy	Class average Accuracy
Direct (Helsinki)	86 %	79 %
Direct (Chicago)	93 %	67 %
Direct (Paris)	85 %	65 %
Mixed	88 %	59 %
Cross (Helsinki-Chicago)	79 %	52 %
Cross (Chicago-Helsinki)	69 %	42 %
Cross (Helsinki-Paris)	59 %	36 %
Cross (Paris-Helsinki)	64 %	41 %
Cross (Chicago-Paris)	61 %	37 %
Cross (Paris-Chicago)	68 %	45 %

Table 2: Scene parsing statistical results, Right table shows confusion matrix for mixed classification. Right table compares global and class average accuracy in whole different experiments.

testing in Chicago (79 % global and 52 % class average accuracy) and vice versa (69% global, 42% class average). Comparing to other cross experiments, Chicago and Helsinki cross experiments represent best parsing accuracy because as discussed earlier there are more similar classes compared to Paris which contains major water in its scene.

Applying SP based segmentation to relatively small classes such as pedestrian and sign-symbol often leads to insufficient number of training samples, and hence, low classification accuracies. The plot in Figure 6 illustrates the qualitative comparison between per class accuracy according to their distribution in our datasets. It should be noted that sky, building, road and tree were well recognized in the street scene (all are over 70%). On the other hand, cars and pedestrian have less than 10% accuracies because these classes occur very rarely in the test images. One possible remedy is to obtain the bounding boxes of these objects with a more suitable technique, e.g. part-based object detector [4].

Our system takes advantage of geographical and intensity statistics information of LiDAR point clouds, which is not available for existing methods e.g. in [2,17].

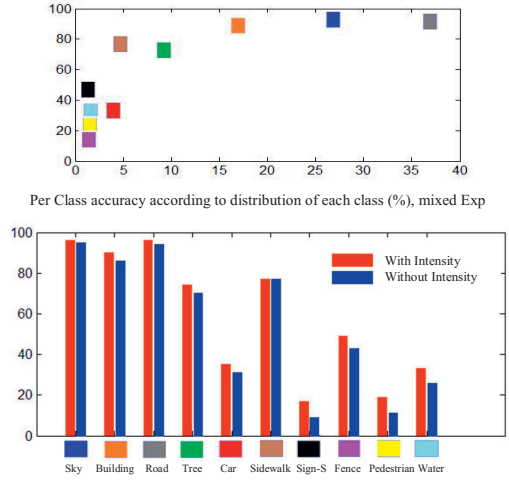


Figure 6: Top plot compares the accuracy of mixed classification based on distribution of existing data. Bottom bar graph shows the impact of intensity feature in mixed training-testing experiment.

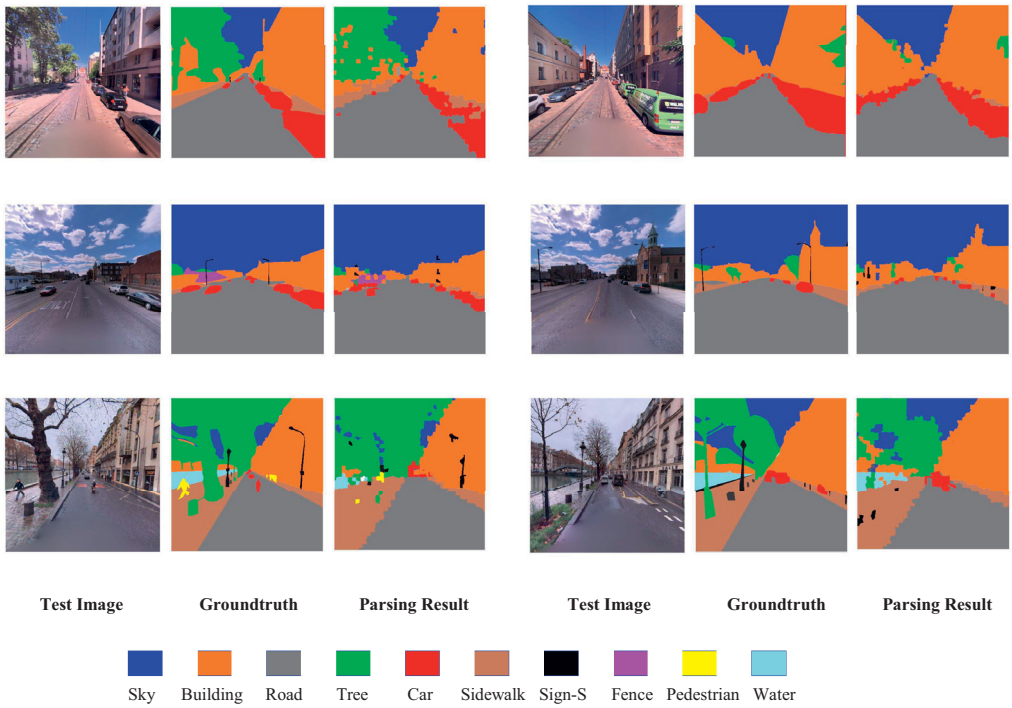


Figure 7: Scene parsing qualitative results. (Left to right): test image, ground truth, parsing result in direct classification, in Helsinki, Chicago and Paris respectively from top to bottom

The bar chart in figure 6 shows that using intensity feature improves classification accuracies, to various extents, for objects e.g. building, car, and signs-symbol and pedestrian. There also seems a discernible increase in its effectiveness as objects become closer to the laser scanner.

5. Conclusions

We have proposed a novel framework for semantic parsing of street view images based on 3D features derived from terrestrial LiDAR point cloud. During the offline training phase, geometrical features associated with 3D patches are extracted and are used to train boosted decision trees classifier. For new input images, the same geometrical features are extracted and semantics labels are assigned to corresponding image regions. One noticeable advantage of our method is the robustness to different lighting condition, shadows and city landscape. Furthermore, by using intensity information from LiDAR data the robustness of classifier is increased for certain object classes.

Future work will focus on the combination of neighboring patches features, to improve the robustness and accuracy of the classification algorithms.

References

[1] S. Becker, and N. Haala, "Combined Feature Extraction for Facade Reconstruction," ISPRSWorkshop on Laser Scanning, 2007.

[2] G.J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and Recognition Using Structure from Motion Point Clouds," ECCV, 2008.

[3] M. Collins, R. Schapire, and Y. Singer, "Logistic regression, adaboost and bregman distances," Machine Learning,, vol. 48, no. 1-3, 2002.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 32, No. 9. (September 2010), pp. 1627-1645

[5] G. Floros, and B. Leibe, "Joint 2D-3D Temporally Consistent Semantic Segmentation of Street Scenes," CVPR, 2012.
[6] M. A. Fischler, R. C. Bolles, "Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," Communications of the ACM, vol. 24, no. 6, pp. 381-395 .
[7] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering Surface Layout from an Image," IJCV, vol. 75, no. 1, 2007.
[8] L. Liu, and I. Stamos, "Automatic 3d to 2d Registration for the Photorealistic Rendering of Urban Scenes," In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2, p. 137–143, 2005.
[9] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, "TurboPixels: Fast Superpixels Using Geometric Flows," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 31, no. 12, p. 2290–2297, 2009.
[10] C. Liu, J. Yuen, and A. Torralba, "Nonparametric Scene Parsing via Label Transfer," IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 12, pp. 2368-2382, 2011.
[11] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments,"Articial Intelligence (KI - Kuenstliche Intelligenz), 2010.
[12] E. Saber, A. M. Tekalp, R. Eschbach, and K. Knox, "Automatic Image Annotation Using Adaptive Color Classification", CVGIP: Graphical Models and Image Processing, vol. 58, pp. 115-126, 1996
[13] M. Turtinen, and M. Pietikainen, "Contextual analysis of textured scene images," 17th British Machine Vision Conference, BMVC, 2006.
[14] R. Wang, F.P. Ferrie, and J. Macfarlane, "Automatic Registration of Mobile LiDAR and Spherical Panoramas," CVPR Workshops, pp. 33-40, 2012.
[15] R. Wang, J. Bach, J. Macfarlane, and F.P. Ferrie , "A New Upsampling Method for Mobile LiDAR Data," WACV, pp. 17-24, 2012.
[16] J. Xiao, and L. Quan, "Multiple View Semantic Segmentation for Street View Images," in Proc. of Intl. Conf. on Computer Vision, 2009.
[17] C. Zhang, L. Wang, R. Yang, "Semantic Segmentation of Urban Scenes Using Dense Depth Maps," in European Conference on Computer Vision, 2010.
[18] G. Zhang, J. Jia, T.T. Wong, and H. Bao , "Consistent Depth Maps Recovery from a Video Sequence," IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 31, no. 6, pp. 974-988, 2009.

PUBLICATION II

Object recognition in 3D point cloud of urban street scene

P.Babahajiani, L.Fan, and M.Gabbouj

Asian Conference on Computer Vision, Springer, 2914, pp 177-190

Object Recognition in 3D Point Cloud of Urban Street Scene

Pouria Babahajiani[1], Lixin Fan[2], Moncef Gabbouj[3]

[1,2] Nokia Research Center, Tampere, Finland
[3] Tampere University of Technology, Tampere, Finland
[1] ext-pouria.babahajiani@nokia.com
[2] fanlixin@ieee.org
[3] moncef.gabbouj@tut.fi

Abstract. In this paper we present a novel street scene semantic recognition framework, which takes advantage of 3D point clouds captured by a high-definition LiDAR laser scanner. An important problem in object recognition is the need for sufficient labeled training data to learn robust classifiers. In this paper we show how to significantly reduce the need for manually labeled training data by reduction of scene complexity using non-supervised ground and building segmentation. Our system first automatically segments grounds point cloud, this is because the ground connects almost all other objects and we will use a connect component based algorithm to oversegment the point clouds. Then, using binary range image processing building facades will be detected. Remained point cloud will grouped into voxels which are then transformed to super voxels. Local 3D features extracted from super voxels are classified by trained boosted decision trees and labeled with semantic classes e.g. tree, pedestrian, car, etc. The proposed method is evaluated both quantitatively and qualitatively on a challenging fixed-position *Terrestrial Laser Scanning* (TLS) Velodyne data set and two *Mobile Laser Scanning* (MLS), Paris-rue-Madam and NAVTEQ True databases. Robust scene parsing results are reported.

1 Introduction

Automatic urban scene objects recognition refers to the process of segmentation and classifying of objects of interest into predefined semantic labels such as building, tree or car etc. This task is often done with a fixed number of object categories, each of which requires a training model for classifying scene components. While many techniques for 2D object recognition have been proposed, the accuracy of these systems is to some extent unsatisfactory because 2D image cues are sensitive to varying imaging conditions such as lighting, shadow etc. In this work, we propose a novel automatic scene parsing approach which takes advantage of 3D geometrical features extracted from Light Detection And Ranging (LiDAR) point clouds. Since such 3D information is invariant to lighting and shadow, as a result, significantly more accurate parsing results are achieved.

While a laser scanning or LiDAR system provides a readily available solution for capturing spatial data in a fast, efficient and highly accurate way, the enormous volume of captured data often come with no semantic meanings. We, therefore, develop techniques that significantly reduce the need for manual labelling of training data and apply the technique to the all data sets. Laser scanning can be divided into three categories, namely, *Airborne Laser Scanning* (ALS), *Terrestrial Laser Scanning* (TLS) and *Mobile Laser Scanning* (MLS). The proposed method is evaluated both quantitatively and qualitatively on a challenging TLS Velodyne data set and two MLS, Paris-rue-Madam and NAVTEQ True databases.

1.1 literature review

Automatic scene parsing is a traditional computer vision problem. Many successful techniques have used single 2D image appearance information such as color, texture and shape [1, 2]. By using just spatial cues such as surface orientation and vanishing points extracted from single images considerably more robust results are achieved [3]. In order to alleviate sensitiveness to different image capturing conditions, , many efforts have been made to employ 3D scene features derived from single 2D images and thus achieving more accurate object recognition [4]. For instance, when the input data is a video sequence, 3D cues can be extracted using Structure From Motion (SFM) techniques [5]. With the advancement of LiDAR sensors and Global Positioning Systems (GPS), large-scale, accurate and dense point cloud are created and used for 3D scene parsing purpose. In the past, research related to 3D urban scene analysis had been often performed using 3D point cloud collected by airborne LiDAR for extracting vegetation and building structures [6]. Hernndez and Marcotegui use range images from 3D point clouds in order to extract k-flat zones on the ground and use them as markers for a constrained watershed [7].Recently, classification of urban street objects using data obtained from mobile terrestrial systems has gained much interest because of the increasing demand of realistic 3D models for different objects common in urban era. A crucial processing step is the conversion of the laser scanner point cloud to a voxel data structure, which dramatically reduces the amount of data to process. Yu Zhou and Yao Yu (2012) present a voxel-based approach for object classification from TLS data [8]. Classification using local features and descriptors such as Spin Image [9], Spherical Harmonic Descriptors [10], Heat Kernel Signatures [11], Shape Distributions [12], and 3D SURF feature [13] have also demonstrated successful results to various extent.

1.2 Overview of the Proposed Framework

In this work, the ground is first segmented and building facades are subsequently detected based on range image morphological operations. We use voxel segmentation that relies on local features and descriptors, to successfully classify different segmented objects in the urban scene.

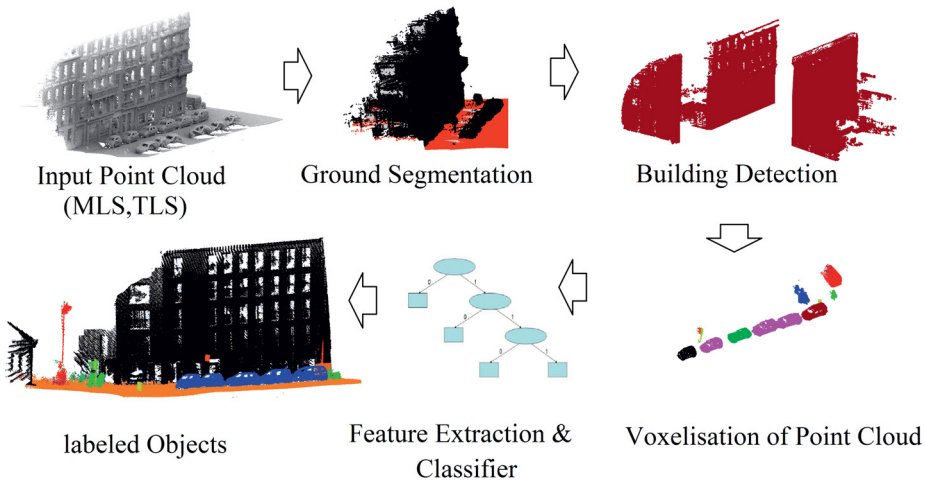


Fig. 1. Overview of the proposed framework

Figure 1 shows the overview of the proposed street scene object recognition pipeline, in which LiDAR Point Cloud (PC) is the input of the processing pipeline and result is PC segments assigned with different class labels. At the outset, the proposed parsing pipeline finds ground points by fitting a ground plane to the given 3D point cloud of urban street scene. Then, non-ground point cloud are projected to range images because they are convenient structure for visualization. Remaining data are processed subsequently to segment building facades. When this process is completed, range images are projected to the 3D point cloud in order to make segmentation on other remained vertical objects. We use a connect component based algorithm to voxilisation of data. The voxel based classification method consists of three steps, namely, a) voxilisation of point cloud, b) merging of voxels into super-voxels and c) the supervised scene classification based on discriminative features extracted from super-voxels.

Using a trained boosted decision tree classifier, each 3D feature vector is then designated with a semantic label such as tree, car, pedestrian etc. The offline training of the classifier is based on a set of 3D features, which are associated with manually labeled super-voxels in training point cloud. Main contributions of this work are the following:

- Develop a novel street object recognition method which is robust to different types of LiDAR point clouds acquisition methods.
- Proposed two-stage (supervised and non-supervised) classification pipeline which requires only small amount of time for training.
- Propose to use novel geometric features leads to more robust classification results (see section 3).

2 Methodology

It is a challenging task to directly extract objects from mobile LiDAR point cloud because of the noise in the data, huge data volume and movement of objects. We therefore take a hybrid two-stage approach to address the above mentioned challenges. Firstly, we adopt an unsupervised segmentation method to detect and remove dominant ground and buildings from other LiDAR data points, where these two dominant classes often correspond to the majority of point clouds. Secondly, after removing these two classes, we use a pre-trained boosted decision tree classifier to label local feature descriptors extracted from remaining vertical objects in the scene. This work shows that the combination of unsupervised segmentation and supervised classifiers provides a good trade-off between efficiency and accuracy. In this section we elaborate our point cloud classification approach.

2.1 Ground Segmentation

The aim of the first step is to remove points belonging to the scene ground including road and sidewalks, and as a result, the original point cloud are divided into ground and vertical object point clouds(Figure 2). The scene point cloud is first divided into sets of 10m×10m regular, non-overlapping tiles along the horizontal x–y plane. Then the following ground plane fitting method is repeatedly applied to each tile. We assume that ground points are of relatively small z values as compared to points belonging to other objects such as buildings or trees (see Fig. 2). The ground is not necessarily horizontal, yet we assume that there is a constant slope of the ground within each tile. Therefore, we first find the *minimal-z-value* (MZV) points within a multitude of 25cm×25cm grid cells at different locations. For each cell, neighboring points that are within a z-distance threshold from the MZV point are retained as *candidate ground points*. Subsequently, a RANSAC method is adopted to fit a plane to candidate ground points that are collected from all cells. Finally, 3D points that are within certain distance (d2 in Fig. 2) from the fitted plane are considered as ground points of each tile. The constant slope assumption made in this approach is valid for our data sets as demonstrated by experimental results in Section 3. The approach is fully automatic and the change of two thresholds parameters do not lead to dramatic change in the results. On the other hand, the setting of grid cell size as 25cm×25cm maintains a good balance between accuracy and computational complexity.

