HAL
open science

Geometric approximation of structured scenes from images

Muxingzi Li

To cite this version:

Muxingzi Li. Geometric approximation of structured scenes from images. Discrete Mathematics [cs.DM]. Université Côte d'Azur, 2021. English. NNT : 2021COAZ4061 . tel-03388295v2

HAL Id: tel-03388295

https://theses.hal.science/tel-03388295v2

Submitted on 5 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Approximation géométrique de scènes structurées à partir d'images

Muxingzi LI

INRIA Sophia-Antipolis Méditerranée

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur
Dirigée par : Florent Lafarge
Soutenue le : 8 octobre 2021

Devant le jury, composé de :

Hui Huang	Professeur	Shenzhen University
Bruno Vallet	Chargé de recherche	IGN
Renaud Marlet	Directeur de recherche	ENPC
Nicolas Mellado	Chargé de recherche	CNRS
Florent Lafarge	Directeur de recherche	INRIA Sophia-Antipolis

Inria

Approximation géométrique de scènes structurées à partir d'images

Geometric approximation of structured scenes from images

Jury:

Rapporteurs
Hui Huang, Professor, Shenzhen University
Bruno Vallet, Chargé de recherche, IGN

Examinateurs
Renaud Marlet, Directeur de recherche, ENPC
Nicolas Mellado, Chargé de recherche, CNRS
Florent Lafarge, Directeur de recherche, INRIA Sophia-Antipolis

Acknowledgements

First, I would like to thank my advisor, Florent Lafarge, for inspiring my thesis. His insightful thoughts and rigorous attitude have shaped my mind as a researcher. I am also grateful for his support that made possible my internship during the pandemic. It is a great pleasure working with you.

Second, I wish to thank my collaborators at ENPC, KAUST and Alibaba for the wonderful time working together. I remember numerous fruitful discussions at the most strange hours in the day, and the camaraderie we formed through fighting the deadlines.

I would also like to express my gratitude towards the committee members, for their time spent on reading and their constructive comments that enrich this thesis.

Many thanks to my fellows at Inria for sharing an amazing experience together. I always remember the coffee breaks, the nights of board games, the view of Inria Sophia Antipolis from the terrace. Special thank Jean-Philippe, Hao and Jean-Dominique for being great seniors who offered help in a lot of things, especially at the beginning of my PhD. Most importantly, I also had a great time outside the office. The trips that we went on together to the Calanques and to Denmark are one of my best memories.

In the end, I would like to thank my family and friends for their constant support throughout my research adventure. In particular, I would thank my parents for always encouraging me to pursue my goals, no matter how far I am away from them.

Résumé

L'approximation géométrique d'objets urbains avec une représentation compacte et précise est un problème difficile en vision par ordinateur et en infographie. La littérature existante se concentre principalement sur la reconstruction à partir de nuages de points de haute qualité obtenus par balayage laser qui sont trop coûteux pour de nombreux scénarios pratiques. Ceci motive l'investigation du problème d'approximation géométrique à partir de données image. La reconstruction dense à partir d'une collection d'images est rendue possible par les progrès récents des techniques de stéréoscopie multi-vues, mais le nuage de points résultant est souvent trop imparfait pour créer un modèle compact. En particulier, nous visons à décrire la scène capturée avec une représentation compacte et précise.

Dans cette thèse, nous proposons deux algorithmes génériques qui abordent différents aspects de l'approximation géométrique basée image. Dans un premier temps, nous présentons un algorithme d'extraction et de vectorisation d'objets dans des images avec des polygones. Dans un second temps, nous présentons un algorithme de recalage global à partir de données géométriques multimodales, typiquement des nuages de points 3D et des maillages surfaciques. Les deux approches exploitent la détection de primitives géométriques pour approcher soit des formes 2D avec des polygones formés à partir de segments de ligne, soit des ensembles de points 3D avec une collection de formes planes. Les algorithmes proposés peuvent être utilisés de manière séquentielle pour former une chaîne de traitement pour l'approximation géométrique d'un objet urbain à partir d'un ensemble de données d'image, composé d'une prise de vue aérienne pour l'extraction de modèles grossiers et de données stéréo multi-vues pour la génération de nuages de points. Nous démontrons la robustesse et l'évolutivité de nos méthodes pour les scènes et objets structurés, ainsi que le potentiel applicatif pour les objets de forme libre.

Mots clés: Partitionnement d'images, extraction d'objets, approximation de forme, minimisation de l'énergies, recalage global, nuage de points, maillage polygonal, primitives géométriques

Abstract

Geometric approximation of urban objects with compact and accurate representation is a challenging problem that concerns both computer vision and computer graphics communities. Existing literature mainly focuses on reconstruction from high-quality point clouds obtained by laser scanning which are too costly for many practical scenarios. This motivates the investigation into the problem of geometric approximation from low-budget image data. Dense reconstruction from a collection of images is made possible by recent advances in multi-view stereo techniques, yet the resulting point cloud is often far from perfect for generating a compact model. In particular, our goal is to describe the captured scene with a compact and accurate representation.

In this thesis, we propose two generic algorithms which address different aspects of image-based geometric approximation. First, we present an algorithm for extracting and vectorizing objects in images with polygons. Second, we present a global registration algorithm from multi-modal geometric data, typically 3D point clouds and surface meshes. Both approaches exploit detection of linear geometric primitives to approximate either 2D silhouettes with polygons consisting of line segments, or 3D point sets with a collection of planar shapes. The proposed algorithms could be used sequentially to form a pipeline for geometric approximation of an urban object from a set of image data, consisting of an overhead shot for coarse model extraction and multi-view stereo data for point cloud generation. We demonstrate the robustness and scalability of our methods for structured scenes and objects, as well as applicative potential for free-form objects.

Keywords: Image partitioning, object contouring, shape approximation, energy minimization, global registration, point cloud, polygon mesh, geometric primitives

Contents

Page

Contents viii

1 Introduction 1
- 1.1 Context and motivations . . . 1
- 1.2 Data types . . . 3
  - 1.2.1 Lidar scans. . . . 3
  - 1.2.2 RGB-D sensors. . . . 6
  - 1.2.3 MVS point clouds/meshes. . . . 7
  - 1.2.4 Overhead images. . . . 8
- 1.3 Problem description . . . 9
- 1.4 Challenges . . . 9
- 1.5 Our contributions . . . 11
  - 1.5.1 Polygonal image segmentation . . . 13
  - 1.5.2 Registration of multi-modal geometric data . . . 13
- 1.6 Outline . . . 15

2 Literature review 17
- 2.1 Generation of compact 3D models . . . 17
  - 2.1.1 Compact polygonal meshes from point clouds . . . 17
    - 2.1.1.1 Approximation methods . . . 17
    - 2.1.1.2 Shape assembling methods . . . 18
  - 2.1.2 Compact model from a single image . . . 20
    - 2.1.2.1 Generation of building models . . . 20
    - 2.1.2.2 Generation of floor layout . . . 22
    - 2.1.2.3 Polygonal image segmentation . . . 23
- 2.2 3D geometry registration . . . 26
  - 2.2.1 Methods with known scale . . . 26
  - 2.2.2 Methods with relative scale estimation . . . 28

3 Polygonal image segmentation 31
- 3.1 Introduction . . . 31
- 3.2 Background on kinetic data structures . . . 32
- 3.3 Algorithm . . . 33
  - 3.3.1 Energy formulation . . . 34
  - 3.3.2 Exploration mechanism . . . 36
- 3.4 Experiments . . . 41
  - 3.4.1 Flexibility and robustness. . . . 42

3.4.2 Ablation study 42
3.4.3 Quantitative evaluation. 42
3.4.4 Performance. 46
3.4.5 Limitations. 48
3.5 Conclusion 48

4 3D registration of multi-modal geometry 51
4.1 Introduction 51
4.2 Background on Lie groups for Sim(3) 52
4.3 Algorithm 54
4.3.1 Planar shape based alignment 55
4.3.2 Refinement 57
4.4 Experiments 59
4.4.1 Dataset and error metrics 59
4.4.2 Robustness 62
4.4.3 Comparisons 62
4.4.4 Performances 64
4.4.5 Limitations 64
4.5 Conclusion 65

5 Application to floor modeling 69
5.1 Principle 69
5.2 Experiments 74
5.3 Limitations 75

6 Conclusion and perspectives 77
6.1 Conclusion 77
6.2 Perspectives 78

A Appendix 85
A.1 Polygonal image segmentation: a pseudo-code for kinetic propagation inside a nested polygon 85
A.2 3D registration of multi-modal geometry: a pseudo-code for clustering surface-normals of planar shapes 87
A.3 3D registration of multi-modal geometry: scale estimation for two-step baselines 89

Bibliography 91

Chapter 1

Introduction

1.1 Context and motivations

Obtaining simple geometric models of structured objects and scenes is a long-standing topic in the fields of computer vision and computer graphics. It has a wide range applications in rendering and simulation where some tasks can be processed more efficiently using simplified representations of the objects whenever exact accuracy is not necessary. It has attracted a significant amount of attention from the research community and researchers have attempted the problem from different perspectives.

The simplified geometric models provide a clean representation of the underlying objects. Unlike a dense mesh which often contains redundant information, a simplified model represents the same shape with a low budget on the number of constituent elements while preserving salient geometric features. Level of Detail (LOD) is concerned with the use of different representations of a geometric object having different levels of abstraction (Figure 1.1). The desired level of accuracy and complexity depends on the application, e.g. physical simulation and virtual reality benefit more from simplified models as they can accelerate the handling of complex models by omitting unessential computation steps, energy-related assessments concerns with the volume and surface area of a building and can achieve sufficiently accurate results with an LOD2 model [GP12]. Typically, a simplified model can be stored as a mesh, consisting of a set of vertices, edges and faces, or a CAD model, defined as an assemblage of parametric geometric shapes such as curves, surface primitives and volumes. These models exhibit interesting properties:

- Compactness. Simplified models are compact in terms of the number of geometric elements. The proper level of compactness is scene-dependent, e.g. man-made objects are usually more suitable for a compact representation while free-form shapes often need more elements

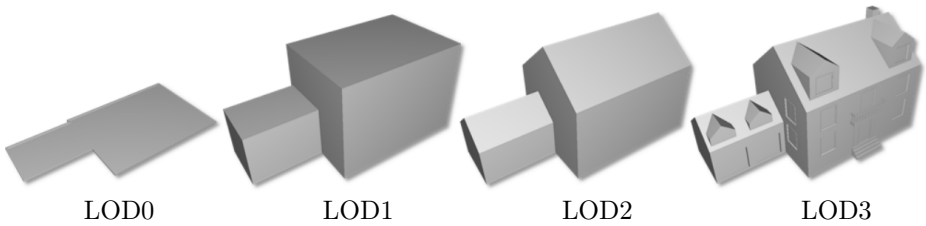


Figure 1.1: Four LODs of a building used by CityGML. Each LOD is suitable for different purpose. Image taken from [VLA15].

to be accurately described. Compactness is crucial for reducing memory consumption and improving computational efficiency for practical applications.

- Editability. Simplified models retain connectivity and topological constraints between constituent elements. This property makes it easy for the users to make modifications to the models, e.g. reshaping a part of the model, and adding texture.

Simplified models have found applications in both scientific and industrial fields. For instance, rendering 3D models at sufficient frame rates is crucial in many practical situations where direct manipulation of the objects and interactive camera control are required. Simple scenes in 3D video-games may only have a few hundred textured polygons, which is often not the case for raw scientific data that involve scenes of widely varying complexity, sometimes up to millions of faces. The complexity and characteristics of the model is a key factor for the rendering cost. Simulating the propagation of acoustic or electromagnetic waves in urban areas also requires modeling of the surrounding environment. The high cost of simulation poses specific constraints on the accuracy and complexity of the 3D models in order to achieve a balance between the computational resources and the accuracy of results. 3D urban modeling from remote sensing data aims at parsing urban scenes consisting of buildings, roads and vegetation at a large scale. In order to efficiently store and visualize objects over large areas on the Earth's surface, the models need to have low complexity (often saved in different LODs) and the requirement on accuracy can be relaxed.

Depending on the types of input data, there exist various approaches for geometric approximation of objects. Creation of compact models from

a dense mesh or a point cloud can be done by experienced experts using interactive softwares, yet it requires tedious operations and domain-specific knowledge depending on the types of objects and scenes to be considered. For large-scale applications, interactive techniques are not adapted to process large amounts of data due to limitations on time and human labor. The majority of existing works on automatic geometric approximation focus on direct simplification of meshes or reconstruction from point clouds. Despite the prevailance of using a single mesh or a point cloud as input, the usage of extra data can sometimes provide auxiliary information and is therefore worth exploration. For instance, a family of approaches that is less discussed yet achieves the same goal is fitting a predefined template model to the observed scene, often represented as a point cloud. This typically propose a two-step pipeline consisting of selection of a candidate template from the predefined collection of objects, followed by alignment of the template to the real scene. Image data can also be utilized for several purposes. Semantic information retrieved from the image which can assist the selection of suitable parameters. In some specific applications like building extraction, it is even feasible to generate a coarse model from a single aerial shot of the building [QZF21].

Motivated by the success of previous works, this thesis experiments on image-based geometric approximation and also attempts to link models generated from different data sources.

1.2 Data types

Various data types can be considered for generating 3D models of urban objects or scenes. In this section, we discuss two families of data types: 3D point clouds and images. Table 1.1 summarizes characteristics of each data types.

1.2.1 Lidar scans.

First introduced in 1961, Lidar [Cra07], an acronym of "light detection and ranging", is an active remote sensing method for sensing distances by measuring the time of flight of a laser. At close range, Lidar is commonly used for navigation and control in robotics and autonomous driving systems, for supporting augmented reality applications on mobile devices, as well as for en-

(a) Laser [KPZK17] (b) RGB-D [CDF+17] (c) MVS [NW17]

Figure 1.2: Sources of point cloud data.

(a) (b)

Figure 1.3: Sources of overhead images. (a) An orthophoto of a house. Image taken from dronegenuity.com. (b) A floor plan image. Image taken from roomsketcher.com.

hancing imaging quality of cameras by improving focus accuracy and speed. At long range, it is used for creating high-resolution maps in land surveying, and for profiling clouds and quantifying various atmospheric components in meteorologic applications.

A Lidar instrument consists of a laser emitter that generates beams of light for illuminating the object of interest, a scanner that directs the beams to scan the target area, and a photodetector that receives the reflected light pattern and converts it to an electric impulse. In a pulse-based system, the laser is emitted in the form of short pulses and the device measures the time between the moment of emission and the moment the pulse is returned to the sensor. In a phase-based system, the distances is measured via interferometry, i.e. by using the phase of a modulated laser beam to calculate the distance as a fraction of the signal's wavelength. This type of system is more precise but less energy efficient, and is mainly used for short-range scanning such as the indoor scenario.

	Noise	Outliers	Range	Uniformity	Reconstruction LOD
Lidar point cloud	Low	Low	High	High	High
RGB-D point cloud	High	High	Low	Low	Median
MVS point cloud	High	High	Median	Low	Median
Overhead image	Low	Low	High	Median	Low

Table 1.1: Qualitative assessment of characteristics of different types of data measurements.