2.2 Building Segmentation

After segmenting out the ground points from the scene, we present an approach for automatic building surface detection. High volume of 3D data impose serious challenge to the extraction of building facades. Our method automatically extract building point cloud (e.g. doors, walls, faades, noisy scanned inner environment of building) based on two assumptions: a) building facades are the

Fig. 2. Ground Segmentation. Left image: Segmented ground and remained vertical objects point cloud are illustrated by red and black color respectively. Right figure: sketch map of fitting plane to one tile.

highest vertical structures in the street; and b) other non-building objects are located on the ground between two sides of street. As can be seen in figure 3, our method projects 3D point clouds to range images because they are convenient structures to process data. Range images are generated by projecting 3D points to horizontal x–y plane. In this way, several points are projected on the same range image pixel. We count the number of points that falls into each pixel and assign this number as a pixel *intensity* value. In addition, we select and store the maximal height among all projected points on the same pixel as *height* value. We define range images by making threshold and binarization of I, where I pixel value is defined as Equation (1)

$$I_i = \frac{P_{intensity}}{Max\_P_{intensity}} + \frac{P_{height}}{Max\_P_{height}} \tag{1}$$

Where I_i is grayscale range image pixel value, $P_{intensity}$ and P_{height} are intensity and height pixel value and $Max\_P_{intensity}$ and Max_P_{height} represent the maximum intensity and height value over the grayscale image.

In the next step we use morphological operation (e.g. close and erode) to merge neighboring point and filling holes in the binary range images (see middle image in Fig. 3). Then we extract contours to find boundaries of objects. In order to trace contours, Pavlidis contour-tracing algorithm [14] is proposed to identify each contour as a sequence of edge points. The resulting segments are checked on aspects such as size and diameters (height and width) to distinguish building from other objects. More specifically, equation (2) defines the geodesic elongation E(X), introduced by Lantuejoul and Maisonneuve (1984), of an object X, where S(X) is the area and L(X) is the geodesic diameter.

$$E(\pi) = \frac{\pi L^2(X)}{4S(X)} \tag{2}$$

The compactness of the polygon shape based on equation (2) can be applied to distinguish buildings from other objects such as trees. Considering the sizes

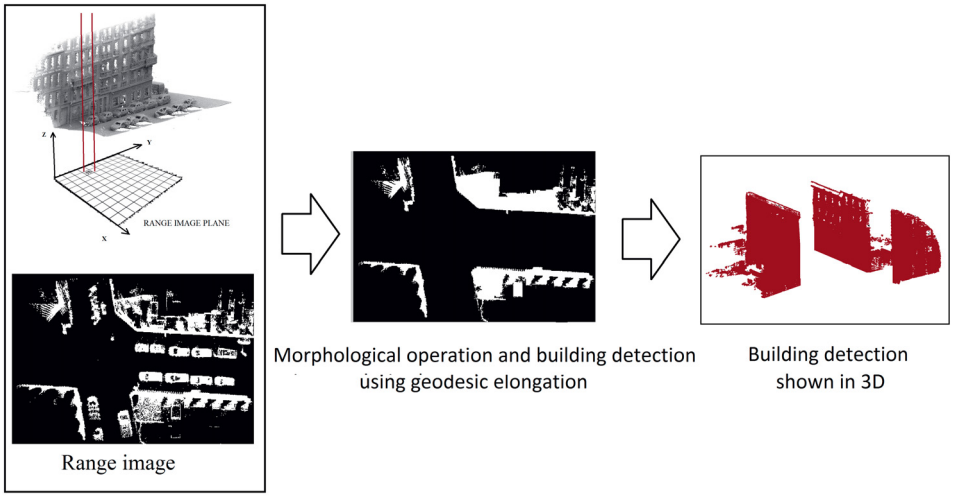


Fig. 3. Building Segmentation

and shape of buildings, the extracted boundary will be eliminated if its size is less than a threshold. The proposed method takes advantage of priori knowledge about urban scene environment and assumes that there are not any important objects laid on the building facades. While this assumption appears to be oversimplified, the method actually performs quite well with urban scenes as demonstrated in the experimental results (see section 3).

The resolution of range image is the only projection parameter during this point cloud alignment that should be chosen carefully. If each pixel in the range image cover large area in 3D space too many points would be projected as one pixel and fine details would not be preserved. On the other hand, selecting large pixel size compared to real world resolution leads to connectivity problems which would no longer justify the use of range images. In our experiment, a pixel corresponds to a square of size .05 m^2.

2.3 Voxel based segmentation

After quick segmenting out the ground and building points from the scene, we use an inner street view based algorithm to cluster point clouds. Although top view range image analysis generates a very fast segmentation result, there are a number of limitation to utilize it for the small vertical object such as pedestrian and cars. These limitations are overcome by using inner view (lateral) or ground based system in which, unlike top view the 3D data processing is done more precisely and the point view processing is closer to objects which provides a more detailed sampling of the objects.

However, this leads to both advantages and disadvantages when processing the data. The disadvantage of this method s includes the demand for more processing power required to handle the increased volume of 3D data. The 3D

point clouds by themselves contain a limited amount of positional information and they do not illustrate color and texture properties of object. According to voxel based segmentation, points which are merely a consequence of a discrete sampling of 3D objects are merged into clusters voxels to represent enough discriminative features to label objects. 3D features such as intensity, area and normal angle are extracted based on these clustersvoxels. The voxel based classification method consists of three steps, voxilisation of point cloud, merging of voxels into super-voxels and the supervised classification based on discriminative features extracted from super-voxels.

2.3.1 Voxelisation of Point Cloude

In the voxelisation step, an unorganized point cloud p is partitioned into small parts, called voxel v. The middle image in figure 4 illustrates an example of voxelisation results, in which small vertical objects point cloud such as cars are broken into smaller partition. Different voxels are labelled with different colors. The aim of using voxelisation is to reduce computation complexity by and to form a higher level representation of point cloud scene. Following [8], a number of points is grouped together to form a variable size voxels. The criteria of including a new point p_{in} into an existing voxel i is essentially determined by the crucial minimal distance threshold d_{th} which is defined as Equation (3):

$$\min(\|p_{im} - p_{in}\|_2) \leq d_{th}, 0 \leq m, n \leq N, m \neq n \tag{3}$$

where p_{im} is an existing 3D point in voxel, p_{in} is a candidate point to merge to the voxel, i is the cluster index, d_{th} is the maximum distance between two point, and N is the maximum point number of a cluster. If the condition is met, the new point is added and the process repeats until no more point that satisfies the condition is found (see Algorithm 1). Equation (3) ensures that the distance between one point and its nearest neighbors belonging to the same cluster is less than d_{th}. Although the maximum voxel size is predefined, the actual voxel sizes depend on the maximum number of points in the voxel (N) and minimum distance between the neighboring points.

repeat
  Select a 3D point for Voxelisation;
  Find all neighboring points to be included in the voxel, with this condition that:
  a point p_{in} directly merge to voxel if its distance to any point p_{in} the voxel will not be farther away than a given distance (d_{th});
until *all 3D points are used in a voxel or the size of cluster is less than (N)*;

Algorithm 1: Voxelisation

2.3.2 Super Voxelisation

For transformation of a voxel to super voxel we propose an algorithm to merge voxels via region growing with respect to the following properties of clusters:

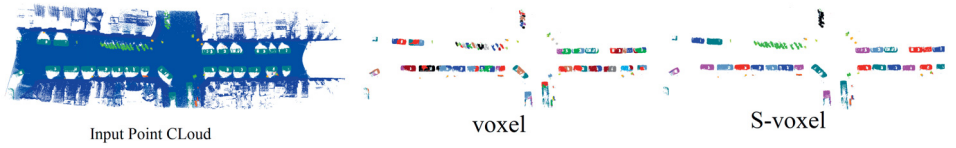


Fig. 4. Voxelisation of Point Cloud. from left to right: top view row point cloud, voxelisation result of objects point cloud after removing ground and building, s-voxelisation approach of point cloud

- **If the minimal geometrical distance, D_{ij}, between two voxels is smaller than a given threshold**, where D_{ij} is defined as Equation (4):

$$D_{ij} = \min(\|p_{ik} - p_{jl}\|_2), k \in (1, m), l \in (1, n) \tag{4}$$

  Where voxels v_i and v_j have m and n points respectively, and p_{ik} and p_{jl} are the 3D point belong to voxel v_i and v_j.
- **If the angle between Normal vectors of two voxels is smaller than a threshold**: In this work, normal vector is calculated using PCA (Principal Component Analysis)[15]. The angle between two s-voxels is defined as angle between their normal vectors (equation 5):

$$\Theta_{ij} = \arccos(< n_i, n_j >) \tag{5}$$

  Where n_i and n_j are normal vectors at v_i and v_j respectively.

The proposed grouping algorithm merges the voxels by considering the geometrical distance ($M < d_{th}$) and normal features of clusters ($\Theta_{ij} < \Theta_{th1}$). All these Voxelisation steps then would be used in grouping these super-voxels (from now onwards referred to as s-voxels) into labeled objects. The advantage of this approach is that we can now use the reduced number of super voxels instead of using thousands of points in the data set, to obtain similar results for classification. The right image in figure 4 illustrates an example of s-voxelisation results, in which different s-voxels are labelled with different colors.

2.3.3 Feature Extraction

For each s-voxel, seven main features are extracted to train the classifier. The seven features are *geometrical shape, height above ground, horizontal distance to center line of street, density, intensity, normal angle* and *planarity*. In order to classify these s-voxels, we assume that the ground points have been segmented well. The object types are so distinctly different however these features as mentioned are sufficient to make a classification. Along with the above mentioned features, geometrical shape descriptors plays an important role in classifying objects. These shape-related features are computed based on the projected bounding box to x - y plane (ground).

Geometrical shape: Projected bounding box has effective features due to the invariant dimension of objects. We extract four feature based on the projected bonding box to represent the geometry shape of objects.

- **Area** : the area of the bounding box is used for distinguishing large-scale objects and small ones.
- **Edge ratio** : the ratio of the long edge and short edge.
- **Maximum edge** : the maximum edge of bounding box.
- **Covariance** : is used to find relationships between point spreading along two largest edges.

• **Height above ground** : Given a collection of 3D points with known geographic coordinates, the median height of all points is considered as the height feature of the s-voxel. The height information is independent of camera pose and is calculated by measuring the distance between points and the road ground.
• **Horizontal distance to center line of street** : Following [16], we compute the horizontal distance of the each s-voxel to the center line of street as second geographical feature. The street line is estimated by fitting a quadratic curve to the segmented ground.
• **Density** : Some objects with porous structure such as fence and car with windows, have lower density of point cloud as compared to others such as trees and vegetation. Therefore, the number of 3D points in a s-voxel is used as a strong cue to distinguish different classes.
• **Intensity** : following [17], LiDAR systems provide not only positioning information but also reflectance property, referred to as intensity, of laser scanned objects. This intensity feature is used in our system, in combination with other features, to classify 3D points. More specifically, the median intensity of points in each s-voxel is used to train the classifier.
• **Normal angle** : Following [18], we adopt a more accurate method to compute the surface normal by fitting a plane to the 3D points in each s-voxel.
• **Planarity** : Patch planarity is defined as the average square distance of all 3D points from the best fitted plane computed by RANSAC algorithm. This feature is useful for distinguishing planar objects with smooth surface like cars form non planar ones such as trees.

2.3.4 Classifier

The Boosted decision tree [19] has demonstrated superior classification accuracy and robustness in many multi-class classification tasks. Acting as weaker learners, decision trees automatically select features that are relevant to the given classification problem. Given different weights of training samples, multiple trees are trained to minimize average classification errors. Subsequently, boosting is done by logistic regression version of Adaboost to achieve higher accuracy with multiple trees combined together. In our experiments, we boost 20 decision trees each of which has 6 leaf nodes. This parameter setting is similar to those in [3], but with slightly more leaf nodes since we have more classes to label. The number of training samples depends on different experimental settings, which are elaborated in Section 3.

3 Experimental Result

The LiDAR technology has been used in the remote sensing urban scene understanding by two main technology: Terrestrial Laser Scanning (TLS), useful for large scale buildings survey, roads and vegetation, more detailed but slow in urban surveys in outdoor environments; Mobile Laser Scanning (MLS), less precise than TLS but much more productive since the sensors are mounted on a vehicle; In order to test our algorithm both type of data sets were used:

1. 3D Velodyne LiDAR as TLS data set [20]
2. Paris-rue-Madame [21] and NAVTAQ True as MLS datasets [17]

We train boosted decision tree classifiers with sample 3D features extracted from training s-voxels. Subsequently we test the performance of the trained classifier using separated test samples. The accuracy of each test is evaluated by comparing the ground truth with the scene parsing results. We report *global accuracy* as the percentage of s-voxel correctly classified, *per-class accuracy* as the normalized diagonal of the confusion matrix and class average which represents the average value of per class accuracies.

3.1 Evaluation Using the Velodyne LiDAR Database

The database includes ten high accurate 3D point cloud scenes collected by a Velodyne LiDAR mounted on a vehicle navigating through the Boston area. Each scene is a single rotation of the LIDAR, yielding a point cloud of nearly 70,000 points. Scenes may contain objects including cars, bicycles, buildings, pedestrians and street signs. Finding ground and building points is discussed in Section 2.1 and 2.2, and the recognition accuracy is approximately 98, 4% and 95, 7% respectively. We train our classifier using seven scene datasets, selected randomly, and test on the remaining three scenes. Table 1 presents the confusion matrices between the six classes over all 10 scenes. Our algorithm performs well on most per class accuracies with the heights accuracy 98% for Ground and the lowest 72% for sign-symbol. The global accuracy and per-class accuracy are about 94% and 87% respectively.

	Tree	*Car*	*Sign*	*Person*	*Fence*	*Ground*	*Building*
Tree	0.89	0.00	0.07	0.00	0.04	0.00	0.00
Car	0.03	0.95	0.00	0.00	0.02	0.00	0.00
Sign	0.17	0.00	0.72	0.11	0.00	0.00	0.00
Person	0.03	0.00	0.27	0.78	0.00	0.00	0.00
Fence	0.03	0.00	0.00	0.00	0.85	0.00	0.12
Ground	0.00	0.00	0.00	0.00	0.00	0.98	0.02
Building	0.00	0.00	0.00	0.00	0.04	0.00	0.96

Table 1. Confusion matrix Velodyne LiDAR Database

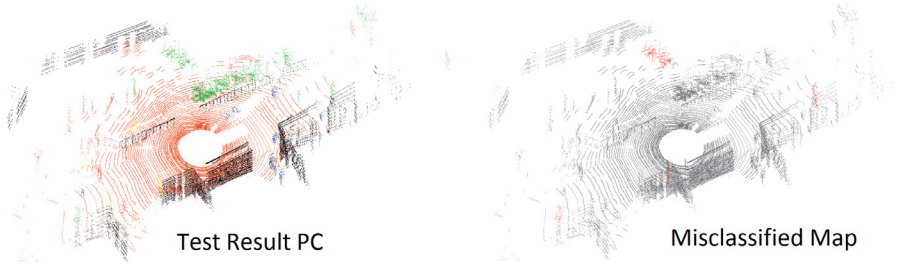


Fig. 5. Left image shows scene object recognition qualitative results, right image represent misclassified points.

Results	*Tree*	*Car*	*Sign*	*Person*	*Fence*	*Ground*	*Building*
Lai s	0.83	0.91	0.80	0.41	0.61	0.94	0.86
Our	0.89	0.95	0.72	0.88	0.85	0.98	0.95

Table 2. Comparison of the class accuracy of our approach and Lais approach

We also compare our approach to the method described by Lai in [20]. Table 1 shows its quantitative testing result. In terms of per class accuracy, we achieve 87% in comparison to 76%. Figure 7 shows some of the qualitative results of the test scene, achieved by our approach.

3.2 Evaluation Using Paris-rue-Madame and NAVTAQ True datasets

Paris-rue-Madame and NAVTAQ True datasets contains 3D MLS data. The Paris-rue-Madame point cloud is collected from rue Madame Street with 160 m long. The dataset contains 20 million points, 642 objects categorized in 26 classes. Its noteworthy that several objects such as wall sign and wall light are considered as building facades. The second MLS dataset is collected by NAVTAQ True system consisting of point cloud form New York streets. This LiDAR data was collected using terrestrial scanners and contains approximately 710 million points covering 1.2 km. These point clouds hold additional information such as RGB color, time step and etc. which is ignored here as our focus remained on using the pure geometry and intensity for the classification of objects. Same as TLS evaluating test we use 11 dominant categories: Building, Tree, Bike, Car, Sign-Symbol, Ground, Building. The Paris-rue-Madame and NAVTAQ True data sets are divided into two portions: the training set, and the testing set. The 70% long of each data set are randomly selected and mixed for training of classifier and 30% remained long of point cloud is used for testing. Table 3 shows the quantities results achieved by our approach.

Comparing to Terrestrial Laser Scanning, our results are not as good as in shown in Table 1. Since mixing two data sets captured from different cities poses

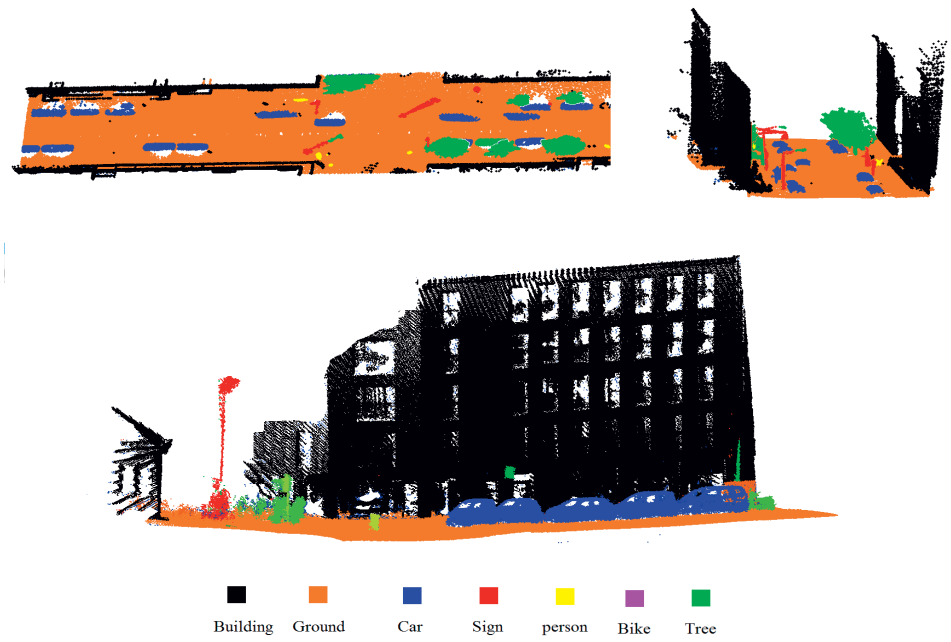


Fig. 6. Scene object recognition qualitative results in different view

serious challenges to the parsing pipeline. Furthermore, 3D street object detection is a much harder task than reconstructing walls or road surface. Because street objects can have virtually any shape and due to small resolution and the fact that the LiDAR only scans one side of the object, the detection is sometimes impossible. Moving objects are even harder to reconstruct based solely on LiDAR data. As these objects (typically vehicles, people) are moving through the scene, which make them appear like a long-drawn shadow in the registered point cloud. The long shadow artifact is not appear in TLS system because in which we face to one point as exposure point to scan the street objects. Figure 8 shows some of the qualitative results of the test scene.

	Tree	*Car*	*Sign*	*Person*	*Bike*	*Ground*	*Building*
Tree	0.75	0.07	0.10	0.00	0.00	0.00	0.08
Car	0.11	0.73	0.00	0.00	0.05	0.00	0.11
Sign	0.09	0.00	0.78	0.13	0.00	0.00	0.00
Person	0.07	0.00	0.21	0.58	0.14	0.00	0.00
Bike	0.03	0.00	0.00	0.04	0.81	0.00	0.12
Ground	0.00	0.00	0.00	0.00	0.00	0.97	0.03
Building	0.05	0.00	0.00	0.00	0.04	0.00	0.95

Table 3. Confusion matrix of Paris-rue-Madame and NAVTAQ True Database

4 Conclusion

We have proposed a novel and comprehensive framework for semantic parsing of street view 3D MLS and TLS point cloud based on geometrical features. First, ground are segmented using a heuristic approach based on the assumption of constant slope group plane. Second, building points are then extracted by tracing contours of projections of 3D points onto the x - y plane. Using this segmentation huge amount of data (more than 75% of points) are labeled, and only small amount of point cloud which have complex shape remained to be segmented. During the offline training phase 3D features are extracted at s-voxel level and are used to train boosted decision trees classifier. For new scene, the same unsupervised ground and building detection are applied and geometrical features are extracted and semantic labels are assigned to corresponding point cloud area. The proposed two-stage method requires only small amount of time for training while the classification accuracy is robust to different types of LiDAR point clouds acquisition methods. To our best knowledge, no existing methods have demonstrated the robustness with respect to variety in LiDAR point data.

References

1. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: Label transfer via dense scene alignment. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 1972–1979
2. Csurka, G., Perronnin, F.: A simple high performance approach to semantic segmentation. In: BMVC. (2008) 1–10
3. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. International Journal of Computer Vision **75** (2007) 151–172
4. Floros, G., Leibe, B.: Joint 2d-3d temporally consistent semantic segmentation of street scenes. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2823–2830
5. Zhang, G., Jia, J., Wong, T.T., Bao, H.: Consistent depth maps recovery from a video sequence. Pattern Analysis and Machine Intelligence, IEEE Transactions on **31** (2009) 974–988
6. Lu, W.L., Murphy, K.P., Little, J.J., Sheffer, A., Fu, H.: A hybrid conditional random field for estimating the underlying ground surface from airborne lidar data. Geoscience and Remote Sensing, IEEE Transactions on **47** (2009) 2913–2922
7. Hernández, J., Marcotegui, B., et al.: Filtering of artifacts and pavement segmentation from mobile lidar data. In: ISPRS Workshop Laserscanning 2009. (2009)
8. Zhou, Y., Yu, Y., Lu, G., Du, S.: Super-segments based classification of 3d urban street scenes. Int J Adv Robotic Sy **9** (2012)
9. Johnson, A.: Spin-Images: A Representation for 3-D Surface Matching. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (1997)
10. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3 d shape descriptors. In: Symposium on geometry processing. Volume 6. (2003)
11. Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. In: Computer Graphics Forum. Volume 28., Wiley Online Library (2009) 1383–1392

12. Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D.: Shape distributions. ACM Transactions on Graphics (TOG) **21** (2002) 807–832
13. Knopp, J., Prasad, M., Van Gool, L.: Orientation invariant 3d object classification using hough transform based methods. In: Proceedings of the ACM workshop on 3D object retrieval, ACM (2010) 15–20
14. Pavlidis, T.: Algorithms for graphics and image processing. Computer science press (1982)
15. Klasing, K., Althoff, D., Wollherr, D., Buss, M.: Comparison of surface normal estimation methods for range sensing applications. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE (2009) 3206–3211
16. Zhang, C., Wang, L., Yang, R.: Semantic segmentation of urban scenes using dense depth maps. In: Computer Vision–ECCV 2010. Springer (2010) 708–721
17. Babahajiani, P., Fan, L., Gabbouj, M.: Semantic parsing of street scene images using 3d lidar point cloud. Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops **13** (2013) 714 – 721
18. Xiao, J., Quan, L.: Multiple view semantic segmentation for street view images. In: Computer Vision, 2009 IEEE 12th International Conference on, IEEE (2009) 686–693
19. Collins, M., Schapire, R.E., Singer, Y.: Logistic regression, adaboost and bregman distances. Machine Learning **48** (2002) 253–285
20. Lai, K., Fox, D.: Object recognition in 3d point clouds using web data and domain adaptation. The International Journal of Robotics Research **29** (2010) 1019–1037
21. Serna, A., Marcotegui, B.: Attribute controlled reconstruction and adaptive mathematical morphology. In: Mathematical Morphology and Its Applications to Signal and Image Processing. Springer (2013) 207–218

PUBLICATION III

Comprehensive automated 3D urban environment modelling using terrestrial laser scanning point cloud

P.Babahajiani, L.Fan, J.-K. Kämäräinen, and M.Gabbouj

IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp.10-18

Comprehensive Automated 3D Urban Environment Modelling Using Terrestrial Laser Scanning Point Cloud

Pouria Babahajiani
Nokia Technologies
Tampere, Finland
pouria.babahajiani.ext@nokia.com

Lixin Fan
Nokia Technologies
Tampere, Finland
lixin.fan@nokia.com

Joni-Kristian Kämäräinen
Department of Signal Processing
Tampere University of Technology
Tampere, Finland
joni.kamarainen@tut.fi

Moncef Gabbouj
Department of Signal Processing
Tampere University of Technology
Tampere, Finland
moncef.gabbouj@tut.fi

Abstract

In this paper we present a novel street scene modelling framework, which takes advantage of 3D point cloud captured by a high definition LiDAR laser scanner. We propose an automatic and robust approach to detect, segment and classify urban objects from point clouds hence reconstructing a comprehensive 3D urban environment model. Our system first automatically segments grounds point cloud. Then building facades will be detected by using binary range image processing. Remained point cloud will be grouped into voxels and subsequently transformed into super voxels. Local 3D features are extracted from super voxels and classified by trained boosted decision trees with semantic classes e.g. tree, pedestrian, and car. Given labeled point cloud the proposed algorithm reconstructs the realistic model in two phases. Firstly building facades will be rendered by ShadVis algorithm. In the second step we apply a novel and fast method for fitting the solid predefined template mesh models to non-building labeled point cloud. The proposed method is evaluated both quantitatively and qualitatively on a challenging TLS NAVTEQ True databases.

1. Introduction

Analysis of 3D spaces comes from the demand to understand the environment surrounding us and to build more and more precise virtual representations of that space. 3D urban environment model is a digital representation of the earth′s surface and its related objects such as building, tree, vegetation, and some manmade feature belonging to the city area. There are various terms used for the 3D urban environment models such as “cyber town”, “virtual city”, or “digital city”. All of them are basically a computerized street model which contains the graphic representation of buildings and other objects in a 3D space. These 3D models are useful in variety of applications such as urban planning, crime prevention and control, virtual reality and robotic cars industry.

In this work, we propose a novel automatic 3D urban modeling approach which takes advantage of 3D geometrical features extracted from Light Detection And Ranging (LiDAR) point cloud. Since such 3D information is invariant to lighting and shadow, as a result, significantly more accurate results can be achieved.

While a laser scanning or LiDAR system provides a readily available solution for capturing spatial data in a fast, efficient and highly accurate way, the enormous volume of captured data often come with no semantic meanings. Some of these devices output several million data points per second. So efficient and fast methods are needed to filter the significant data out of these streams and high computing power is needed to post-process all this large amount of data. In despite of [1, 2], in this research we focus on a hybrid two stage voxel based classification to address the above mentioned challenges. Firstly, we adopt an unsupervised segmentation method to detect and remove dominant ground and buildings from other LiDAR data points, where these two dominant classes often correspond to the majority of point clouds. Secondly, after removing these two classes, we use a pre-trained boosted decision tree classifier to label local feature descriptors extracted from remaining vertical objects in the scene.

While most city modelling approaches are focused on facade modeling [3, 4, 5, 6], the focus of this work is automatic detection, segmentation and classification of whole urban objects. Given a point cloud containing one or more objects of interest and a set of labels corresponding to a set of models known to the system, the semantic segmentation system assigns correct labels to regions, or a set of regions, in the point cloud.

Given a labeled 3D point cloud with known position, orientation and shape, as well as the set of solid predefined street object templets, the 3D urban model will be generated.

1.1. Literature Review

While many image and video processing techniques for 2 dimensional object recognition have been proposed [7, 8], the accuracy of these systems is to some extent unsatisfactory because 2D image cues are sensitive to varying imaging conditions such as lighting, shadow and etc. In order to alleviate sensitiveness to different image

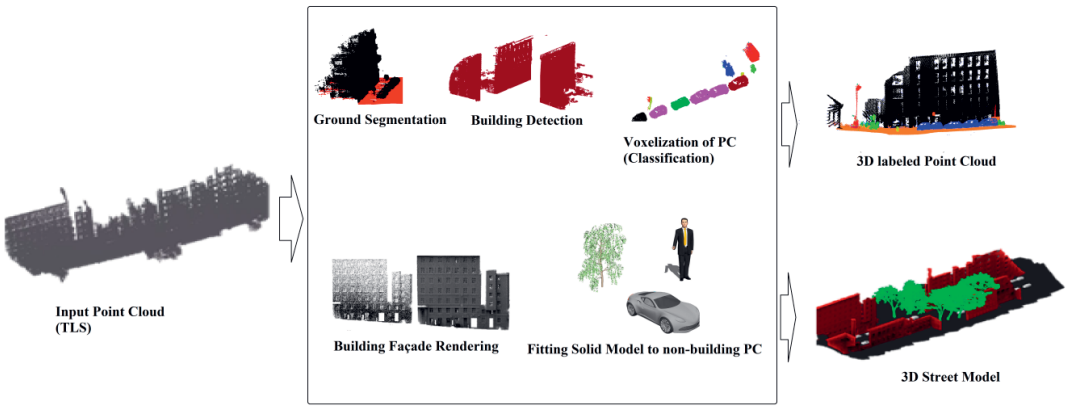


Fig 1. Framework of proposed mythology

capturing conditions, many efforts have been made to employ 3D scene features derived from single 2D images and thus achieving more accurate object recognition [9]. In the last decade, as the 3D sensors begun to spread and computing capacity for large scale 3D data processing became available, new methods and applications were born. Since such 3D information is invariant to lighting and shadow, as a result, significantly more accurate parsing results are achieved. 3D urban scene analysis had been often performed using 3D point cloud collected by airborne LiDAR for extracting vegetation and building structures [10]. Recently, classification of urban street objects using data obtained from mobile terrestrial systems has gained much interest because of the increasing demand of realistic 3D models for different objects common in urban era. A crucial processing step is the conversion of the laser scanner point cloud to a voxel data structure, which dramatically reduces the amount of data to process. Zhou and Yu (2012) present a voxel-based approach for object classification from TLS data [2].

Visual SLAM (Visual Simultaneous Localization and Mapping) has recently received a great attention within the robotics and vision communities which prepares consistent estimation of the 3D structure of the environment [13, 14] without considering the type and affordance of object in the scene.

Vosselman and Dijkman proposed a methodology and algorithm for 3D building model reconstruction from point clouds and ground plan [11]. They used the well-known Hough transform for the extraction of planar faces from the irregularly distributed point clouds. Ming et al., [12] investigated the methodology and algorithms for automatic generation of three dimensional photo realistic models from Lidar and image data. They implemented automatic 3D point cloud registration, automatic target recognition that is used for geo-referencing and automatic plane detection algorithm that is used for surface modeling, and texture mapping.

2. Proposed Methodology

It is a challenging task to directly extract objects from mobile LiDAR point cloud because of the noise in the data, huge data volume and movement of objects. We therefore take a hybrid two-stage approach to address the above mentioned challenges. Firstly, we adopt an unsupervised segmentation method to detect and remove dominant ground and buildings from other LiDAR data points, where these two dominant classes often correspond to the majority of point clouds. Secondly, after removing these two classes, we use a pre-trained boosted decision tree classifier to label local feature descriptors extracted from remaining vertical objects in the scene. This work shows that the combination of unsupervised segmentation and supervised classifiers provides a good trade-off between efficiency and accuracy. The output of classification phase is 3D labeled point cloud and each point is labeled with a predefined semantic classes such as building, tree, pedestrian and etc.

As photorealism cannot be achieved by using geometry cues alone, and because we aim to only use point cloud data (without image cues to decrease the procedure complexity) we present two post processing phases to enhance our proposed model visual quality. Firstly building facades will be rendered by ShadVis algorithm, then for non-building labeled point cloud we localize predefined template meshes to achieve more realistic model.

The contribution of this work are as follows:

- A complete scene parsing system is devised and experimentally validated using 3D urban scenes point cloud that have been gathered with LiDAR acquisition devices. The steps such as segmentation,

feature extraction, voxelization are generic and adaptable to solve object class recognition problems in different streets with varying landscape.

- Proposed two-stage (supervised and non-supervised) classification pipeline which requires only small amount of time for training.
- Propose to use novel geometric features leads to more robust classification results
- Generating textured façade of cities and localizing predefined templates for remained small objects such as car and pedestrian to enhance the visual quality of proposed model.

The framework of the proposed mythology is given in figure 1, in which 3D LiDAR point cloud is the inputs of the processing pipeline and parsing results are presented as 3D labeled point cloud. Final semi-photorealistic model of scene consisting textured 3D building facade and small template object are shown as post classification result.

A. Ground Segmentation

The aim of the first step is to remove points belonging to the scene ground including road and sidewalks, and as a result, the original point cloud are divided into ground and vertical object point clouds (Figure 2, A). Given a 3D point cloud of an urban street scene, the proposed approach starts by finding ground points by fitting a ground plane to the scene. This is because the ground connects almost all other objects and we will use a connect component based algorithm to over-segment the point clouds in the following step.

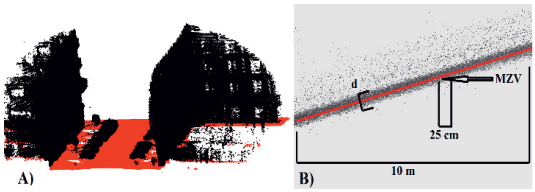


Fig 2. Ground Segmentation. A) Segmented ground and remained vertical objects point cloud are illustrated by red and black colour respectively. B) Sketch map of fitting plane to one tile

The plane RANSAC fitting method is used to approximate ground section of the scene. The RANSAC algorithm was developed by Fischler et al. [15] and is used to provide a more robust fitting of a model to input data in the presence of data outliers. Given a 3D point cloud of an urban street scene, the scene point cloud is first divided into sets of 10m×10m regular, non-overlapping tiles along the horizontal x–y plane. Then the following ground plane fitting method is repeatedly applied to each tile. We assume that ground points are of relatively small z values as compared to points belonging to other objects such as buildings or trees (see Fig 2).

The ground is not necessarily horizontal, yet we assume that there is a constant slope of the ground within each tile. Therefore, we first find the minimal-z-value (MZV) points within a multitude of 25cm×25cm grid cells at different locations. For each cell, neighboring points that are within a z-distance threshold from the MZV point are retained as candidate ground points. Subsequently, a RANSAC method is adopted to fit a plane to candidate ground points that are collected from all cells. Finally, 3D points that are within certain distance (d in Figure 2, B) from the fitted plane are considered as ground points of each tile. The approach is fully automatic and the change of two thresholds parameters do not lead to dramatic change in the results. On the other hand, the setting of grid cell size as 25cm×25cm maintains a good balance between accuracy and computational complexity.