Different types of Lidar systems differ in three aspects: wavelength, size of footprint, and positional alignment. Wavelengths of Lidar instruments vary to suit the absorption and backscattering properties of the target material: from about 10 micrometers to approximately 250 nanometers. For instance, topographic Lidar surveys the land typically using near-infrared light, while bathymetric Lidar opts for water-penetrating green light to measure seafloor and riverbed elevations. The footprint of Lidar is modeled as a Gaussian distribution on the horizontal plane. Small footprint Lidar, typically at the scale of centimeters, is more commonly used today for airborne and terrestrial Lidar due to its accuracy. The lateral resolution of Lidar data is determined by the footprint spacing, which is the nominal distance between the centers of consecutive beams along and between the scanning lines, as well as the beam divergence, which refers to the increase in beam diameter as the distance between the laser source and the target lateral plane increases. Depending on its positional alignment, Lidar scanners can be categorized into two families.

Terrestrial Lidar. A stationary terrestrial Lidar sits on a tripod and scans the hemisphere. It is particularly suitable for scanning buildings and indoor scenes. A laser scanner can also be mounted on a ground-based moving platform, like a vehicle for instance, or even on mobile devices. Due to occlusions and limitations on the scanning range, absence of points in some areas is one of the typical issues in the data acquisition process. For terrestrial Lidar scans, the sampling density is usually from 100 up to thousands of points per squared meter, with a precision at the scale of millimeters.

Airborne Lidar. An airborne Lidar can be mounted on airplanes or drones. It is particularly suitable for surveying large-scale scenes like cities and landscapes. Since the airborne Lidar scans the scene from a top-down view with a

typical scanning angle no more than 15 degrees, the point density on sloping surfaces is much lower than on the horizontal planes, e.g. sparse or missing points on the sides of a building. The density of points clouds acquired in airborne Lidar scans is lower than those in terrestrial scans, as its spatial resolution varies between 1 and 50 points per square meter. The precision also decreases at the scale of decimeters.

1.2.2 RGB-D sensors.

Affordable consumer RGB-D cameras emerged in 2010, featuring Microsoft Kinect and Asus Xtion PRO sensors. The basic principle of this per-pixel depth sensing technology is as follows: An infrared light pattern is first projected onto the target scene. The deformed pattern is captured by an infrared camera, and compared part-by-part to reference patterns which are previously recorded at known depths and stored in the device. The per-pixel depth is then determined based on the matches between the projected pattern and the reference patterns. Finally, the depth data recorded by the infrared sensor is correlated to an RGB camera to yield an RGB-D image. Additionally, approximated surface normals can also be stored at each point in the image.

Images recorded by consumer RGB-D cameras usually comprises more than one million pixels. The accuracy of the depth data deteriorates as the distance between the sensor and the objects in the scene increases. For instance, the depth resolution of Microsoft Kinect varies from 0.2 cm to 7 cm for a depth range from 0.5 m to 5 m [KE12].

RGB-D sensors are widely applicable to many scenarios. On a large scale, RGB-D sensors can be integrated into robotic systems for the mapping of unknown environments, formally known as simultaneous localization and mapping (SLAM), which is of great importance for building efficient indoor navigation systems. RGB-D streams can also be used on a much smaller scale to create more detailed reconstructions of objects in smaller environments. The most well-known reconstruction framework is Kinect-Fusion [IKH+11], which employs a volumetric representation of data, where each voxel stores the running average of its distance to an assumed object. A typical reconstruction pipeline from RGB-D images involve a loop consisting of: conversion from depth maps to point clouds, estimation of camera poses

with respect to the so-far reconstructed model, mapping of 3D points into the global coordinate system.

Reconstruction from properly acquired RGB-D data nowadays yields 3D models with a good level of detail, with an average reconstruction error of a few millimeters. However the meshes generated by these techniques are generally complex and require post-processing.

1.2.3 MVS point clouds/meshes.

Reconstruction of a 3D plausible geometry of an object from images alone is a classic problem of great interest for the computer vision community. The problem is formulated as the estimation of the most likely 3D shape that explains the input images.

A generic MVS pipeline consists of two stages. Over the first phase, the camera intrinsic and extrinsic parameters are recovered for each image using a Structure-from-Motion (SfM) algorithm which relies on matching 2D features across images. The camera parameters are often refined by a bundle adjustment step which minimizes camera reprojection error. The second phase is the reconstruction of the 3D geometry from the image and the corresponding camera parameters using a set of consistency metrics. The output of the MVS pipeline is usually a dense point cloud (and the camera parameters of each input image as output by the SfM step). It can optionally output a mesh surface by combining with a post-processing surface reconstruction step.

Image acquisition is the first critical step for MVS reconstruction. General guidelines for successful image acquisition involves: accuracy of the camera model whose reprojection error should be ideally sub-pixel, image quality concerning high-resolution, noise-free and well-focused capture with stable illumination, image overlap which requires each 3D point to be observed by at least three images for robustness, image baseline in the range of 5-15 degrees from a 3D point to input camera locations.

Based on the scale of the scene, MVS data capture setups can be classified into three categories: small-scale indoor scene capture, small-scale outdoor scene capture, and large-scale scene capture using fleets or crowd-sourcing, e.g. cars, drones, and internet images. Generation of point clouds from community photo collections is out of the scope of this thesis. It typically concerns images taken at different times by more than one camera devices

with different intrinsic parameters, and often involves an extra view selection step to identify good groups of images on which to run dense reconstruction. Specifically, the thesis concerns 3D points reconstructed from a collection of images taken from a single calibrated camera.

1.2.4 Overhead images.

Recent years have witnessed the increase in the amount of attention in the computer vision community towards acquiring geometric information from a single image. Different types of input images have been considered for this relatively new task.

Aerial photography. Aerial photography is the taking of photographs from flying objects for acquiring data of urban scenes for geometric modeling. It consists in mounting a digital camera onto a drone, an aircraft or a balloon that goes over an area following a predefined flight map, and taking images satisfying certain overlapping constraints. The images taken can be further processed by photogrammetry softwares for the generation of 2D or 3D digital models. Oblique aerial photography, taken at an angle relative to the Earth's surface, is useful for power line inspection and surveillance. Vertical aerial photography, taken from a straight down angle, is commonly used for topographic maps and land-use planning. Vertical images are often used to create orthophotos, which are simulations of images taken from an infinite distance without perspective. Orthophotos, having been geometrically corrected so as to be usable as a map, have applications in geographic information systems (GIS), and are useful for extracting building footprints and road networks. This thesis concerns orthophotos as a way of collecting overhead images of the scene.

Floor plan images. A floor plan is a measured drawing to scale of the layout of a floor in a building, showing relationships between rooms, spaces, walls, and other physical features. The orientation of the view is downward from above, similar to a map. A plan is drawn as a orthogonally projected plane cut at the typical four foot height above the floor level. The purpose of floor plans is to depict 3D layouts in a 2D manner, and it is possible to convert a 2D floor plan back to a 3D model with various processing tools.

1.3 Problem description

This thesis addresses the problem of relating a set of images to a coarse model for the building exterior and interior. We consider input composed of one overhead image of the target scene, which allows extraction of a coarse model of the object via contour approximation, and a series of Multi-View Stereo (MVS) images, which are used for generating an MVS point cloud representation. In the context of building exterior, the overhead image can come from an aerial photo containing the target building (Figure 1.3 (a)). In the context of indoor scenes, the floor plan image of the site (Figure 1.3 (b)) is considered as the overhead image. The output is a coarse model of the target object with a set of images registered to its coordinate system. The algorithms are designed with the following objectives:

- Fidelity. The output should conform to the observed data. Specifically, the coarse model should be a reasonable approximation of the underlying object, and the registration should recover the ground truth pose of the object.

- Simplicity. The coarse models should contain as small number of geometric elements as possible. The desired level of simplicity is dependent on the scene types, as well as the fidelity requirement.

- Automation. The algorithms should achieve a high level of automation and only take limited user input, i.e. ideally a few user-specified parameters. This is a necessary condition for many practical applications with large amounts of data involved.