B. Building Segmentation

Our method automatically extract building point cloud (e.g. doors, walls, facades, noisy scanned inner environment of building) based on two assumptions: a) building facades are sufficiently tall compare to the other structures in the street; and b) other non-building objects are located on the ground between two sides of street. As can be seen in figure 3, our method projects 3D point clouds to range images because they are convenient structures to process data. Range images are generated by projecting 3D points to horizontal x–y plane. In this way, several points are projected on the same range image pixel. We count the number of points that falls into each pixel and assign this number as a pixel intensity value. In addition, we select and store the maximal height among all projected points on the same pixel as height value. We define range images by making threshold and binarization of I, where I pixel value is defined as equation (1):

$$I_i = \frac{P_{intensity}}{\text{Max}\_P_{intensity}} + \frac{P_{\text{height}}}{\text{Max}\_P_{\text{height}}} \qquad (1)$$

Where I_i is grayscale range image pixel value, $P_{intensity}$ and P_{height} are intensity and height pixel value and $Max\_P_{intensity}$ and Max_P_{height} represent the maximum intensity and height value over the grayscale image. On the range image, an interpolation is required in order to fill holes caused by occlusions, missing scan lines and LiDAR back projection scatter.

In the next step we use morphological operation (e.g. close and erode) to merge neighbouring point and filling holes in the binary range images (figure 3). The morphological interpolation does not create new regional maxima, furthermore it can fill holes of any size and no parameters are required. Then we extract contours to find boundaries of objects. In order to trace contours, Pavlidis contour-tracing algorithm [16] is proposed to identify each contour as a sequence of edge points. The resulting

segments are checked on aspects such as size and diameters (height and width) to distinguish building from other objects. More specifically, equation (2) defines the geodesic elongation E(X), introduced by Lantuejoul and Maisonneuve (1984), of an object X, where S(X) is the area and L(X) is the geodesic diameter.

$$E(X) = \frac{\pi L^2(X)}{4S(X)} \qquad (2)$$

Considering the sizes and shape of buildings, the extracted boundary will be eliminated if its size is less than a threshold. The resolution of range image is the only projection parameter during this point cloud alignment that should be chosen carefully. If each pixel in the range image cover large area in 3D space too many points would be projected as one pixel and fine details would not be preserved. On the other hand, selecting large pixel size compared to real world resolution leads to connectivity problems which would no longer justify the use of range images. In our experiment, a pixel corresponds to a square of size 0.05 m^2.

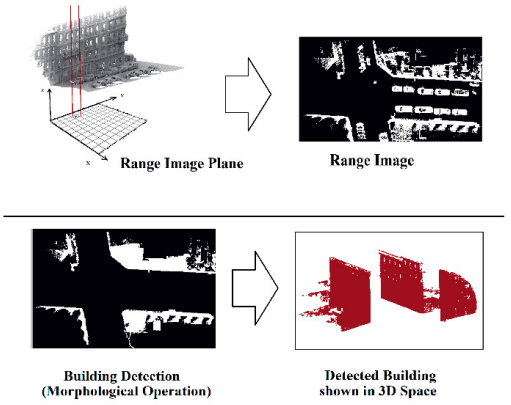


Fig 3. Building Segmentation

The 2D image scene is converted back to 3D space by extruding it orthogonally to the point cloud space. The x–y pixels coordinate of the binary image labeled as building facades are preserved as x–y coordinate of 3D point cloud (with open z value) labeled as building, and not considered in the remainder of our approach. Other points (negligible amount compare to the size of whole point cloud) are labeled as non-building class and will be later be classified as other classes e.g. car, tree, pedestrian and etc.

C. Voxel based classification

Although top view range image analysis generates a very fast segmentation result, there are a number of limitation to utilize it for the small vertical object such as pedestrian and cars. These limitations are overcome by using inner view (lateral) or ground based system in which, unlike top view the 3D data processing is done more precisely and the point view processing is closer to objects which provides a more detailed sampling of the objects. However, this leads to both advantages and disadvantages when processing the data. The disadvantage of this method includes the demand for more processing power required to handle the increased volume of 3D data.

According to voxel based segmentation, points which are merely a consequence of a discrete sampling of 3D objects are merged into voxels to represent enough discriminative features to label objects. 3D features such as intensity, area and normal angle are extracted based on these voxels. The voxel based classification method consists of three steps, voxelization of point cloud, merging of voxels into super-voxels and the supervised classification based on discriminative features extracted from super-voxels.

1) Voxelization of Point Cloud: In the voxelization step, an unorganized point cloud p is partitioned into small parts, called voxel v. The middle image in figure 4 illustrates an example of voxelization results, in which small vertical objects point cloud such as cars are broken into smaller partition. Different voxels are labelled with different colours. The aim of using voxelization is to reduce computation complexity and to form a higher level representation of point cloud scene. A collection of points is grouped together to form a variable size voxels. The criteria of including a new point pin into an existing voxel i is essentially determined by the crucial minimal distance threshold d_{th} which is defined as equation (3).

$$\min(\|P_{im} - P_{in}\|_2) \le d_{th}, \quad 0 \le m, n \le N, \quad m \ne n \qquad (3)$$

Where p_{im} is an existing 3D point in voxel, p_{in} is a candidate point to merge to the voxel, i is the voxel index, d_{th} is the maximum distance between two point and N is the maximum point number of a voxel. If the condition is met, the new point is added and the process repeats until no more point that satisfies the condition is found. Equation (3) ensures that the distance between one point and its nearest neighbours belonging to the same voxel is less than d_{th}. Although the maximum voxel size is predefined, the actual voxel sizes depend on the maximum number of points in the voxel (N) and minimum distance between the neighbouring points.

2) *Super Voxelization*: For transformation of a voxel to super voxel we propose an algorithm to merge voxels via

region growing with respect to the following properties of voxels:

- If the *minimal geometrical distance*, D_{ij}, between two voxels is smaller than a given threshold, where D_{ij} is defined as equation (4):

$$D_{ij} = \min\left(\|P_{ik} - P_{jl}\|_2 \right), \quad k \in (1, m), l \in (1, n) \qquad (4)$$

Where voxels v_i and v_j have m and n points respectively, and p_{ik} and p_{jl} are the 3D point belong to voxel v_i and v_j.

- If the angle between normal vectors of two voxels is smaller than a threshold: In this work, normal vector is calculated using PCA (Principal Component Analysis) [17]. The angle between two s-voxels is defined as angle between their normal vectors as equation (5):

$$\theta_{ij} = \arccos(< n_i , n_j >) \qquad (5)$$

Where n_i and n_j are normal vectors at v_i and v_j respectively. The proposed grouping algorithm merges the voxels by considering the geometrical distance ($D_{ij} < d_{th}$) and normal features of voxels ($\theta_{ij} < \theta_{th1}$). All these Voxelization steps then would be used in grouping these super-voxels (from now onwards referred to as s-voxels) into labeled objects.

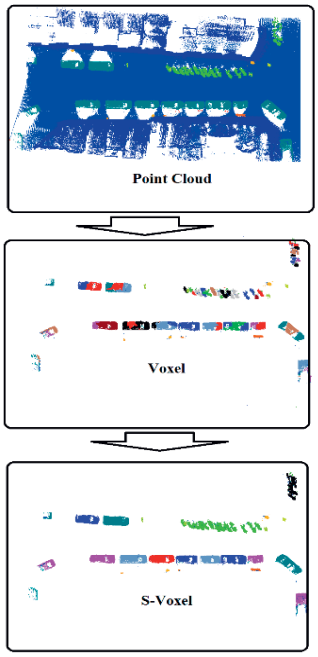


Fig 4. Voxelization of Point Cloud. from top to down: top view row point cloud, voxelization result of objects point cloud after removing ground and building, s-voxelization approach of point cloud

The advantage of this approach is that we can now use the reduced number of s-voxels instead of using thousands of points in the dataset, to obtain similar results for classification.

3) *Feature extraction:* For each s-voxel, seven main features are extracted to train the classifier.

Geometrical shape:
Projected bounding box has effective features due to the invariant dimension of objects. We extract four feature based on the projected bonding box to represent the geometry shape of objects.

-Area: the area of the bounding box is used for distinguishing large-scale objects and small ones.

-Edge ratio: the ratio of the long edge and short edge.

-Maximum edge: the maximum edge of bounding box.

-Covariance: is used to find relationships between points spreading along two largest edges.

Height above ground: Given a collection of 3D points with known geographic coordinates, the median height of all points is considered as the height feature of the s-voxel. The height information is independent of camera pose and is calculated by measuring the distance between points and the road ground.

Horizontal distance to center line of street: Following [1], we compute the horizontal distance of the each s-voxel to the centre line of street as second geographical feature. The street line is estimated by fitting a quadratic curve to the segmented ground.

Density: Some objects with porous structure such as fence and car with windows, have lower density of point cloud as compared to others such as trees and vegetation. Therefore, the number of 3D points in a s-voxel is used as a strong cue to distinguish different classes.

Intensity: Following [1], LiDAR systems provide not only positioning information but also reflectance property, referred to as intensity, of laser scanned objects. This intensity feature is used in our system, in combination with other features, to classify 3D points. More specifically, the median intensity of points in each s-voxel is used to train the classifier.

Normal angle: Following [18], we adopt a more accurate method to compute the surface normal by fitting a plane to the 3D points in each s-voxel. The surface normal is important properties of a geometric surface, and is frequently used to determine the orientation and general shape of objects.

Planarity: Patch planarity is defined as the average square distance of all 3D points from the best fitted plane

computed by RANSAC algorithm. This feature is useful for distinguishing planar objects with smooth surface like cars form non planar ones such as trees.

4) *Classifier:* The Boosted decision tree [19] has demonstrated superior classification accuracy and robustness in many multi-class classification tasks. Acting as weaker learners, decision trees automatically select features that are relevant to the given classification problem. Given different weights of training samples, multiple trees are trained to minimize average classification errors. Subsequently, boosting is done by logistic regression version of Adaboost to achieve higher accuracy with multiple trees combined together. Each decision tree provides a partitioning of the data and outputs a confidence-weighted decision which is the class-conditional log-likelihood ratio for the current weighted distribution. In our experiments, we boost 10 decision trees each of which has 6 leaf nodes.

D. City model reconstruction using labeled point cloud

The 3D modelling methods are mainly categorized based on the input data techniques, one is photogrammetry (aerial, satellite and close range based model), and another one is the laser techniques (aerial, and terrestrial based model) which is the subject of this work. Most of the time the 3D model is generated using the photogrammetry techniques or when the model is built by laser scanners the registered image data is used to reconstruct textured 3D facades model. In these model implementation the 3D point cloud is registered with image data, automatic plane detection is used for surface modelling and texture mapping will be done using image data [4, 11]. Based on our knowledge, there is no literature review available on 3D city modelling just using 3D laser geometric data to generate realistic models without color imaging cues.

This subsection illustrates our approach that reconstructs the realistic model using input labeled point cloud in two phases. Firstly we use ShadVis algorithm [20] to render the building facades since the algorithm calculates the illumination of a point cloud (or vertices of a mesh) with the light coming from a theoretical hemisphere or sphere around the object. In the second step we apply a methods for fitting the solid predefined template models to non-building labeled point cloud.

1) *Building façade rendering:*

Recently, point-based geometry has become a very popular 3D object representation for geometry processing and graphics. The design of rendering tools using this point wise geometry representation is relatively straightforward. In terms of computer graphics research, this part has been inspired by previous research in point-based geometry and skydome/terrain rendering [20, 21].

We use the data structure of [22] for our point-based representation of the data since it allows flexible multi-level rendering with small overhead.

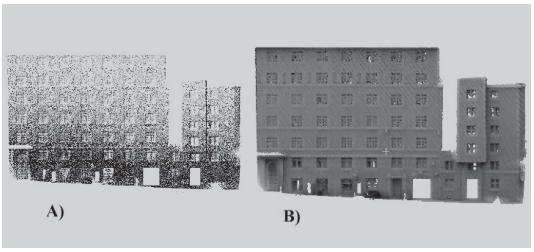


Fig 5. A) Input labelled point cloud representing building facades. B) The rendering result with the 3D skydome approach.

Even though graphics hardware rendering pipelines have been designed for polygons, the rendering of points is even easier than for polygons. So as [20], we use the adapted version algorithm of [22] for our purposes since it could only process surfaces and we have points as input. The accuracy of the result is concerned with the resolution of the 3D data. A snapshot from the 3D viewer of a facades with approximately 15 m length is shown in figure 5.

2) *Method for fitting solid model to non-building labelled point cloud*

In the last part we generate 3D building model using just 3D point cloud. Due to complexity of different urban scene most studies have been focused just on facades or building modelling [3, 4]. In this step we present the method based on fitting predefined template street view object to non-building label point cloud (such as car, tree, pedestrian and etc.) to devise a complete virtual 3D model of the urban scene.

The input of this approach is the classified labeled point cloud and the output is the solid meshes. We divided the street view object categories modelling into two subsets and adopt different object fitting approaches for them. The first subset is related to the object classes which their solid mesh structure orientation is not important and their object models will be completed based on only their position and dimension. These class model fitting include tree, pedestrian, sign symbol and etc. Unlike a lot of work which calculate the distance of a given points to the closet surface and use time consuming iterative procedure to fit the solid model into the point cloud or reconstructed surface [23], we propose a novel approach to solve this problem in a straightforward and computationally lightweight manner.

For each separated point cloud collection, the center and its boundaries will be calculated. Based on the size of the existing solid library meshes, we localize the best

isodiametric meshes to the point cloud. As the object orientation is not important we only fit the mesh by stretching it to an appropriate size.

Object orientation should be considered for the second street view object categories such as car, bus, bike and general vehicle, therefore, we propose to fit the model based on the bounding box center matching. Similar to subset one the center of mesh and point cloud will be matched and then the corresponding model will be chosen from library based on the dimension of the vehicle bounding box. Then Iterative Closet Point (ICP) algorithm [23] is applied to automatically refine the registration of two entities.

All the fitting method assume that the correspondence between the points and the meshes will be successfully resolved during iterations of the fitting, unless after several iteration the orientation of the last vehicle will be considered. In figure 6 we show the result of fitting model to the point cloud. Notice that even with the difference in target and template type of the car (the solid template mesh is sedan and point cloud is hatchback) the pose is recovered accurately.

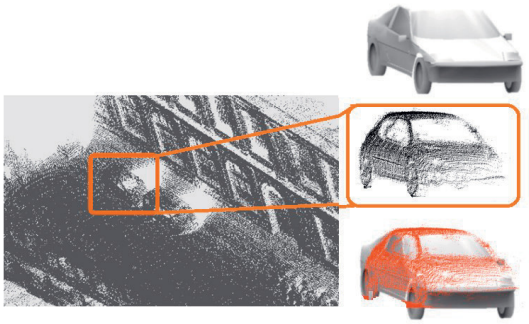


Fig 6. Fitting the mesh model into labeled vehicle point cloud. The top car is one of the solid model candidate. The middle car is the segmented point cloud which is labeled using supervised classifier. And the bottom image shows the fitting result.

The proposed method takes advantage of priori knowledge about urban scene environment and assumes that there are enough distance between different object in the street so that they are not connected.

3. Experimental Result

Our methodology is evaluated on three Mobile Laser Scanning (MLS) databases to get comparative results with state of the art: two NAVTAQ True [1] from Helsinki and Chicago, and rues Soufflot dataset from Paris [24].

As a general remark, our experiments starts with ground and building segmentation then we train boosted decision tree classifiers with sample 3D features extracted from training s-voxels. Subsequently we test the performance of the trained classifier using separated test samples. The accuracy of each test is evaluated by comparing the ground truth with the scene parsing results. Since our proposed modeling approach (fitting model to street view point cloud) has not been done before in existing work, we only compare the proposed classification approach both quantitatively and qualitatively.

We created and used labeled dataset of driving sequence from NAVTAQ True, provided by HERE, consists of best of sensors in positioning and LiDAR. The two NAVTAQ point cloud datasets contains more than 800 million points covering approximately 2.4 km altogether.7 semantic object classes are defined to label the LiDAR dataset: building, tree, car, sign symbol, person, fence and ground. It's noteworthy that several objects such as wall sign and wall light are considered as building facades.

The whole two NAVTAQ True datasets are mixed and divided into two portions: the training set, and the testing set. The 70% long of dataset are randomly selected and mixed for training of classifier and 30% remained long of point cloud is used for testing. Table 1 shows the quantities results achieved by our approach.

Table 1. Confusion matrix, NAVTAQ True datasets

	Tree	Car	Sign	person	Fence	Ground	Building
Tree	0.89	0.00	0.07	0.00	0.04	0.00	0.00
Car	0.03	0.95	0.00	0.00	0.02	0.00	0.00
Sign	0.17	0.00	0.72	0.11	0.00	0.00	0.00
person	0.02	0.00	0.20	0.78	0.00	0.00	0.00
Fence	0.03	0.00	0.00	0.00	0.85	0.00	0.12
Ground	0.00	0.00	0.00	0.00	0.00	0.98	0.02
Building	0.00	0.00	0.00	0.00	0.04	0.00	0.96

Mixing data from different cities poses serious challenges to the parsing pipeline, which is reflected by the decrease in the class average accuracy. Nevertheless, it seems our algorithm performs well on most per class accuracies, with the highest accuracy 96% achieved for the building and ground and the lowest 72% for sign. This low accuracy for small objects (e.g. person, sign) is mainly due to lack of sufficient training examples, which naturally lead to a less statistically significant labelling for objects in these classes. Moving objects are also hard to be reconstructed based solely on 3D data. As these objects (typically vehicles, people) are moving through the scene, which make them appear like a long-drawn shadow in the registered point cloud. The global accuracy is about 91 %. As it can be seen in the figure 7, successful point cloud classification and alignment have been done accurately.