- Applicability. Although the overall pipeline is proposed for the context of urban objects due to the fact that a 2.5D representation does not exist for all objects, individual components should be applicable to other types of objects and scenes.

1.4 Challenges

The geometric approximation of structured objects and scenes is a difficult problem that poses scientific challenges which we explore below:

Low-cost devices. Expensive laser scanners are not widely available for many applications constrained with a low-budget constraint. One of the challenges is to develop algorithms that are suitable for low-cost data such as sparse and noisy MVS point clouds reconstructed from image sequences instead of high-quality scanned point cloud. Unlike dense and accurate point clouds produced by high-cost laser scanners, generating a compact geometric model from MVS point clouds without additional prior information is often problematic due to several problems, e.g. low point density, non-uniform sampling, absence of data points in textureless regions, missing data due to occlusions, and the presence of noise and outliers. This suggests the introduction of additional information on the object which can come from alternative image data sources.

Data defects. Images captured from the real world are often corrupted by defects. The most typical issue is noise, which is a direct consequence of the physical acquisition process. Blur is another important factor that significantly reduces the image quality, resulting in ambiguity on the position of object boundaries, which impact the accuracy of segmentation of the object from the background. Presence of unwanted objects such as vegetation, pedestrians or cars in the case of urban scenes are also frequent. The lighting condition is another factor that significantly affects the image quality, as shadows and overexposure may result in loss of information and hinder the processing of the affected regions of data. These defects in images directly affect both coarse model extraction and point cloud generation. Finally, imperfections in the point cloud also pose specific problems for developing robust applications for registration of this type of data.

Heterogeneous data sources. Registering MVS images to a coarse model involves processing of data gathered from different sources, one being point sets generated from images and the other being a clean yet much simplified model. The differences in characteristics such as data density, accuracy and levels of details constitute a challenge for processing such data, meaning that different treatment is necessary for heterogeneous data. Ambiguity of scales may exist for geometric data acquired from different modalities. For example, while laser-scanned point clouds come with a innate absolute notion of distance, MVS point clouds represent the scenes up to an unknown scale. The coarse model estimated from the overhead image also lack the notion

of its absolute size. This naturally gives rise to the requirement for relative scale estimation in order to register data from different sources.

Level of automation. The ultimate goal of geometric approximation is to provide efficient and fully automatic algorithms. Given the capacities of the start-of-the-art of automatic techniques, interactive techniques is typically considered for some specific fields, like in architecture or in the entertainment industry where the expected quality of the models is still beyond the reach. Interactive techniques are incapable of processing of large amounts of data, such as in remote sensing where the industry and academia aim at parsing objects at the scale of an entire city or even the entire Earth's surface. For the purpose of practical applications at a large scale where the quest for accuracy is typically relaxed, the algorithms are required to reach ideally full automation and adapt automatically to various types of objects and scenes.

Scalability. Scalability of the algorithms is also concerned. Existing methods typically perform well on small scenes or simple objects. The styles of buildings and indoor scenes may vary greatly from one location to another. Even within a small scene, very different objects can be found in terms of shapes and appearances. One of the challenges is to design methods which are not limited to only certain types of scenes. Some geometric approximation techniques for urban environments rely on strong geometric assumptions on for example the regularity of the objects, and the parallel/orthogonal relations present in the objects such as the Manhattan World assumption, but these approaches have difficulty generalizing to scenes with different characteristics. On the other hand, assumption-free approaches are more flexible, but the results tend to be less structured. The requirement for the applicability to urban scenes with different styles and levels of complexity indicates exploration of more general characteristics of the objects, aiming to achieve a good balance between the strength of assumptions and the diversity of scenes.

1.5 Our contributions

The traditional geometric modeling pipeline in the context of urban scenes, typically consisting of the acquisition of dense point clouds and the recon-

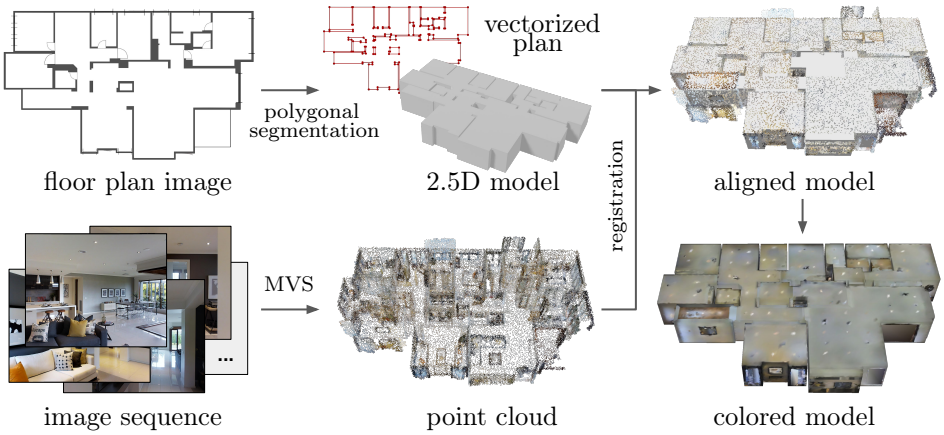


Figure 1.4: Overview of our proposed pipeline for generating a compact model of the building interior. On the top, the input floor plan image is first converted to a vectorized form. The 2.5D model can be generated by extruding the floor with an estimated height. On the bottom part, a point cloud is generated from an input image sequence of the environment via SfM followed by MVS reconstruction. Finally, the model is registered to the point cloud to output an aligned model. Optionally, texture can be added to the model by projecting color from the registered raster images, or from the colored point cloud.

struction of 3D models from such point clouds, is not effective for addressing our problem for the following reasons: (1) Point clouds generated from MVS images are often sparse, and corrupted by noise and other defects like outliers and absence of points in some surfaces due to lack of texture, which makes it difficult for surface reconstruction methods to produce satisfactory results. (2) The reconstructed 3D models, often with high complexity, are not necessarily at the desired LOD for the application purpose, e.g. low-LOD representation is sometimes useful for representing large-scale city scenes [VLA15, BLS16].

In this thesis, we present two generic and scalable algorithms for polygonization of objects of interest in images, and 3D registration of multi-modal geometric data. When combined together, they form a pipeline for geometric approximation of urban objects (e.g. floors and buildings) from a set of images consisting of an overhead image and a sequence of multi-view images. In order to address the aforementioned challenges of traditional

reconstruction-and-simplification pipelines, our pipeline directly estimates a low LOD model from the auxiliary overhead image of the target scene. The MVS image sequence is registered to the model via a point-cloud-to-model registration step. Our key contributions are summarized below:

1.5.1 Polygonal image segmentation

Our first contribution is an algorithm for capturing objects of interest in images by compact floating polygons, as detailed in chapter 3. From a general point of view, this algorithm provides a way to approximate 2D free-form shapes with a compact representation as opposed to traditional pixel-wise segmentation. In the context of urban environments, it is possible to generate coarse models for some objects from a single input image, e.g. an aerial shot of a residence house (Figure 1.3 (a)), and a photo of the floor plan of a building (Figure 1.3 (b)).

We design a novel iterative approach which optimizes the configuration of a polygonal partition in a discrete fashion. First, we design an energy function to measure the quality of a polygonal partition by taking into account both the fidelity to input data (image and semantic information)and the complexity of the output polygons. Second, we propose an efficient optimization scheme to minimize that energy. We explore the solution space by splitting and merging cells within the polygonal partition. The mechanism is controlled by a priority queue that sorts the operations that are most likely to decrease the energy.