It is noteworthy that our algorithms were initially developed to process NAVTAQ True data sets. One of the main advantages of our method is that it can be easily generalized to other datasets without any major

modification. We compare our approach to the method described by Hernandez in [24].

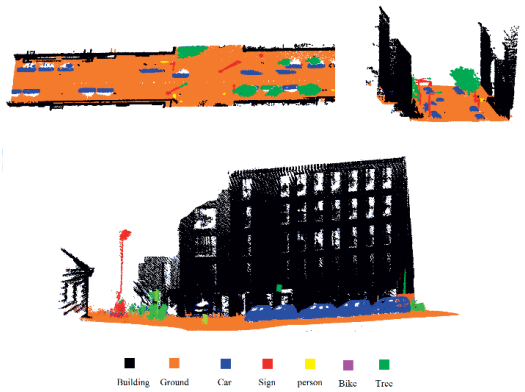


Figure 7. 3D Scene object recognition qualitative results in different view (point cloud), NAVTAQ True dataset

They use MLS labeled dataset from rue Soufflot with approximatively 500 m including pedestrians, cars, and lampposts to test their pipeline accuracy. The SVM classifier has been trained based on geometrical and contextual features to label the point cloud. Table 2 compare quantities results on the test set.

Table 2: Classification results on rue Soufflot test set.

	Lampposts	Car	Pedestrians	Ground	Building
Hern´andez	1.00	0.92	0.70	-	-
Our	0.95	0.95	0.86	1	1

Our algorithm performs well on most per class accuracies however main problems appear with Lampposts classes because training dataset does not contain enough information to train the classifier well.

Comparing to the NAVTAQ True experiment, the per class accuracy in the Soufflot experiment is much higher because the NAVTAQ datasets scenes have more complex landscape, in addition the quality of the LiDAR NAVTAQ (data and groundtruth) is quite fragile. Furthermore, mixing two NAVTAQ datasets from different cities poses serious challenges to the classification pipeline, which is reflected by the decrease in the class average accuracy.

4. Conclusion

We have proposed a novel and comprehensive framework for 3D model of street view scene. Using unsupervised segmentation huge amount of data (more than 75% of points) are labeled, and only small amount of point cloud which have complex shape remained to be segmented. The proposed two-stage method requires only small amount of time for training while the classification accuracy is robust to different LiDAR point clouds datasets. We have processed data on a large scale and achieved satisfactory accuracy and performance (figure 8). To our best knowledge, no existing methods have demonstrated the robustness with respect to variety in LiDAR point datasets.

This algorithm was done based on characteristics of objects in urban street scenes but it can be done for other environment like faubourg, country or even indoor spaces by inspecting characteristics of its objects and appropriate definition of constraints.

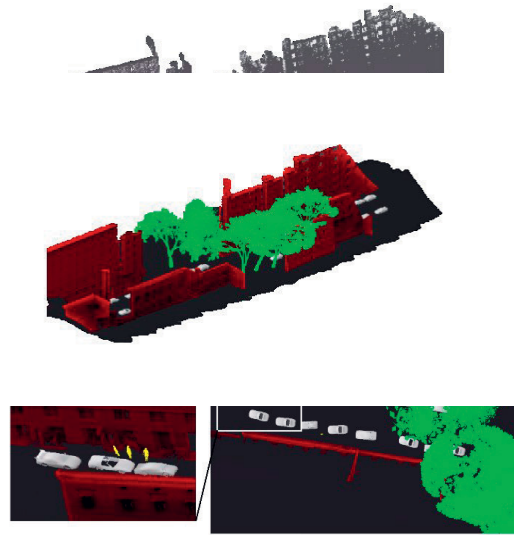


Figure 8. 3D city modelling. (Top): Raw point cloud. (Two down images): Resulted model in different view

References

[1] P. Babahajiani, L. Fan, M. Gabbouj, Object Recognition in 3D Point Cloud of Urban Street Scene, IEEE Asian Conference on Computer Vision (ACCV), Singapore 2014.

[2] Y. Zhou, Y. Yu, G. Lu, S. Du, Super-segments based classification of 3D urban street scenes. Int J Adv Robotic Sy 9, 2012.

[3] C. Dold and C. Brenner, Registration of terrestrial laser scanning data using planar patches and image data, in International Archives of Photogrammetry and Remote Sensing, pp. 25– 27, 2006.

[4] G. Stamos, G. Yu, Wolberg, and S. Zokai, 3d modeling using planar segments and mesh elements, in 3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and

Transmission. Washington, DC, USA: IEEE Computer Society, pp. 599–606, 2006.
[5] S. Becker and N. Haala, Combined feature extraction for facade reconstruction, in ISPRS Workshop on Laser Scanning and SilviLaser, Espoo, Finland, pp. 44–50, 2007.
[6] H. Boulaassal, T. Landes, P. Grussenmeyer, and F. TarshaKurdi, Automatic segmentation of building facades using terrestrial laser data, Laser, p. 65, 2007.
[7] C. Liu, J. Yuen, Torralba, nonparametric scene parsing: Label transfer via dense scene alignment. In: Computer Vision and Pattern Recognition, CVPR 2009. IEEE Conference on, IEEE 2009.
[8] G. Csurka, F. Perronnin, A simple high performance approach to semantic segmentation. In: BMVC, 2008
[9] G. Floros, B. Leibe, Joint 2d-3d temporally consistent semantic segmentation of street scenes. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2823-2830.
[10] W. L Lu, K. P. Murphy, and J. J. Little, A. Shefier, H. Fu, A hybrid conditional random field for estimating the underlying ground surface from airborne lidar data, Geoscience and Remote Sensing, IEEE Transactions on 47 (2009) 2913-2922.
[11] G. Vosselman, S. Dijkman, 3D building model reconstruction from point clouds and ground plan, International Archives of Photogrammetry and Remote Sensing, Volume XXXIV-3/W4 Annapolis, MD, 22-24 Oct 2001.
[12] J. Ming, Li-Chee, D. Gumerov, T. Ciobanu, C. Armenakis, Generation of Three Dimensional Photo-Realistic Models from Lidar and Image Data, IEEE-TIC-STH, 2009
[13] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6):1–16, 2007
[14] H. Jin, P. Favaro, and S. Soatto. A semi-direct approach to structure from motion. The Visual Computer, 19:1–18, 2003.
[15] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24 (6), 381–395, 1981.
[16] T. Pavlidis, Algorithms for graphics and image processing. Computer science press, 1982.
[17] K. Klasing, D. Althoff, D. Wollherr, M. Buss, Comparison of surface normal estimation methods for range sensing applications. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE (2009) 3206–3211, 2009
[18] J. Xiao, L. Quan, Multiple view semantic segmentation for street view images. In: Computer Vision, 2009 IEEE 12th International Conference on, IEEE 2009.
[19] M. Collins, R. E. Schapire, Y. Singer, Logistic regression, adaboost and bregman distances. Machine Learning 48, 2002.
[20] D. Florent, D. George, G. Daniel, M. Jean-Luc, and S. Francis. A point-based approach for capture, display and illustration of very complex archeological artefacts. In Proceedings of the 5th International conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage (VAST'04), 2004
[21] L. Max, Horizon mapping: shadows for bump-mapped surfaces. The Visual Computer 4, 2 (July 1988), 109–117.
[22] D. Duguet, G. Drettakis, Flexible pointbased rendering on mobile devices. IEEE Computer Graphics and Applications 24, 4, July-August 2004.
[23] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm, Proc. Int'l Conf. on Recent Advances in 3D Digital Imaging and Modeling, Québec City, Canada, 145-52, 2001.
[24] J. Hern´andez, B. Marcotegui, Point cloud segmentation towards urban ground modeling. In: The 5th GRSS/ISPRS Joint Urban Remote Sensing Event (URBAN2009). Shangai, China, pp. 1–5, 2009b

PUBLICATION IV

Urban 3D segmentation and modelling from street view images and LiDAR point clouds

P.Babahajiani, L.Fan, J.-K. Kämäräinen, and M.Gabbouj

Machine Vision and Applications, 28.7, 2017, DOI 10.1007/s00138-017-0845-3.

Machine Vision and Applications (2017) 28:679–694
DOI 10.1007/s00138-017-0845-3

SPECIAL ISSUE PAPER

Urban 3D segmentation and modelling from street view images and LiDAR point clouds

Pouria Babahajiani[1] · **Lixin Fan**[1] · **Joni-Kristian Kämäräinen**[2] · **Moncef Gabbouj**[2]

Received: 4 September 2016 / Revised: 23 March 2017 / Accepted: 26 April 2017 / Published online: 29 May 2017

Abstract 3D urban maps with semantic labels and metric information are not only essential for the next generation robots such autonomous vehicles and city drones, but also help to visualize and augment local environment in mobile user applications. The machine vision challenge is to generate accurate urban maps from existing data with minimal manual annotation. In this work, we propose a novel methodology that takes GPS registered LiDAR (Light Detection And Ranging) point clouds and street view images as inputs and creates semantic labels for the 3D points clouds using a hybrid of rule-based parsing and learning-based labelling that combine point cloud and photometric features. The rule-based parsing boosts segmentation of simple and large structures such as street surfaces and building facades that span almost 75% of the point cloud data. For more complex structures, such as cars, trees and pedestrians, we adopt boosted decision trees that exploit both structure (LiDAR) and photometric (street view) features. We provide qualitative examples of our methodology in 3D visualization where we construct parametric graphical models from labelled data and in 2D image segmentation where 3D labels are back projected to the street view images. In quantitative evaluation we report classification accuracy and computing times and compare results to competing methods with three popular databases: NAVTEQ True, Paris-Rue-Madame and TLS (terrestrial laser scanned) Velodyne.

Keywords Urban 3D · Point cloud · LiDAR · Street view · Semantic segmentation · Robotics

Pouria Babahajiani
pouria.babahajiani.ext@nokia.com

Lixin Fan
Lixin.Fan@nokia.com

Joni-Kristian Kämäräinen
joni.kamarainen@tut.fi

Moncef Gabbouj
Moncef.Gabbouj@tut.fi

[1] Nokia Technologies, Tampere, Finland

[2] Department of Signal Processing, Tampere University of Technology, Tampere, Finland

1 Introduction

3D urban map model is a digital representation of the earths surface at city locations consisting of terrestrial objects such as buildings, trees, vegetation and manmade objects belonging to the city area. 3D maps are useful in different applications such as architecture and civil engineering, virtual and augmented reality, and modern robotics (autonomous cars and city drones). Creating photorealistic and accurate 3D urban maps requires high volume and expensive data collection. For example, Google and Nokia HERE have cars mounted with cameras and Light Detection And Ranging (LiDAR) scanners to capture 3D point cloud and street view data along streets throughout the world. While laser scanning or LiDAR systems provide a readily available solution for capturing spatial data in a fast, efficient and highly accurate way, the semantic labelling of data would require enormous man power if done manually. Therefore, the problem of automatic labelling (parsing) of 3D urban data to associate each 3D point with a semantic class label (such as "car", "tree") has gained momentum in the computer vision community [11,17,24,42].

Automatic segmentation and labelling of urban point cloud data is challenging due to a number of data specific challenges. First, high-end laser scanning devices output millions of data points per second, and therefore the methods need to be efficient to cope with the sheer volume of the

urban scene datasets. Second, point cloud sub-regions corresponding to individual objects are imbalanced, varying from sparse representations of distant objects to dense clouds of nearby objects, and incomplete (only one side of objects is scanned by LiDAR). Third, for accurate object recognition a sufficiently large labelled training data (ground truth) are needed to train the best supervised methods.

In this work, we tackle the efficiency issue by proposing a hybrid method which consists of following three steps: First, certain simple but frequently occurring structures, such as building facades and ground surface, are quickly segmented by rule-based methods. The rule-based method can typically label 70–80% of the point cloud data and rule-based methods are more than 6× faster than the otherwise efficient boosted decision trees [17]. Second, the remaining points are processed with our fast supervised classifier. To construct high-quality features for our classifier, we first over-segment the points to 3D voxels which are further joined into super-voxels from which structure features are extracted. Moreover, as the 3D points are aligned with street view images we also extract photometric features. Our classifier of choice is a boosted decision tree classifier which is trained to label the remaining points using the super-voxel features. Third, We solve the problem of incomplete data by utilizing parametric 3D templates of certain classes (cars, trees and pedestrians) and fit them to the boosted decision tree labelled super-voxel point clouds. The final step also improves the visual quality of the semantic 3D models output from our processing pipeline, especially for those sparse and incomplete point clouds corresponding to small objects. Another application of our method is semantic segmentation of street view images which is achieved by backprojecting the semantic labels of the point cloud points to the corresponding street view images. Figure 1 depicts the overall workflow of our method. We provide qualitative examples of 3D visualization and 2D segmentation and in quantitative experiments we report and compare our segmentation accuracy and computing time to previous works. This work is based on the preliminary results in [2,3], but provides a significant extension since it contains experimental results on three publicly available datasets, comparison to other recent works, refined processing steps and an extensive ablation study over the method parameters.

Contributions Preliminary results on components of our processing pipeline have been reported in [2,3], and in this work we make the following novel contributions:

- We have demonstrated a complete urban map data processing pipeline, which annotate all 3D LiDAR points with semantic labels. Our method is made efficient by combining fast rule-based processing for building and street surface segmentation and super-voxel-based feature extraction and classification for remaining map elements (cars, pedestrians, trees and traffic signs).
- We propose two back ends for semantically labelled urban 3D map data that exemplify two important applications: (i) 3D urban map visualization and (ii) semantic segmentation of 2D street view images by backprojection of the 3D labels.
- Parameters of the different processing stages have clear physical and intuitive meaning, and therefore they are easy to set for novel data or optimize by cross-validation over certain ranges. We have made extensive experiments on larger datasets, and moreover, optimal parameter settings are cross-validated against labelled datasets. Experimental results verify superior accuracy and efficiency of our method as compared to the existing works on three difficult datasets.

As such we provide full processing pipeline from 3D LiDAR point cloud and street view image data (cf. Google Maps and Nokia HERE) to urban 3D map data visualization and to 2D semantic segmentation. All parameters have physical meaning, and the system automatically adapts to the dataset size.

2 Related work

3D segmentation and labelling (classification) using image and point cloud data of urban environments have many potential applications in augmented reality and robotics and therefore research on these topics has gained momentum during the last few years. In the following, we briefly mention the most important 2D approaches, but focus on 3D point cloud methods and methods particularly tailored for urban 3D processing. Several important surveys have been recently published where more details of specific approaches can be found [26,27,37].

Image-based methods Due to the lack of affordable and high-quality 3D sensors until the introduction of Kinect in 2010 the vast majority of the works is still based on 2D (RGB) image processing. Progress in 2D over the years has been remarkable and for 2D object classification and detection there have been several breakthrough methodologies [23,45], in particular, the visual Bag-of-Words (BoW) [6,35], Scale Invariant Feature Transform (SIFT) [46] and the Deformable Part Model (DPM) [9]. Recently, these methods have been superseded by methods using deep convolutional neural networks, e.g. AlexNet [19] and R-CNN [10]. Direct applicability of these methods is unclear since the datasets used in training contain objects in various non-urban environments (ImageNet, Pascal VOC) and sources of detection failures may therefore be different. The deep neural network methods also require large annotated datasets. Moreover, mapping the 2D

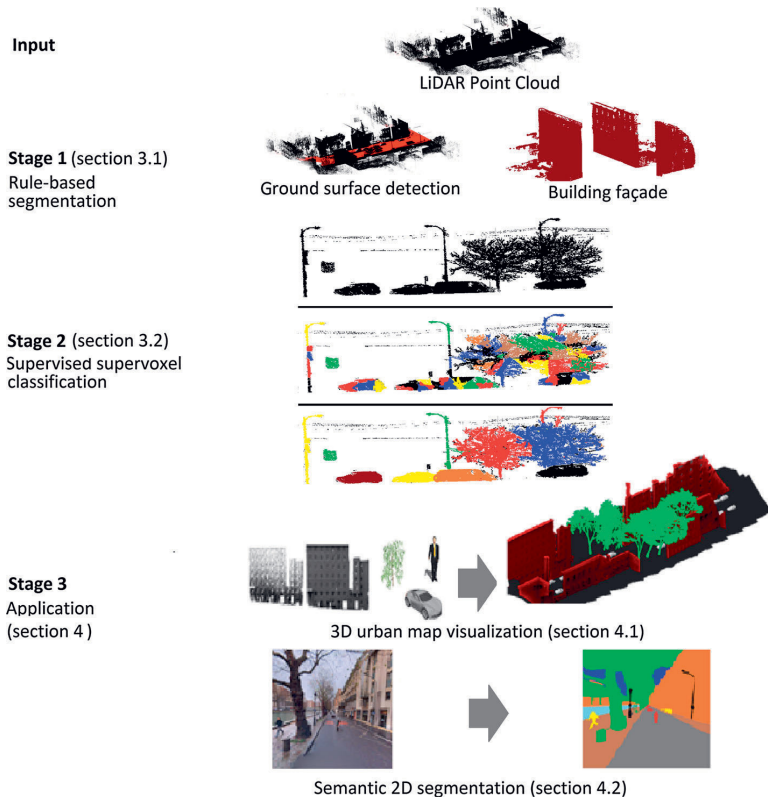


Fig. 1 The overall workflow of the proposed methodology

bounding boxes to 3D point cloud object boundaries is not trivial. To summarize, state-of-the-art 2D methods provide a potential research direction as combined with state-of-the-art 3D methods, but in this work we focus on methods particularly developed for urban 3D map data segmentation.