Our algorithm features another key ingredient, the kinetic data structure, that enables the proposal of candidate polygonal partitions. We propose to exploit the kinetic data structure for the splitting of an existing polygon into sub-polygons, given a set of detected line segment as the geometric primitives that propagate across time. Unlike previous works that define the boundary to be a regular shape, in our work the primitives are allowed propagate within an arbitrary region enclosed represented by nested polygons.

1.5.2 Registration of multi-modal geometric data

Our second contribution is an algorithm for 3D registration of multi-modal geometric data, i.e. point clouds or meshes obtained from different sources, as detailed in chapter 4. Multi-modal registration is an increasingly common issue when working with 3D objects, e.g. registering a low-resolution point

set to a high-resolution mesh with large variations in detail, scale and coverage. Bringing such data into the same coordinate system is essential before visualizing, comparing and archiving them. Our algorithm consists in, first, a rotational alignment that analyses the surface-normal distributions of the planar shapes detected from the input, and then a local refinement based on continuous optimization.

We propose a novel way of utilizing planar primitives for 3D registration. The motivation lies in three aspects. First, planar shape detection methods offer robustness to noise, outliers and varying sampling density, as opposed to directly working with raw point clouds. Second, it gives a natural approximation of the distance field of the underlying surface of the point cloud. Third, the surface-normal representation is invariant to scaling and translation, which enables the estimation of the initial rotation matrix in a decoupled fashion.

In contrary to previous works, We formulate scale estimation as a part of the continuous optimization problem, without the need of an accurate initial guess for the relative scale between the source and the target surfaces. Our non-feature-based approach is robust to variations of levels of details, noise and point density across different types of inputs, and is suitable for processing large point clouds.

The algorithm is not limited to just urban objects. It is capable of registering generic point cloud to a predefined model, or to another point cloud. A typical application of our algorithm is the localization of cameras to an existing compact 3D model in a multi-view stereo setup, which follows as a direct consequence from the registration of the associated MVS point cloud to the model. In particular, this is useful for transferring pixel or point attributes (color, label etc.) to the model.

In addition to our two contributions, we propose a pipeline for generating a coarse floor model from input consisting of a floor plan image and a multi-view image sequence of the building interior, as outlined in Figure 1.4. Starting from an overhead image of the floor, e.g. a floor plan image, the first step is to extract the object of interest, i.e. the wall structure in a floor plan, in terms of a polygonal representation. Provided with a height, the extracted 2D polygonal representation can be lifted into a 2.5D coarse model. In the end, the coarse model can be registered to the MVS point cloud generated

from the remaining set of captured images. The aligned coarse model can be converted into a textured model by projecting color from raster images onto the model, or even simpler, by taking color from closest points in the point cloud.

In chapter 5, we demonstrate the efficacy of the pipeline on real-world data from existing datasets, providing visualization of the output mesh. The proposed pipeline can also be extended to building model generation when used in conjunction with an algorithm for roof model generation. To this end, the input should consist of an orthophoto of the building and images of the building exterior.

1.6 Outline

The structure of this thesis is organized as follows:

Chapter 2 covers the related works of these problems.

In chapters 3 and 4, we present our algorithms for polygonal image segmentation and multi-modal 3D registration.

In chapter 5, we show the applicative potential of the proposed methods by applying it to the vectorization of floor plans and the generation of coarse floor models from input floor plans and MVS point clouds. In chapter 6, the conclusion of our works is drawn.

Chapter 2

Literature review

In this chapter, we review the literature related to two aspects of our proposed pipeline: generation of compact 3D models, and 3D geometry registration.

In this thesis, we do not bring contribution to the problem of point cloud reconstruction from multi-view images. Instead, we assume as input an MVS point cloud of the target scene, with estimated camera poses associated to the point cloud. We identify difficulties in directly processing a reconstructed point cloud into a coarse mesh for buildings and indoor scenes, and propose to fit to the point cloud a pre-built coarse template from alternative sources. The strategy of using a pre-built template is suited for large-scale reconstruction where a shape may appear repeatedly in the whole scene, e.g. multiple residential buildings sharing the same geometry in the neighborhood.

2.1 Generation of compact 3D models

Compact polygonal mesh representation for 3D scenes can be generated from point clouds or images.

2.1.1 Compact polygonal meshes from point clouds

We summarize the methods for converting a point cloud into a compact polygonal mesh in two categories.

2.1.1.1 Approximation methods

The most straightforward strategy is a two-step approach: a reconstructing step followed by a simplification step. The reconstruction step estimates a smooth surface from the point cloud using existing reconstruction algorithms such as Poisson surface reconstruction [KBH06]. The second step simplifies the dense triangle mesh into a coarser mesh. A first way

for mesh decimation is the edge collapse method, which consists of iteratively replacing an edge with a single vertex based on various cost strategies [DEGN98, GH97b, LT98], removing 2 triangles per operation. Several costs has been proposed for better approximating piece-wise planar structures during the edge collapse process [SLA15, CSaLM13]. These methods have two limitations: First, the decimation approach is unlikely to produce exactly coplanar facets on planar parts of the object. Second, despite its effectiveness on clean input that are both geometrically and topologically accurate, it often fail to deliver faithful results on real-world data corrupted with defects.

Another class of work reduces the search space of the mesh generation problem by imposing geometric assumptions on the output shapes. For instance, the Manhattan-World assumption [CY01] restricts output facets to only three orthogonal directions, therefore enforces the generation of polycubes [FCSS09a, HJS+14, IYF15] which is suited to simple architectural structures. More relaxed assumptions such as parallelism, orthogonality, and z-vertical are also explored for urban scenes [VLA15, FPH21]. However, such geometric assumptions are only relevant for specific scenes and lack generalization ability to diverse domains.

2.1.1.2 Shape assembling methods

Another line of work for approximating shapes with compact meshes consists in extracting a collection of planar shapes from the input point cloud, and assembling them into a final polygonal mesh.

Planar shape detection. The representation of 3D data as a set of basic shapes brings simplicity to large-scale data and robustness to commonly seen defects in real-world data such as noise, outliers or nonuniform sampling density. The intermediate level of abstraction in the form of 3D line-segments, planes, cylinders and cuboids has shown effectiveness in many geometry-related vision applications such as scene reconstruction or recognition. In particular, we focus on recent planar shape detection methods for unstructured point clouds.

The RANSAC paradigm is a conceptually simple technique for plane detection. The RANSAC-based plane detection algorithms [FB81, RL93] extract planes by randomly proposing plane hypotheses on minimal sets and testing against all data points to determine the inlier sets. After a given

number of trials, the plane hypothesis with the most inlier points is added to the output set. These methods are non-deterministic, and its high computational cost requires the design of optimized algorithms. Improvements have been proposed for improving its robustness [TZ00], adapting to local geometric complexity [GLCV19], and reducing its computational cost to achieve real-time performance requirements [Nis03, SWK07a, SWWK08].

Region growing [MLM01, RVDHV06, OLA16] is an alternative approach which consists in continuously propagating a local hypothesis from an initial seed point to fit neighboring points and testing its validity according to a threshold on the minimal number of inlier points covered. The regularities of extracted shapes can be improved by considering parallel and orthogonal relationships between clusters of planes during the detection process [OLA16], or by post-processing optimization [BL20, VLA15]. Different levels of detail can be reached by tuning the parameters of the extraction process [FLD18]. The region growing methods are by design slower than RANSAC, but offers improved robustness of the shape detection process.

Learning-based approaches have also been introduced for shape detection from point clouds. One strategy consists in learning local shape properties in a supervised fashion and extract planes accordingly [GKOM18]. 3D-PRNN [ZYY+17] is a recurrent neural network which generates a set of piece-wise planar primitives to the input depth map. The supervised learning technique can also be applied to fitting different types of geometric primitives such as cylinders [LSD+19].