Point cloud-based methods The success of local descriptors in 2D has inspired to develop 3D local descriptors for point cloud data, e.g. 3D SURF (speeded up robust features) [18] and 3D HOG (histogram of oriented gradients) and DoG (difference of Gaussians) [43], and their comparison can be found from the two recent surveys [13,14]. These methods and also many direct point cloud-based methods, e.g. [4,15,28], are designed to recognize a specific object stored as a point cloud model, and therefore practical use of these methods for urban 3D often requires various geometric features to robustify matching [38].

Urban 3D segmentation Most of the existing city modelling approaches directly or indirectly tackle the problem through 3D point cloud analysis. Lafarge and Mallet [20] applied a Markov Random Field (MRF) -based on optimization technique, using the graph-cut framework for object detection using airborne laser scanner (ALS) data. In this work, we omit 3D data generated by airborne devices (see, e.g. [12,20]) and assume that the 3D map data have been collected via terrestrial and mobile laser scanning—this kind of data is available, for example, in Google Maps and Nokia HERE maps where the 3D data are mobile laser scanner (MLS) LiDAR generated point cloud. It is noteworthy that urban 3D segmentation has also been investigated for stereo-generated point clouds [32], but there noise level is orders of magnitude higher and therefore we focus on high-quality LiDAR data. Douillard et al. [8] compared various segmentation approaches for dense and sparse LiDAR data and found simple clustering methods the best and noted that street pre-segmentation improves the results. These findings were verified in the survey by Nguyen and Le [27] who also pointed that learning-based methods are needed for more complex objects due to noise, uneven density and occlusions. Inspired by these two important findings, we adopt a fast rule-based approach for simple and frequently appearing structures (streets and building facades) and the learning-based approach for more complex structures. The combination of clustering, extracting geometric features and using a supervised classifier to recognize objects was proposed in [11], but in our approach we accelerate computation by the rule-based pre-processing and by adopting the efficient super-voxel clustering that has been used in video process-

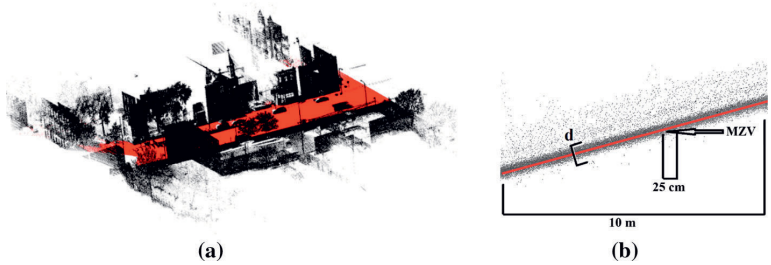


Fig. 2 Example of rule-based segmentation of road surfaces. **a** 3D LiDAR point cloud segmented to *road surface points* (*red*) and *other points* (*black*); **b** a sketch illustrating our plane fitting to one tile

ing [41]. Fast 3D-only methods exist [24], but it has also been argued that joint 2D image cues (e.g. colour, texture, shape) and 3D information (point cloud) provide higher accuracy [40,42,44] and therefore we collect features from 3D and 2D for our classifier.

3 Proposed methodology

The overall processing steps of our approach are illustrated in Fig. 1. The input to our processing algorithm is 3D LiDAR point cloud $\mathcal{P} = \{\boldsymbol{p}_i\}$ ($\boldsymbol{p} \in \mathbb{R}^3$) and street view images $\mathcal{I} = \{\boldsymbol{I}_i\}$ ($\boldsymbol{I} \in \mathbb{R}^{W \times H \times 3}$). The camera and LiDAR sensors are calibrated with respect to each other. The first processing step of our methodology is the rule-based segmentation of the ground surface (Sect. 3.1.1) and building facades (Sect. 3.1.2). The points labelled by the rule-based processing cover approximately 75% of the urban city data, and the remaining points proceed to the next step. The next step is super-voxel clustering (Sect. 3.2.1) and feature extraction from each super-voxel after which the super-voxels are classified using the boosted decision tree classifier (Sect. 3.2.2). The output of the method is a fully labelled 3D point cloud where each point is labelled to belong to one of the pre-defined semantic classes (Fig. 1). We also present two different applications of our system: (i) 3D urban map visualization (Sect. 4.1) by using parametric models generated from the labelled super-voxels and (ii) 2D segmentation (Sect. 4.2) by mapping the 3D labels to the RGB images.

3.1 Rule-based segmentation of simple structures

Our empirical findings are in align with [8,27] which pointed clear computational advances for using pre-processing to segment geometrically simple and dominating structures. Therefore, we devise simple rule-based detectors for *road surfaces* and *building facades* that span majority of urban scene point clouds (75% on average in the datasets used in the experiments). Both road surfaces and building facades form large and dense horizontal and vertical planar regions, and therefore it is easy to devise geometric rules constraining them and providing fast segmentation as compared to the learning-based approaches. The rule-based steps detect and label the road surface points $\mathcal{P}_{\text{road}}$, and building points $\mathcal{P}_{\text{building}}$ which are removed from the original point cloud $\mathcal{P}' = \mathcal{P} \setminus (\mathcal{P}_{\text{road}} \cup \mathcal{P}_{\text{building}})$ and then passed to the next processing step (learning-based segmentation).

3.1.1 Road surfaces

The goal of the first step is to detect road surface points including the car path and side-walk, and as a result, the original point cloud is divided into *road surface* ($\mathcal{P}_{\text{road}}$) and other ($\mathcal{P}_{\text{other}} = \mathcal{P} \setminus \mathcal{P}_{\text{road}}$) points (Fig. 2). Starting from the road surfaces is also beneficial for the later steps that are based on point cloud connectivity as the road and ground surfaces connect almost all points together. We apply the fast and robust Random sample consensus (RANSAC)-based plane fitting similar to Lai and Fox [21] who used it to remove planar regions from Google 3D Warehouse point clouds.

To adapt the Lai and Fox algorithm for our case of large urban city maps we need to do two additional steps: (i) local fitting and (ii) windowed candidate surface point selection. The first step is needed to allow varying street slope (cf. "San Francisco" landscape). The second step is needed to decrease the total number of points for plane fitting to make it faster.

Therefore, our adapted algorithm consists of following three steps: First, the original point cloud is divided into smaller point clouds $\{\mathcal{P}_{10\text{ m}\times 10\text{ m}}\}_k$ which span 10 × 10 m square areas. Secondly, each $\{\mathcal{P}_{10\text{ m}\times 10\text{ m}}\}_k$ point cloud is further divided into 0.25 m × 0.25 m cells, and for each cell the Minimal-Z-Value (MZV) is computed by averaging 10 lowest z-value points.[1] Thirdly, for each cell, all points lying within $\pm\tau_{\text{MZV}}$ distance from MZV are selected for plane fitting (Fig. 2). The selection process reduces the number of points to around 0.1% from the original, and in all experiments we fixed the threshold to $\tau_{\text{MZV}} = 0.02$ m. For each local point cloud the points that are within the distance $\tau_{\text{road}} = \pm 0.08$ m from the fitted plane are added to $\mathcal{P}_{\text{road}}$. The average processing time of a single 10 m × 10 m region is about 15 ms. This approach is not sensitive to the selec-

[1] Using 10 lowest z-value points is to make the MZV estimation insensitive to outliers.

tion of the two thresholds and efficiently segments the road surface points.

3.1.2 Building facades

The workflow of our rule-based building facade segmentation is shown in Fig. 3. At first, we form a GPS-defined x-y plane similar to the road surface detection and divide the plane to 0.25 m × 0.25 m cells. Our detection rules are derived from the dominant characteristics of building facades in LiDAR point clouds: LiDAR provides high (z dimension) and dense regions. Since the x-y plane is now divided to the discrete cells, Δ_x, Δ_y, we can compute height and density features. We use proportional measures that make them invariant to the average height of a city (e.g. Paris vs. New York City). As a height feature we use

$$P_h(\Delta_x, \Delta_y) = \frac{\underset{z}{\text{argmax}}\ P(\Delta_x, \Delta_y, z)}{\underset{z}{\text{argmax}}\ p(:, :, z)}, \tag{1}$$

and as a density feature

$$P_d(\Delta_x, \Delta_y) = \frac{|P(\Delta_x, \Delta_y, z)|}{\max\limits_{\Delta_x, \Delta_y} |P(\Delta_x, \Delta_y, z)|}. \tag{2}$$

Equations. (1) and (2) are combined to the final "building score":

$$P_{\text{building}}(\Delta_x, \Delta_y) = P_h(\Delta_x, \Delta_y) + \lambda_d P_d(\Delta_x, \Delta_y). \tag{3}$$

In our experiments we used simple maximum heights, but more robust score can be constructed by adopting robust statistics (rank-order statistics) where the maximum value is replaced, e.g. by the value that is higher than 95% of all points. The maximum value performed well in our experiments and we fixed the balancing factor $\lambda_d = 1.0$ for equal weighting for the height and density scores. Moreover, this approach is insensitive to the number of points to compute the score number. In our experiments a notable speed up is attained without significant loss in accuracy, in the case that up to 30% of input points have been randomly removed from computation. The rule-based segmentation of buildings is achieved by computing the building score in (3) to the cells of size 0.25 m × 0.25 m (the same as before) and thresholding each cell by $\tau_{\text{building}} = 1.80$. This generates a binary x-y map (Fig. 3) for which we compute the standard shape compactness features for each connected shape S_i [22]:

$$P(S_i) = \frac{\pi \cdot \text{diameter}^2(S_i)}{4 \cdot \text{area}(S_i)}. \tag{4}$$

Again $P(S_i)$ score is thresholded by $\tau_{P(S_i)} = 15$ and the binary label as {building, ¬building} is backprojected to each 3D point within each cell. Note that this process is executed for a point set from which the street surface points have already been removed $\mathcal{P}_{\text{other}}$ and this rule-based step creates another set $\mathcal{P}'_{\text{other}} = \mathcal{P}_{\text{other}} \backslash \mathcal{P}_{\text{building}} = \mathcal{P} \backslash \left(\mathcal{P}_{\text{road}} \cup \mathcal{P}_{\text{building}}\right)$.

3.2 Boosted decision tree detector for super-voxel features

In our case, the number of 3D points is still large after the rule-based segmentation of roads and buildings and therefore we must consider both performance and efficiency issues for the supervised detection stage. In Fig. 4 is depicted the workflow of our supervised detection that processes the point cloud $\mathcal{P}'_{\text{other}}$. Our approach is inspired by the super-voxel-based processing successfully used in video analysis [41].

3.2.1 Super voxels

The first step is 3D point-wise agglomerative clustering that over-segments the input point cloud to voxels (Fig. 4b). The clustering algorithm incrementally picks a random seed points, adds points to the seed cluster to construct a voxel until no more points pass a distance-based merging rule and then pick a new seed point until all points have been processed. For the random seed point $\mathcal{P}_i$ new points $\boldsymbol{p}_j$ are added $\mathcal{P}_i = \mathcal{P}_i \cup \boldsymbol{p}_j$ if they pass the distance rule,

Fig. 3 Example of rule-based segmentation of building facades. **a** original 3D point cloud and GPS-define x-y plane for projection; **b** x-y projected points; **c** binary x-y map; **d** points that pass the building facade detection step; **e** backprojection of the detected points to 3D

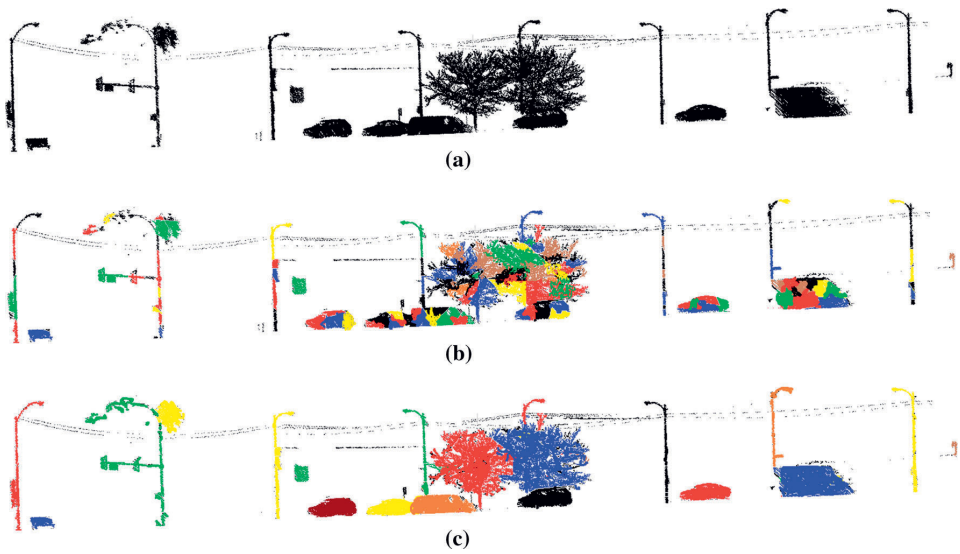


Fig. 4 The workflow of our supervised detection. **a** Input $\mathcal{P}\prime_{other}$ point cloud where road surface and building facade 3D points have been removed; **b** point cloud over-segmentation to 3D voxels by agglomerative clustering; **c** super-voxelization by voxel-level agglomerative clustering

$$\min_i \text{dist}(\boldsymbol{P}_i, \boldsymbol{p}_j) \leq \tau_{\text{voxel}}, \tag{5}$$

where $\text{dist}(\cdot,\cdot)$ is the minimal distance between a set and a point, and the distance threshold is set to $\tau_{\text{voxel}} = 0.005$ m. After the first step, all points have been assigned to a single voxel. The distance threshold avoids setting the number of clusters which highly depends on the size of the point cloud and therefore metric threshold is more intuitive. We refer acute readers to Sect. 5.2 for ablation study concerning the setting of this crucial distance threshold.

The procedure of *super-voxelization* is to merge those voxels that are close to each other and share similar orientation. Formally, the proximity between two voxels $\mathcal{P}_i$ and $\mathcal{P}_j$ is defined as

$$\min_{i,j} \text{dist}(\mathcal{P}_i, \mathcal{P}_j) \leq \tau_{sv\_\text{prox}} \tag{6}$$

which is equivalent to the minimum-link distance rule in agglomerative clustering. The surface orientation is computed using the PCA method in [39] for each voxel and two voxels are combined if their normals are similar

$$\arccos\left(\text{norm}_{PCA}(\mathcal{P}_i), \text{norm}_{PCA}(\mathcal{P}_j)\right) \leq \tau_{sv\_\text{orient}}. \tag{7}$$

We set the super-voxelization thresholds to $\tau_{\text{svoxel1}} = 0.01$ m and $\tau_{\text{svoxel2}} = 15$ which produce high-quality super-voxels on all our datasets (see Fig. 4). The two thresholds with intuitive physical interpretation again avoid setting the number of clusters that would depend on the size of the point cloud.

3.2.2 Super-voxel classification

The automatically generated super-voxels can be classified by computing the popular 3D shape descriptors as features [13,14], but we found these slow to compute and due to variance in point density their robust usage would require re-sampling which is a slow procedure as well. Instead, motivated by success of features with true physical meaning in voxelization and super-voxelization, we adopt several fast-to-compute physical measures as features. The selected features are listed in Table 1.

The features are fed to the boosted decision tree classifier [5] which is extremely efficient and produces high accuracy for multi-class classification tasks. The boosting is based on minimizing the exponential loss:

$$\sum_{i=1}^{M} \exp(-y_i f_\lambda(\boldsymbol{x}_i)) \tag{8}$$

where $\boldsymbol{x}_i$ are the input features and y_i the ground-truth class labels and $f_\lambda(\cdot)$ is the estimated label constructed from

$$f_\lambda(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j h_j(\boldsymbol{x}) \tag{9}$$

where $h_j(\cdot)$ is a weak learner and λ_j its corresponding weight parameter. Selection of the weak learners and optimization of the weights to minimize the loss function can be done efficiently by parallel updating which is faster than the

Table 1 Simple geometric and photometric primitives used to classify super-voxels into pre-defined categories

Feature	Motivation
Geometric features	
Area	Small versus large objects
Edge ratio	Maximum and minimum edges
Max edge	Longest dimension
Covariance	overall shape
Location and orientation features	
Height above road	
Distance to street	Horizontal distance to the car GPS
Normal angle	With respect to the surface orientation
3D and photometric texture features	
Mean intensity	Overall reflection property of the voxel
Density	Density of the points
Planarity	Average distance to the best fitted plane

sequential-update algorithm [5], but we adopted the sequential version due to its simplicity and widespread availability. In the experiments, we used a forest of ten decision trees with each of them having six leaf nodes and this classifier leads to satisfactory classification results for the benchmark datasets used in our work.

4 Applications

The outputs of the two rule-based steps and the supervised detector based step are two large point clouds $\mathcal{P}_{\text{road}}$ and $\mathcal{P}_{\text{building}}$ and a number of smaller point clouds $\mathcal{P}_i$ with assigned labels

$$l_{road}, l_{building}, l_i \in \{road, building, tree, car, pedestrian, \ldots\}\,.$$

Using the point clouds, street view images and the labels we introduce two important applications: (1) enhanced 3D visualization using model-based rendering and (2) 2D semantic segmentation. In the first application we replace the annotated point clouds with 3D graphical models whose parameters are derived from the point cloud properties which provides visually more plausible view to the 3D map data. In the second application we back project the point cloud labels to 2D street view images and demonstrate their usage in 2D semantic segmentation.