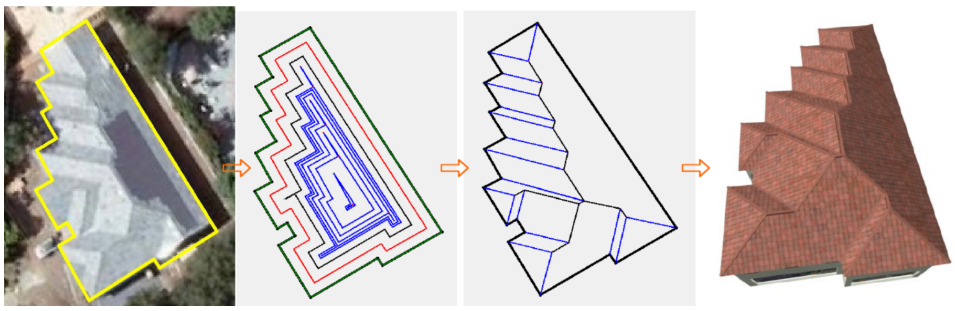
Polygonal mesh generation. Connectivity-based methods extract vertices, edges and facets of the output mesh through analyzing the adjacency relation between the detected planar shapes [CC08, SWF11, VKVLV11, ASF+13, FL20]. Although these methods are fast, they suffer from a lack of robustness to defect-laden data, in particular to over- and under-detection of primitives or erroneous adjacency graphs.

Space partitioning is another technique for surface reconstruction. Several works are based on a 3D Delaunay triangulation from the point cloud [VKVLV13, LA13]. These methods require a dense point cloud and are prone to noise. Another class of approaches utilize space partitions by propagating planar primitives. Slicing techniques [BDLGM14, OLA14, MMP16, NW17] decompose a bounding volume into polyhedral cells by the supporting planes

of detected planar shapes, or decompose a bounding area into polygonal facets by the projected supporting planes in a 2D space [FLPH21]. The surface is extracted by assigning inside or outside labels to the cells, which is a discrete optimization problem with high algorithmic complexity. The large number of polyhedrons can be significantly reduced by introducing a kinetic data structure into the space partitioning step [BL20]. BSP-Net [CTZ20] is a neural network approach which integrates the BSP tree data structure. The network groups the half-spaces to create a collection of convex shape parts that are assembled to reconstruct the overall object, but is constraint to a fixed number of slicing planes.

2.1.2 Compact model from a single image

In this thesis, we focus on generating coarse 3D models from a single image in the context of urban scenes, including buildings and floor layout.

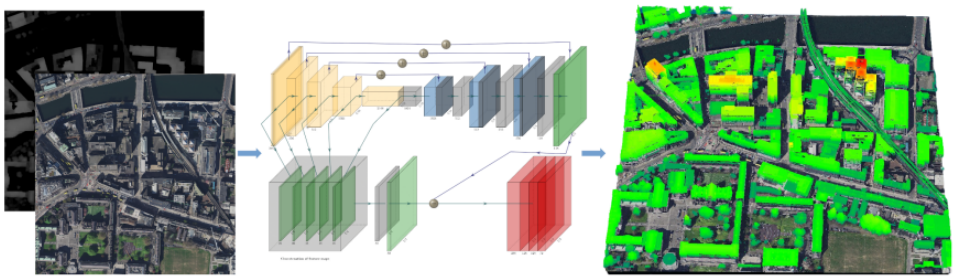


(a) (b) (c) (d)

Figure 2.1: An example of roof generation from the building footprint by the straight skeleton method. (a) Input aerial image with annotated building polygon. (b) Polygons shrinking process. (c) The straight skeleton structure obtained by the shrinking process. (d) Generated roof model with added texture. Image taken from [SK18].

2.1.2.1 Generation of building models

Reconstruction of buildings from single aerial image input is a challenging task. We categorize existing approaches in two classes according to the formulation of the sub-tasks.



(a) (b) (c)

Figure 2.2: An example of building height estimation method from aerial imagery. (a) Aerial image as input, and Lidar data as ground truth reference for training. (b) A neural network with an encoder–decoder architecture. (c) Output digital surface model (DSM) image. Image taken from [LKK+20].

Building height estimation. A typical pipeline [AA19, LKK+20, MPBF20] concerns the inference of building heights, with which a 2D footprint can be lifted into a digital surface model (DSM). Supervised learning is a popular strategy for this task. The challenge of these learning methods lies in the acquisition of high-quality training data with accurate ground truth depth and low misalignment with the input, as well as the diversity of scenes for generalizability of the model.

Roof model reconstruction. Another interesting line of work consists in constructing 3D roof models from building footprints, which lifts the 2.5D model into an enriched 3D model. Weighted straight polygons [BHH+15], as an extension to the straight polygon algorithm [AA96, SK13], has been utilized to construct realistic polyhedral roofs of houses [LD03, KW11]. The straight skeleton is a wavefront process emanated from the polygon, and the wavefront traces out a roof shape. These methods require as input a geometrically correct graph that describes the building footprint. RoofGAN [QZF21]) is a generative adversarial network that generates roof structures as a fixed number of roof primitives and their relationships. This generative approach by design can not produce output model conforming to a specified image.

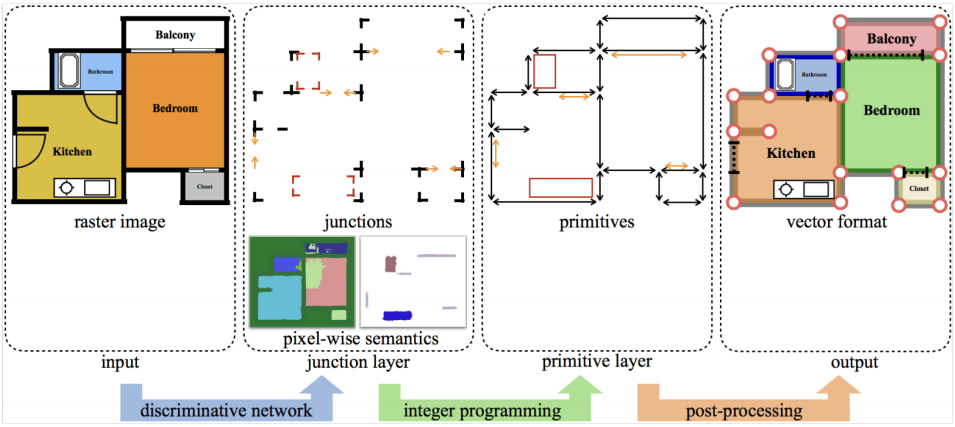


Figure 2.3: An example of the floor plan vectorization method. The input floor plan image is converted to the vectorized output via two intermediate representation layers, with one determining the junction types and locations, and the other determining a set of primitives that represent walls, opennings and icons. Image taken from [LWKF17].

2.1.2.2 Generation of floor layout

A long line of work exists on coarse 3D layout estimation from a single perspective image [LHK09, RPJT13, ML15, LBMR17] or a panorama [ZSTX14, ZCSH18, SHSC19, YWP+19], under Manhattan world assumptions. Existing approaches generate layout hypothesis on the basis of orientation maps [LHK09], line segments corresponding to vanishing points [HHF09], geometric context [HEH07], and junctions [RPJT13]. More recent techniques train neural networks for the prediction of corners [LBMR17], layout surfaces (wall, floor, ceiling) [YWP+19], boundary maps for floor-wall and ceiling-wall boundaries [SHSC19], or a combination [ZCSH18]. These methods are limited to the inference of the layout of a single room. House-scale estimation is discussed in some work [FCSS09b, CF14], but multiple images are required as input.

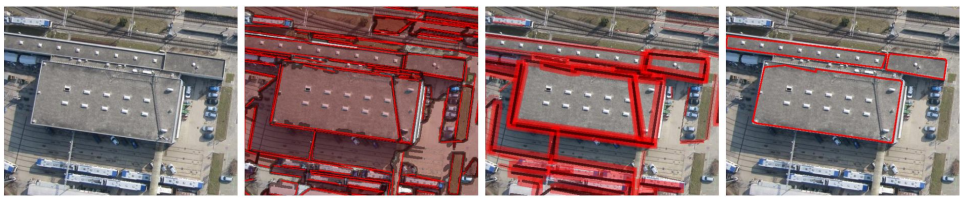
Several methods are proposed for converting the raster floor plan image into a vectorized format which provides an accurate representation of the floor layout. In particular, Raster-to-Vector [LWKF17] trained a convolutional neural network (CNN) to identify junction points in a floor plan image and connected the junctions to locate walls. A fully convolutional network (FCN) can be trained to label the pixels in a floor plan with several classes

[YZT18]. Deep floor plan recognition [ZLYF19] enriches the output by recognizing additional elements including doors, windows and types of rooms. These methods are constraint to certain types of structures or dependent on the diversity of the training set. In contrast, our formulate floor plan vectorization as a polygonal image segmentation problem which generalizes better to different types of layout. The vector floor plan can be further processed into a textured 3D geometry, given a set of photos assigned to the corresponding rooms [VWF+21].

2.1.2.3 Polygonal image segmentation

Different from traditional pixel-wise segmentation, polygonal image segmentation represents the segmented objects with polygonal shapes. We distinguish four families of existing, related methods.

Vectorization pipelines. The most popular strategy consists in extracting the object contours by chains of pixels that are then simplified into polygons. Contour extraction can be performed by various methods such as Grabcut [RKB04], superpixel grouping [LSD10] or the popular object saliency detection algorithms [CMH+15, LY16, WWL+16]. The subsequent simplification step traditionally relies upon the Douglas-Peucker algorithm [WM03] or mechanisms that simplify Delaunay triangulations [DDS09, DGCSAD11]. Because these algorithms only measure the geometric deviation from an initial configuration of highly complex polygons, their output can easily drift from the object silhouettes, leading to high accuracy loss in practice.



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

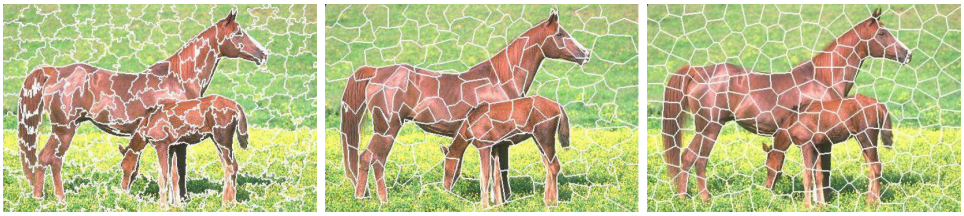
Figure 2.4: An example of assembling methods based on geometric primitives. (a) Aerial image. (b) Detected line segments. (c) Assigned graph nodes and edge weights. (d) Assembled polygonal objects. Image taken from [SCF14].

Primitive assembling methods. Another strategy consists in detecting geometric primitives such as line segments in the input image and assemble them into closed contours. The assembling step can be performed by analyzing an adjacency graph between line segments [SCF14], or by gap filling reasoning [ZFW+12]. These algorithms however do not guarantee the output polygons to be intersection-free. Polygonal Markov random fields [KvLS07] are an alternative to sample polygons from images directly. But this model is very slow to simulate in practice and operates on simple synthetic images only. Delaunay point process [FLBA20] allows the sampling of vertices within a Delaunay triangulation while grouping the triangulation facets into polygons.

(a) pixel-wise segmentation (b) direct extraction of polygons

Figure 2.5: Pixel-wise vs. polygonal segmentation of aerial images based on NN architectures. (a) Pixel-wise segmentation buildings, predicted by [LQQ+18]. Vectorizing contours of the predicted building masks requires numerous approximations and the output may drift from the building silhouettes. (b) Direct extraction of polygons using [LDWL19]. Image taken from [LDWL19].

NN architectures. Polygon-RNN [CKUF17] and its improved version [ALKF18] offer a semi-automatic object annotation with polygons. These models produce polygons with possible self-intersections and overlaps, let alone because the RNN-decoders considers only three preceding vertices when predicting the next vertex at each time step. PolyMapper [LDWL19] proposes a more advanced solution based on CNNs and RNNs with convolutional long-short term memory modules. In contrast, PolyCNN [GT18] is a CNN-based architecture that avoids self-intersections. It is however restricted to output simple polygons with only four vertices. Curve-GCN [LGK+19] alleviates the sequential nature of Polygon-RNN and predicts all



SNIC[AS17] SNICPOLY[AS17] CONPOLY[DL15]

Figure 2.6: An example of superpixels by SNIC and polygonal partitions by SNICPOLY and CONPOLY. Image taken from [AS17].

vertices simultaneously, given a predefined number of control points. In practice, these deep learning techniques give good results for extracting polygons with a low number of edges, typically residential buildings from remote sensing images. However, extracting more complex shapes with potentially hundred of edges per polygon is still a challenging issue.

Methods based on polygonal partitions. A last strategy consists in over-segmenting an image into polygonal cells, and then grouping them to approximate the object silhouettes. The vectorization of superpixels [AS17] is a straightforward way to create a polygonal partition, that is however composed of non-convex cells whose spatial connection is not clearly defined. Polygonal partitions can be more robustly created by fitting a geometric data structure on the input image. Many methods have been built upon the Line Segment Detector [VGJMR10] to geometrically characterize object contours with a set of disconnected line segments. The latter are then used for constructing a Voronoi diagram whose edges conform to these line segments [DL15], a convex mesh with constrained edges [FKF16], or a planar graph using a kinetic framework [BL18]. The cells of such polygonal partitions are then grouped to form polygons, either by graph-cut [BL18] or other aggregation mechanisms [LZMC12, RS13]. This strategy delivers accurate results when the polygonal partition fits well the input image, which is rarely the case in practice. Unfortunately, the refinement of polygonal partitions has not been deeply explored in the literature. The only solution proposed to our knowledge consists in a splitting phase which incrementally refines a Delaunay triangulation before merging the triangles [GS97]. Unfortunately, handling triangular cells does not allow to produce compact polygons.

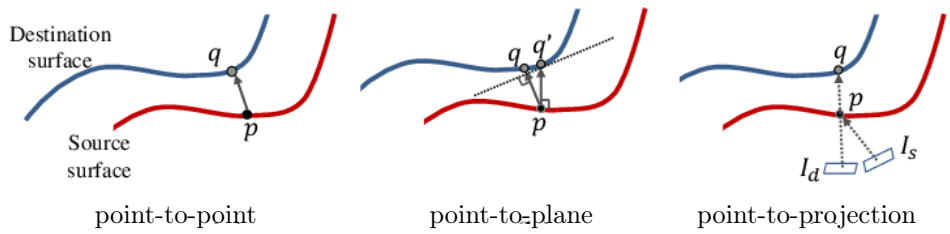


Figure 2.7: Illustration of the three categories of the ICP registration techniques. Image taken from [CWP13].

2.2 3D geometry registration

3D geometry registration aims to align multiple sets of 3D data in a common reference coordinate system. This thesis concerns the registration of rigid objects.

2.2.1 Methods with known scale

The majority of existing works focus on the registering a pair of point clouds at the same scale. We give a brief discussion on these methods:

Local registration. ICP [BM92] is the best-known algorithm for finding the SE(3) transformation between rigid objects. Variants of ICP [Fit01, Rus19, RL01, SHT09, BTP13, ZYD21] are proposed to address different issues, such as radius of convergence, computational efficiency, noise, partiality and sparsity. Probabilistic approaches as the EM-ICP [GP02] and Gaussian Mixture Model methods [JV11, EKT+15, EH18, GT19] are introduced for robustness to noise and outliers. Another branch of work concerns direct matching of the distance functions of input surfaces [PRR02, CSL13, BSKK13, SKNI16], which is more accurate and robust than ICP given a sufficient voxel grid resolution.