4.1 Visualization of 3D urban maps

Our LiDAR point cloud and street view images are registered, i.e. the 3D projective transformations from the street view images to the point clouds are available, due to the common data acquisition by a data collection vehicle. A textured 3D model is typically generated by directly using image values or using parametric models [26]. Image RGB mapping is a fast procedure, but requires mesh generation as the pre-processing step which is time-consuming for large point clouds and is error-prone for noisy datasets. In this section, we introduce our fast rendering-friendly approach that reconstructs 3D urban map model in two stages (Fig. 5). Firstly we use the enhanced ShadVis algorithm [36] to fast render the building facades with high-quality details. The algorithm calculates the illumination of a point cloud with the light coming from a theoretical hemisphere or sphere around the object. In the second step we apply methods to fit pre-designed template models to non-building labelled point clouds.

4.1.1 Building facade rendering

For fast rendering with a high level of details we apply the ShadVis technique in [36]. ShadVis estimates model illuminants as if the light was coming from a theoretical hemisphere or sphere around the object. The graphics hardware rendering pipelines have been designed for polygons, but in our case it is computationally more attractive to render only the points. Therefore, we adopt the simple but effective algorithm in [25]. The accuracy of the result depends on the resolution of the 3D point cloud dataset (see Fig. 6 for a typical case).

4.1.2 Rendering object models

Buildings are large objects with sufficient number of 3D points for high-quality point-wise rendering, but this is not the case for small objects such as cars, trees and pedestrians. However, there are available numerous high-quality 3D models of many visual classes (e.g. 3D Warehouse http://3dwarehouse.sketchup.com) and these can be used to construct more plausible map view. The main problem in using

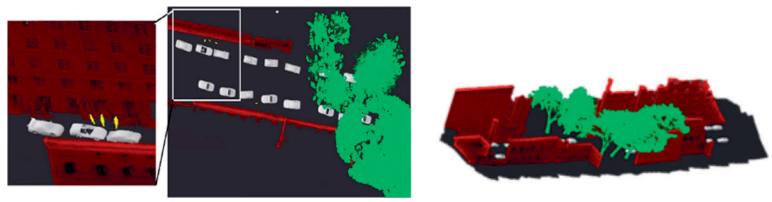


Fig. 5 Example urban 3D map with rendered object models

Fig. 6 Example of 3D points in $\mathcal{P}_{building}$ (approx. 15 m distance, *top*) and results of our point rendering algorithm (*bottom*)

3D models is fitting model to a 3D point cloud. The fitting method is categorized into two types, depending on whether orientation of an object in question plays an important role in rendering (which is critical in our system where better visualization is the goal). The first type of fittings is related to the object classes which their pre-designed mesh structure orientation is not important and their object models will be based on their position and dimension only. The first object type fitting includes trees, pedestrians and sign symbols. Unlike a lot of work which calculate the distance of a given points to the closest surface and use time-consuming iterative procedure to fit the pre-designed model into the point cloud or reconstructed surface [7], we propose a novel approach to solve this problem in a straightforward and computationally lightweight manner. For each separated point cloud $\mathcal{P}_i$, the centre and its boundaries (3D bounding box) will be calculated. Based on the size of the existing pre-designed library meshes, we localize the best isodiametric meshes to the point cloud. Then, as the object orientation is not important we fit the mesh by stretching it to get an appropriate size. This is a similarity transformation of estimated isotropic scale s and transformation $\boldsymbol{t} = (t_x, t_y, t_z)^T$. It is also possible to estimate a similarity transformation where scale is applied to each dimension $\boldsymbol{s} = (s_x, s_y, s_z)^T$.

The second type object requires also x-y orientation angle θ and is needed for different types of vehicles in our data (car, bus, bike). First, the centre of pre-designed mesh is computed and point cloud will be matched and then the corresponding model will be chosen from library based on the dimension of the vehicle 3D bounding box. Then Iterative Closet Point (ICP) algorithm [30] is applied to automatically refine the registration of point clouds with desired mesh. The scene prior knowledge reduces the number of possible vehicle orientations as the road surface is determined (sect. 3.1.1) and only rotations around the z-axis of the road are considered. The ICP algorithm that we apply optimizes the RMS (Root mean Square) distance between closest point pairs of the models vertices to the point cloud [30]

$$err_{\text{RMS}}(\mathcal{P}_{\text{model}}, \mathcal{P}_i) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} ||\boldsymbol{p}_{n,\text{model}} - \boldsymbol{p}_{n,\mathcal{P}_i}||^2} \quad (10)$$

In Fig. 7 is illustrated model fitting to a point cloud. Notice that even with a different target model of the car (sedan vs. hatchback) correct pose is readily estimated.

4.2 Semantic segmentation in 2D

Thanks to the Global Positioning System (GPS) and Inertial Measurement Unit (IMU) measurements in the mobile LiDAR and RGB data acquisition system there is accurate information to register the 3D point cloud and street view (RGB) data (Fig. 8). Therefore, it is straightforward to map the semantic labels of 3D point cloud points to the street view images. For computational efficiency, input images

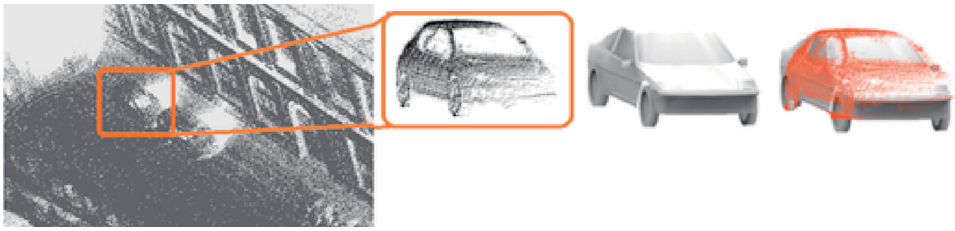


Fig. 7 Example of a 3D car model fitted and rendered to a point cloud

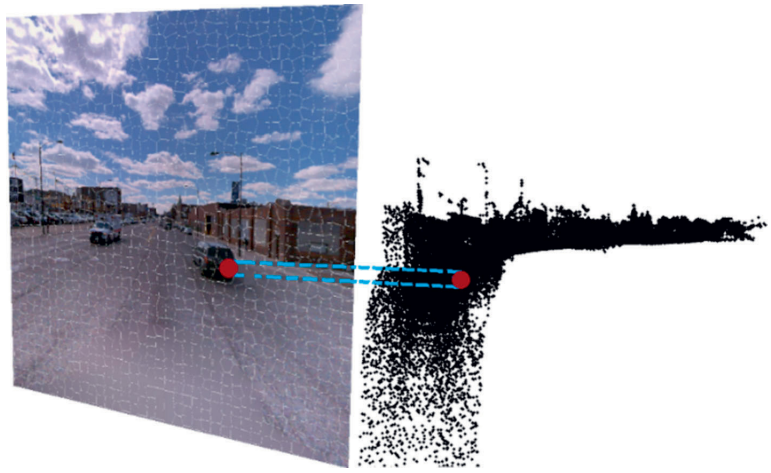


Fig. 8 Mapping between the 3D LiDAR points and 2D street view images

are over-segmented into super-pixels and each image plane super-pixel is associated with a collection of labelled 3D points (Fig. 8). For projection 3D points to image plane we use the generic camera model (images are already rectified to remove the optical distortions) [16]:

$$\boldsymbol{p}_{img} = \boldsymbol{K}\,[\boldsymbol{R}|\boldsymbol{t}]\,\boldsymbol{p}_{3d} \tag{11}$$

where $\boldsymbol{t}$ is a 3×1 translation vector, $\boldsymbol{R}$ is a 3×3 rotation matrix and $\boldsymbol{K}$ is a 3×3 camera matrix. The input and output data are given in the homogeneous coordinate system. All LiDAR points are transformed to each street view image and mapped to the closest super-pixel. The mapping uses z-buffering (within the same pixel only the closest 3D is selected), and majority vote label of 3D points projected to the same super-pixel is selected. The image super-pixels without any label are labelled as "sky".

5 Experiments

In this section, we provide qualitative and quantitative results for the applications of visualization of urban 3D map data and semantic 2D segmentation. We compute the point-wise and pixel-wise classification accuracies and compare our method to various recently proposed methods.

5.1 Datasets

NAVTEQ True the dataset used in this work is described in our previous work [2] and is composed of 500 high-quality street view images of 1032×1032 resolution and corresponding LiDAR point clouds collected from three cities: Chicago, Paris and Helsinki. The data were collected using the NAVTEQ True systems of high-density 360° rotating LiDAR system, 360° panoramic camera and an inertial navigation system (IMU/GPS) for precise position, orientation and attitude tracking of the sensors. Information from all these sensors is synchronized to create an accurate and comprehensive dataset. The LiDAR system has 64 lasers and rotates at 600 rpm covering a full 360° field of view around the car. The LiDAR system scans 3D points at the rate of around 1.2 million points per second. NAVTEQ dataset is acquired in various weather conditions and urban landscapes and represents the most challenging data available at the moment. Seven semantic object classes are defined to label the LiDAR dataset and its corresponding street view images: building, tree, car, traffic sign, pedestrian, road, water (and

sky for unlabelled super-pixels in 2D images). Since the two other datasets do not contain street view images corresponding to LiDAR point clouds we use only this dataset to experiment 2D semantic segmentation.

Paris-Rue-Madame dataset presented in [34] is used to compare our method with other recent works on 3D segmentation and labelling. This dataset is used for urban detection-segmentation-classification methods, consists of accurate annotated 3D point clouds acquired by MLS system on Madame Street in Paris. The division of data to the training and test sets is described in [33,34] and we compare our results to their reported accuracies.

TLS (terrestrial laser scanning) Velodyne dataset [21] includes ten high-quality 3D point cloud scenes collected by a Velodyne LiDAR mounted on a car navigating through the Boston streets. Due to the specific nature of this dataset, we evaluate our method using each LiDAR rotation as a single scene (approximately 70,000 points).

Performance measure Both 3D LiDAR point segmentation and 2D street view segmentation are evaluated point/pixel-wise. We report accuracies for each label and compute other metrics, such as average precision, to compare to the existing works.

5.2 Urban 3D segmentation and classification

Paris-Rue-Madame

The point-wise classification results for our method and for the two recently proposed methods by Aijazi et al. [1] and Serna and Marcotegui [33] are shown in Table 2. Our method achieved an average accuracy of 94.1% with notable margin of 8.5 and 22.2% with respect to existing methods [1] and [33], respectively. Note that even for this relatively easy dataset, the traffic sign class turned out to be particularly challenging due to lack of sufficient training samples. Significant performance deteriorations were observed for all methods: the drop in our method was about 10% while for existing methods 15 and 71%, respectively.

TLS Velodyne

It is notable that our algorithm was initially designed to analyse MLS LiDAR point clouds. One of the main advantages of our method is that it easily adapts to other types of LiDAR datasets such as terrestrial laser scanning (TLS) and airborne laser scanning (ALS) point clouds without major modification as long as the point units are in metric system (thresholds are set in metres). To exemplify this we evaluated our method with the same fixed parameters on the TLS Velodyne LiDAR dataset which contains 3D point clouds in local coordinate system of the LiDAR. The total number of points in each ten scene is nearly 70,000 and the average point density is about 12 points/m^2. We compare our method to Lai and Fox [21]. We selected seven scenes for training and the three remaining scenes for testing similar to them and report per class average precision and F-score computed as

$$\frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}.$$

The results in Table 3 show that for 5 out of 6 classes our method is clearly better and our F-score for each class is better than the average F-score of Lai and Fox.

NAVTEQ True

The NAVTEQ True dataset is our main target - high-quality large-scale ground acquired dataset. NAVTEQ True collected from Boston, Paris and Chicago contains more than 80 million points and covers approximately 2.4 km of road altogether. Seven semantic object classes are defined to label the scenes: building, road, river, car, tree, traffic sign and pedestrian. The point clouds from the three cities are divided into two portions: the training set, and the testing set. The 70% of the total street length is selected for training and 30% for

Table 2 Comparison of our method to other reported results on 3D point cloud classification with the Paris-Rue-Madame dataset

Method	Building	Road	Tr. sign	Car	Class AVE ACCY
Aijazi et al. [1]	0.914	0.901	0.710	0.900	0.856
Serna et al. [33]	0.986	0.940	0.000	0.950	0.719
Our	**0.991**	**0.950**	**0.841**	**0.982**	**0.941**

The bold numbers are related to best results comparing different methods

Table 3 Comparison of our method to other reported results on 3D point cloud classification with the TLS Velodyne dataset

Measure	Method	Tree	Car	Tr. sign	Pedestrian	Fence	Building	Class AVE ACCY
Precision	Lai and Fox [21]	0.83	0.91	**0.80**	0.41	0.61	0.86	0.73
	Our	**0.89**	**0.95**	0.72	**0.88**	**0.85**	**0.95**	**0.87**
F-score	Lai and Fox [21]	0.76	0.79	**0.69**	0.47	0.42	0.91	0.67
	Our	**0.85**	**0.93**	**0.69**	**0.88**	**0.80**	**0.95**	**0.85**

The bold numbers are related to best results comparing different methods

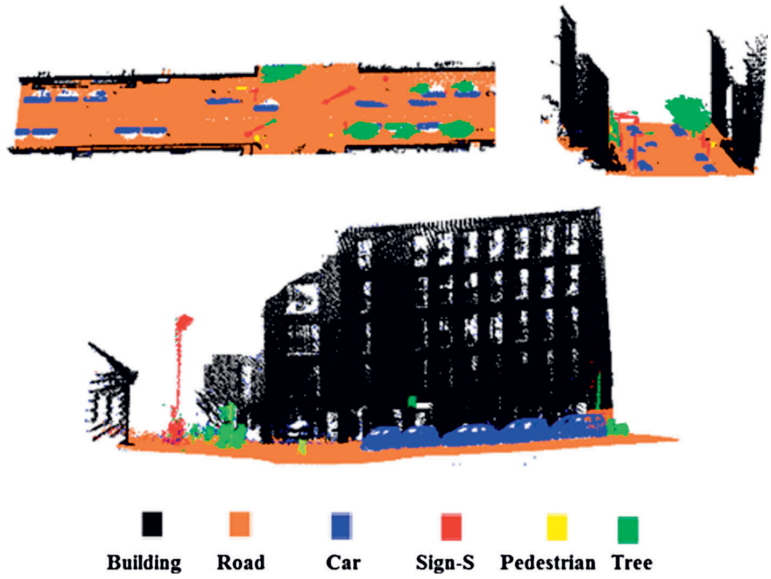


Fig. 9 Segmented and classified 3D LiDAR points of the NAVTEQ True dataset from Helsinki (colours encode the different labels)

Table 4 Confusion matrix of our method for classification of the the NAVTEQ True dataset

	Building	Road	River	Car	Tree	Tr. sign	Pedestrian
Building	0.885	0.083	0.000	0.000	0.115	0.000	0.000
Road	0.041	0.958	0.000	0.003	0.015	0.001	0.000
River	0.000	0.249	0.733	0.000	0.000	0.000	0.000
Car	0.000	0.018	0.000	0.847	0.000	0.007	0.000
Tree	0.004	0.001	0.000	0.000	0.897	0.007	0.000
Tr. sign	0.000	0.002	0.000	0.000	0.113	0.735	0.002
Pedestrian	0.000	0.049	0.000	0.000	0.008	0.000	0.782

Table 5 Computing times of our method with and without the rule-based steps quick for road surface and building detection. Without the rule-based step all points are classified using the super-voxel and boosted decision tree method

Our method	# of voxels	Comp. time (mins)	Overall accuracy
w rule-based	32,891	46	86%
w/o rule-based	246,548	291	75%

testing. Some typical results are illustrated in Fig. 9. Confusion matrix in Table 4 shows that the average accuracy (over all classes) is about 83%, with rule-based classification accuracies above 88%. Relatively low accuracies were reported for certain classes, e.g. pedestrian (78%), traffic sign (73%) and river (73%). These challenging cases are ascribed to the lack of sufficient training samples for each class.

Computing time

The proposed method has various advantages. The main contribution of this work is about achieving high accuracy within reasonable computing time. Considering the large-scale LiDAR datasets, we believe that fully supervised classification methods are computationally too expensive. In this experiment, we switched off the rule-based processing stage, but performed super-voxel-based supervised training and classification in Sect. 3.2. The results are collected to Table 5, and the computing time is wall time on Intel (R) Core(TM) i7-4710MQ 2.5 GHz CPU with 32 GB RAM. The results show that without the rule-based segmentation step the supervised classifier must construct and classify 7.5× more voxels and thus the computation time is 6.3× longer and requires much more memory usage. Moreover, without rule-based processing, the classification accuracy degraded significantly, partially due to the connectedness problem, i.e. roads surfaces and buildings are mis-segmented with other objects. In contrast, the removal of road surfaces and building facades created better isolated point clouds and hence improved classification accuracy from 75 to 86%.

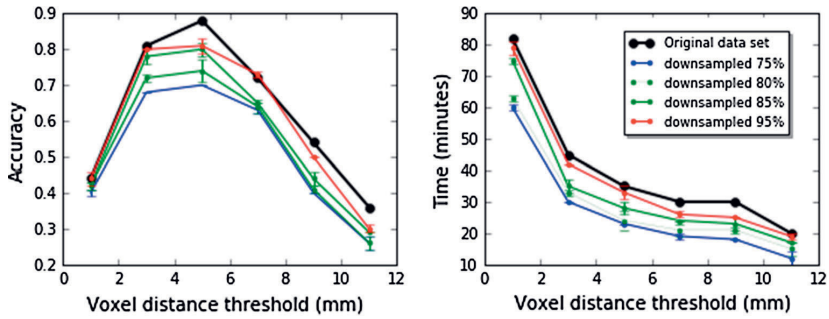


Fig. 10 Super-voxel classification accuracy on NAVTEQ True dataset (*left*) and computing time (*right*) with respect to the distance threshold τ_{voxel} in (5)

Table 6 Confusion matrix of pixel-wise accuracies of our method for 2D semantic classification of the NAVTEQ True street view images

	Sky	Building	Road	Tree	Car	Tr. sign	Pedestrian	River
Sky	0.960	0.020	0.000	0.020	0.000	0.000	0.000	0.000
Building	0.030	0.870	0.024	0.075	0.000	0.000	0.000	0.000
Road	0.000	0.015	0.920	0.000	0.065	0.000	0.000	0.000
Tree	0.000	0.280	0.080	0.640	0.000	0.000	0.000	0.000
Car	0.050	0.000	0.250	0.020	0.680	0.000	0.000	0.000
Tr. sign	0.010	0.280	0.090	0.000	0.000	0.370	0.250	0.000
Pedestrian	0.010	0.340	0.020	0.020	0.000	0.330	0.280	0.000
River	0.000	0.050	0.250	0.050	0.000	0.000	0.000	0.650

Parameter settings

An important parameter controlling our method's accuracy and computing time is the threshold used to generate super-voxels (Sect. 3.2.1). In our experiments this was set to $\tau_{\text{voxel}} = 0.005$ m, but to further study the effect of this parameter we conducted an ablation study with the NAVTEQ True dataset by varying the threshold value. The results of this experiment are shown in Fig. 10 (displayed in black curves) where the setting 5 mm clearly provides high accuracy with reasonable computation time.

The performance is evaluated in both robustness and accuracy terms with four sub-sampled of original point clouds. The testing point clouds are down-sampled uniformly to 75, 80, 85, 95% of the original point cloud density [31]. Refer Fig. 10 (colourful curves) for a summary of the algorithm performance results. Note that the dataset was down-sampled in multiple runs and the average accuracies as well as deviations were plotted in Fig. 10. The results show that the optimal threshold is consistent (around 5 mm) in despite that the average accuracy decreases with the percentage of down-sampling.

5.3 Semantic 2D segmentation

Dense scene labelling/segmentation is an important problem in robot and computer vision [24,29,44] and in our case this can be achieved by backprojecting the labelled 3D points to 2D camera view plane (Sect. 4.2). For evaluation of 2D semantic segmentation we generated 2D ground truth by backprojecting the ground-truth 3D labels to the corresponding street view images in 500 randomly selected images in the NAVTEQ True test set. The back-projection results were manually verified and corrected. Pixel-wise classification accuracies are in Table 6. The sky, building and road regions were accurately labelled in ($\geq$85% accuracy). The traffic signs and pedestrians were more poorly segmented, and this can be explained by the fact that there are not many examples for our classifier and therefore it makes misclassifications to the more frequent classes. However, the pixel-wise accuracies may give wrong interpretation of the results which qualitatively looked good as shown in the illustrative examples in Fig. 11.

6 Discussion

Firstly, rule-based classification is dedicated to the dominant objects, i.e. roads and buildings presented in LiDAR datasets, whereas rules are designed based on prior knowledge of these objects in terms of their sizes, relative positions, etc. A systematic approach to fine-tuning rules is to cross-validate rule parameters with respect to a separate dataset accompanied

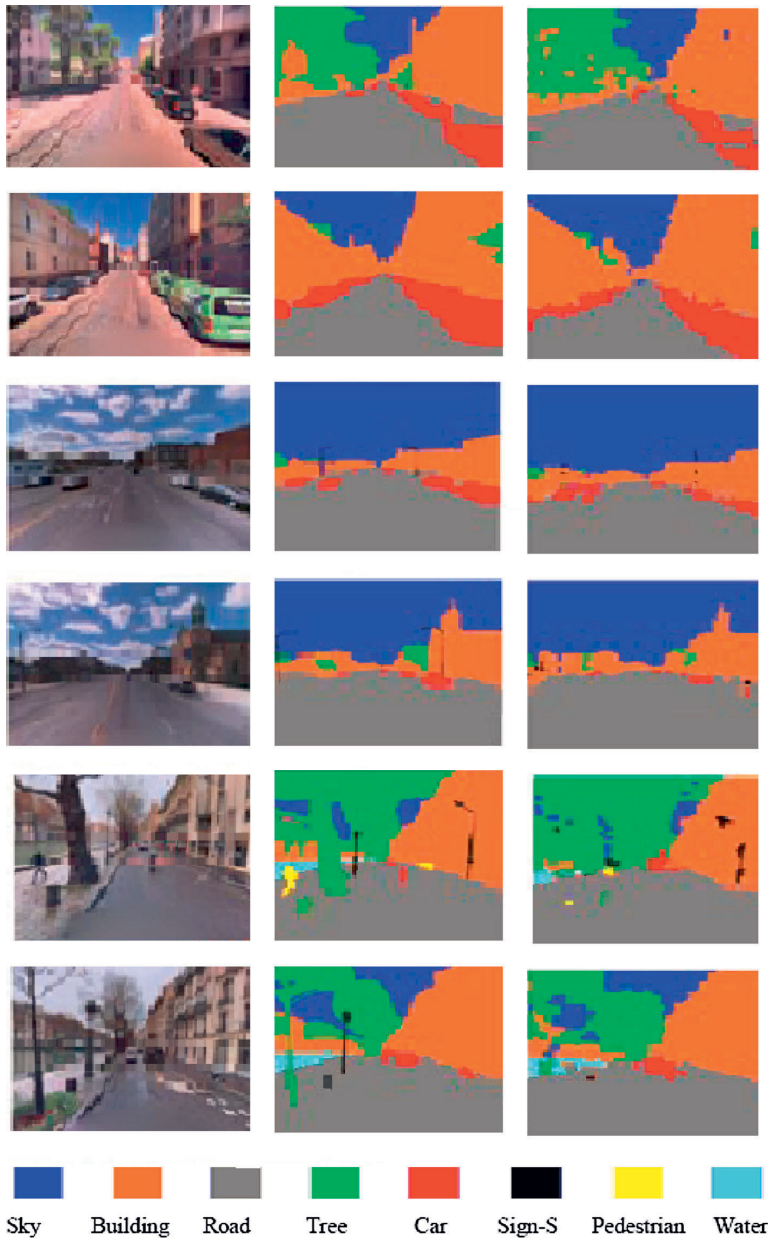


Fig. 11 2D street view image segmentation using 3D label back projection. *Left* test image; *middle* ground truth; *right* our results

with ground-truth labels. Adding new rules can be treated in the similar manner. Nevertheless, a great deal of ground-truth labels is required to pursue this approach, making it only suitable for applications with ample ground-truth data available. Secondly, the street view 3D modelling application is restricted to the diversity and number of the pre-designed mesh templates in library. This problem can be solved by creating a big library of street view objects such as trees and cars to generate more real 3D models.

7 Conclusions

We have proposed an efficient and accurate two-stage method to segment and semantically label urban 3D city maps of reg-

istered LiDAR point clouds and RGB street view images. Our method can process 80 million 3D points (2.4 km street distance) in less than an hour on commodity desktop hardware. The first processing stage uses rule-based detectors for road surfaces and building facades that span more than 75% of city point clouds. The rules are based on robust and adaptive processing (e.g. to the average building height of a specific city) with thresholds that have clear physical meaning and setting them is therefore intuitive. The remaining point cloud is processed by methodology that first constructs voxels (point clusters), and the super-voxels are then classified by an ensemble of boosted decision trees. Voxel construction, super-voxel construction and the extracted features are also based on thresholds and measures with clear physical meaning which allows their intuitive setting for other types of 3D map data. The rule-based stage makes computing 6× faster as compared to classifier-only and improves the segmentation accuracy. Moreover, we proposed two applications of our method: 1) model-based 3D visualization for better user experience and 2) 2D semantic segmentation for 2D applications. Both applications were also experimentally validated and our method performs favourably as compared to other existing methods. Our future work will address adaptation of the method for other 3D map data than urban city centres.

References

1. Aijazi, A., Checchin, P., Trassoudaine, L.: Segmentation based classification of 3D urban point clouds: a super-voxel based approach with evaluation. Remote Sens. **5**(4), 1624–1650 (2013)
2. Babahajiani, P., Fan, L., Kämäräinen, J.K., Gabbouj, M.: Automated super-voxel based features classification of urban environments by integrating 3D point cloud and image content. In: IEEE International Conference on Signal and Image Processing Applications (2015)
3. Babahajiani, P., Fan, L., Kämäräinen, J.K., Gabbouj, M.: Comprehensive automated 3D urban environment modelling using terrestrial laser scanning point cloud. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Large Scale 3D Data (2016)
4. Buch, A., Yang, Y., Krüger, N., Petersen, H.: In search of inliers: 3d correspondence by local and global voting. In: CVPR (2014)
5. Collins, M., Schapire, R.E., Singer, Y.: Logistic regression, AdaBoost and Bregman distances. Mach. Learn. **48**(1), 253–285 (2002)
6. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV Workshop (2004)
7. Dohan, D., Matejek, B., Funkhouser, T.: Learning hierarchical semantic segmentations of LIDAR data. In: International Conference on 3D Vision (3DV) (2015)
8. Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., Frenkel, A.: On the segmentation of 3d lidar point clouds. In: ICRA (2011)
9. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)
10. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
11. Golovinskiy, A., Kim, V., Funkhouser, T.: Shape-based recognition of 3d point clouds in urban environments. In: ICCV (2009)
12. Guo, B., Huang, X., Zhang, F., Sohn, G.: Classification of airborne laser scanning data using JointBoost. Isprs J. Photogramm. Remote sens. **100**, 71–83 (2015)
13. Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J.: 3d object recognition in cluttered scenes with local surface features: a survey. PAMI **36**(11), 2270–2287 (2014)
14. Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., Kwok, N.: A comprehensive performance evaluation of 3D local feature descriptors. Int. J. Comput. Vis. **116**, 66–89 (2016)
15. Guo, Y., Sohel, F., Bennamoun, M., Lu, M., Wan, J.: Rotational projection statistics for 3d local surface description and object recognition. Int. J. Comput. Vis. **105**(1), 63–86 (2013)
16. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge press, Cambridge (2003)
17. Käler, O., Reid, I.: Efficient 3d scene labeling using fields of trees. In: ICCV (2013)
18. Knopp, J., Prasad, M., Willems, G., Timofte, R., van Gool, L.: Hough transform and 3D SURF for robust three dimensional classification. In: ECCV (2010)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)
20. Lafarge, F., Mallet, C.: Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. Int. J. Comput. Vis. **99**(1), 69–85 (2012)
21. Lai, K., Fox, D.: Object recognition in 3d point clouds using web data and domain adaptation. Int. J. Robot. Res. **29**(8), 1019–1037 (2010)
22. Lantuejoul, C., Maisonneuve, F.: Geodesic methods in quantitative image analysis. Pattern Recognit. **17**(2), 177–187 (1984)
23. Li, J., Li, X., Yang, B., Sun, X.: Segmentation-based image copy-move forgery detection scheme. IEEE Trans. Inf. Forensics Secur. **10**(3), 507–518 (2015)
24. Martinovic, A., Knopp, J., Riemenschneider, H., Van Gool, L.: 3D all the way: Semantic segmentation of urban scenes from start to end in 3D. In: CVPR (2015)
25. Max, N.: Logistic regression, AdaBoost and Bregman distances. Mach. Learn. **48**(1–3), 253–285 (2002)
26. Musialski, P., Wonka, P., Aliaga, D., Wimmer, M., van Gool, L., Purgathofer, W.: A survey of urban reconstruction. Comput. Graph. Forum **32**(6), 146–177 (2013). doi:10.1111/cgf.12077
27. Nguyen, A., Le, B.: 3d point cloud segmentation: a survey. In: IEEE Conference on Robotics, Automation and Mecathronics (RAM) (2013)
28. Papazov, C., Burschka, D.: An efficient RANSAC for 3D object recognition in noisy and occluded scenes. In: ACCV (2010)
29. Ren, X., Bo, L., Fox, D.: RGB-(D) scene labeling: features and algorithms. In: CVPR (2012)
30. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: International Conference on 3D Digital Imaging and Video (2001)
31. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai (2011)
32. Sengupta, S., Greveson, E., Shahrokni, A., Torr, P.: Urban 3d semantic modelling using stereo vision. In: ICRA (2013)

33. Serna, A., Marcotegui, B.: Detection, segmentation and classification of 3D urban objects using mathematical morphology and supervised learning. ISPRS J. Photogramm. Remote Sens. **93**, 243–255 (2014)
34. Serna, A., Marcotegui, B., Goulette, F., Deschaud, J.E.: Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In: 3rd International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014 (2014)
35. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV (2003)
36. Tarini, M., Cignoni, P., Scopigno, R.: Visibility based methods and assessment for detail-recovery. In: Proceedings of IEEE Visualization, pp. 60–67 (2003)
37. Theologou, P., Pratikakis, I., Theoharis, T.: A comprehensive overview of methodologies and performance evaluation frameworks in 3d mesh segmentation. Comput. Vis. Image Underst. **135**, 49–82 (2015)
38. Velizhev, A., Shapovalov, R., Schindler, K.: Implicit shape models for object detection in 3d point clouds. In: ISPRS Congress (2012)
39. Xiao, J., Quan, L.: Multiple view semantic segmentation for street view images. In: ICCV (2009)
40. Xie, J., Kiefel, M., Sun, M., Geiger, A.: Semantic instance annotation of street scenes by 3d to 2d label transfer. In: CVPR (2016)
41. Xu, C., Whitt, S., Corso, J. J.: Flattening supervoxel hierarchies by the uniform entropy slice. In: CVPR (2013)
42. Yu, T., Wang, R.: Scene parsing using graph matching on street-view data. Comput. Vis. Image Underst. **145**, 70–80 (2016)
43. Zaharescu, A., Boyer, E., Horaud, R.: Keypoints and local descriptors of scalar functions on 2d manifolds. Int. J. Comput. Vis. **100**, 78–98 (2012)
44. Zhang, R., Candra, S., Vetter, K., Zakhor, A.: Sensor fusion for semantic segmentation of urban scenes. In: CVPR (2015)
45. Zheng, Y., Jeon, B., Xu, D., Wu, Q., Zhang, H.: Image segmentation by generalized hierarchical fuzzy c-means algorithm. J. Intell. Fuzzy Syst. **28**(2), 961–973 (2015)
46. Zhou, Z., Wang, Y., Wu, Q.J., Yang, C.N., Sun, X.: Effective and efficient global context verification for image copy detection. IEEE Trans. Inf. Forensics Secur. **12**(1), 48–63 (2017)

Pouria Babahajiani received his BS degree in Biomedical Engineering, Signal Processing, from Tehran Polytechnic (a.k.a Amirkabir University of Tech), and his MS in Electrical Engineering, Signal Processing from Tampere University of Technology, Tampere, Finland. Currently, he is continuing research in the field of Computer Vision as a PhD candidate in Tampere University of Technology and as an external researcher in Nokia Technologies. His research interests include big data analytics, augmented and virtual reality, multimedia, convolutional neural networks, nonlinear signal and image processing, and 3D data processing and rendering.

Dr Lixin Fan is a principal scientist at Nokia Technologies. His research areas of interests include 3D image and video processing, computer vision, machine learning, big data analysis, intelligent human-computer interface, augmented and virtual reality, mobile ubiquitous and pervasive computing. His recent research activities are related to Nokia/HERE map 3D data processing and rendering. Dr Fan is the (co)author of more than 40 international journal & conference publications. He also (co)invented dozens of granted and pending patents filed in USA, Europe and China. Before joining Nokia in 2004, Dr Fan was affiliated with Xerox Research Center Europe and his research work included the well recognized Bag of Keypoints method for image categorization. Dr Fan received his MSc and PhD in Computer Science from National University of Singapore in 1998 and 2002, respectively.

Dr. Joni-Kristian Kämäräinen is associate professor of signal processing at the Department of Signal Processing, Tampere University of Technology, Finland. He holds MSc and PhD degrees from Lappeenranta University of Technology in 1999 and 2003, respectively. He leads the Computer Vision Group and his research focuses on 2D and 3D scene analysis, object detection and recognition, signal processing and machine intelligence.

Moncef Gabbouj received his BS degree in electrical engineering in 1985 from Oklahoma State University, Stillwater, and his MS and PhD degrees in electrical engineering from Purdue University, West Lafayette, Indiana, in 1986 and 1989, respectively. Dr. Gabbouj is a Professor of Signal Processing at the Department of Signal Processing, Tampere University of Technology, Tampere, Finland. He was Academy of Finland Professor during 2011–2015. He held several visiting professorships at different universities. Dr. Gabbouj is currently the TUT-Site Director of the NSF I/UCRC funded Center for Visual and Decision Informatics. His research interests include Big Data analytics, multimedia content-based analysis, indexing and retrieval, machine

learning, pattern recognition, nonlinear signal and image processing and analysis, voice conversion, and video processing and coding. Dr. Gabbouj is a Fellow of the IEEE and member of the Academia Europaea and the Finnish Academy of Science and Letters. He is the past Chairman of the IEEE CAS TC on DSP and committee member of the IEEE Fourier Award for Signal Processing. He served as Distinguished Lecturer for the IEEE CASS. He served as associate editor and guest editor of many IEEE, and international journals. Dr. Gabbouj was the recipient of the 2015 TUT Foundation Grand Award, the 2012 Nokia Foundation Visiting Professor Award, the 2005 Nokia Foundation Recognition Award, and several Best Paper Awards. He published over 650 publications and supervised 45 doctoral and 58 Master theses.

PUBLICATION V

Depth-Augmented Stereo Panorama from 360 Degree Camera for 6-DOF VR Rendering

P.Babahajiani, S.Husseini, and M.Gabbouj