Global registration. A popular family of methods involve establishing feature correspondences [SWK07b, RBB09, HIT+15]. Fast Global Registration (FGR) [ZPK16] improves the inlier ratio of the correspondence set effectively by simple tests without recomputation. 4PCS [AMCO08] and Super4PCS [MAM14] effectively lower the complexity of RANSAC by ex-

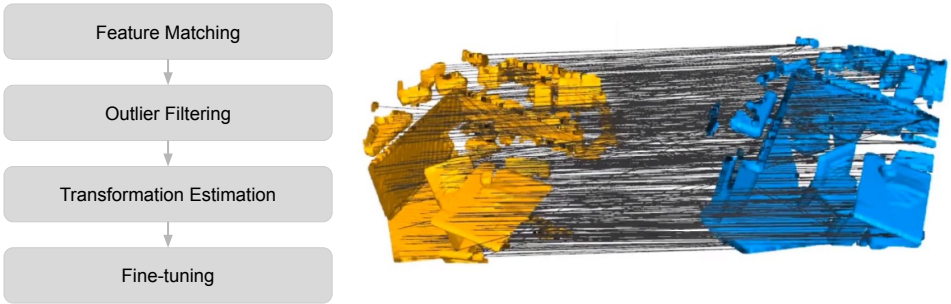


Figure 2.8: Typical pipeline for feature-based registration methods. Left: pipeline overview. Right: correspondences computed from FCGF features. Image adapted from [CDK20].

ploring the motion space by matching co-planar 4-point quadrilaterals. Another family of methods use a branch-and-bound (BnB) strategy to exhaustively explore the solution space for a good optimum, but suffer from slow convergence (Go-ICP [YLJ13], GOSMA [CPK+19]). Fast rotation search algorithm with a new bounding function for BnB has been introduced for acceleration [PBCE+16]. Eckart et al. [EKK18] propose a multi-scale point matching process using a hierarchy of Gaussian Mixtures. Another family of methods utilize the Fourier transform to decouple rotation and translation [KSA06, BB13], but is sensitive to the voxel resolution.

Learning-based methods. Recent advances in deep learning lead to the development of several neural networks for point cloud registration. The models can be roughly categorized as non-iterative and iterative methods. Non-iterative models have a natural speed advantage. Deep Closest Point [WS19a] utilizes a transformer network for feature matching coupled with SVD for point-to-point registration. DeepGMR [YEK+20] avoids point-to-point matches by integrating the network inside a probabilistic registration paradigm for lower complexity and improved robustness. Iterative methods are believed to be more robust to partially overlapping inputs [AGASL19, WS19b, LZX+20, CDK20]. In particular, PointNetLK [AGASL19] adapts PointNet into the Lucas-Kanade algorithm. PRNet [WS19b] extends DCP to an iterative pipeline with keypoint detection designed for partial-to-partial registration. IDAM [LZX+20] proposes a distance-aware similarity matrix convolution for finding correspondences. Deep Global Registration [CDK20]

is an end-to-end 6D ConvNet built upon FCGF [FA20] and works well on real-world dataset. MS-SVConv [HDG21] proposes a multi-scale network to improve the generalizability of U-Net architectures for point cloud registration.

2.2.2 Methods with relative scale estimation

The registration of multi-modal geometric data often involves estimating the relative scale between different types of data, e.g. when aligning a CAD model to a point cloud scan. Please refer to Saiti et al. [ST20] for a survey covering issues and methods related to this task. The most straightforward method which simply normalizes scales in pre-processing [HZF+17] is unsuitable for partially overlapping and noisy data. Extensions of ICP integrate scale factor estimation by including a separate minimization step [ZSN05, Rus19], by incorporating a bounded scale matrix [DZY+07], by registering cumulative contribution rate curves [LTR+13], and by using the maximum correntropy criterion [WCD+19]. Coherent Point Drift [MS10] and its extensions [Hir20b, Hir20a] formulate the task as a probability density estimation problem and re-parametrizes GMM centroids with rigid parameters including the scale. Corsini et al. [CDG+13] extend 4PCS and propose a method for point-cloud-to-3D-model registration. Bulow et al. [BB18] extend The Fourier transform approach to incorporate scale estimation. Paudel et al. [PHDV15] formulate the task as a point-to-plane assignement problem utilizing a plane-based assumption of the 3D scene. Mellado et al. [MDS16] introduce a descriptor based on Growing Least Squares for scale-invariant matching.

Another sub-family of methods concern aligning CAD models from a collection of prespecified categories to depth scans. These approaches determine the scale via object detection in terms of 3D bounding boxes, therefore are limited to training categories. Among these studies, Song et al. [SX14] assume that the gravity direction is known and estimate rotation only around the gravity axis. Gupta et al. [GAGM15] rely on traditional ICP for aligning the input point set and the point set rendered from the model. Izadinia et al. [IS20] proposes a learning-based ICP approach which formulates the rotation estimation problem as a policy learning task for viewpoint prediction. Deformation of the CAD model is considered by a few works [NXS12, ADD+19, IBA+20] for better fitting. Our approach differs from the above pipelines by integrating scale, rotation and translation into a single

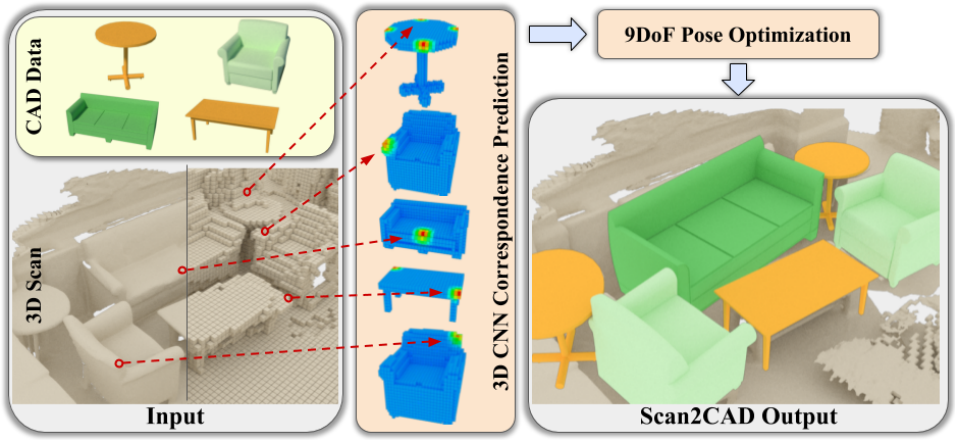


Figure 2.9: An example of CAD model alignment against input depth scans via a 3D CNN approach, which predicts heatmap correspondences between the scan and the CAD model. Image taken from [ADD+19].

optimization framework.

Chapter 3

Polygonal image segmentation

3.1 Introduction

Extracting objects in images is traditionally operated at the pixel scale, one object being represented as a group of pixels. Such a resolution-dependent representation is often not adapted to the end-users. In many application scenarios as urban mapping or sketching, objects need to be captured with more compact and editable vector representations. In particular, polygons with floating coordinates allow both the approximation of free-form shapes, e.g., organic objects, and the fine description of piecewise-linear structures, e.g., buildings and many other man-made objects.

We consider the task of capturing objects in images by polygons with three objectives. First, *fidelity*: the output polygons should approximate well the object silhouettes in the input image. Second, *complexity*: the output polygons should be composed of a small number of edges to offer a compact and editable representation. Last, *geometric guarantees*: the output polygons should be intersection-free, closed, potentially with holes, and form an image partition.

The simplest way to capture the silhouette of an object as a polygon is to vectorize a chain of pixels representing the object contours [DGCSAD11, DDS09, WM03]. While the complexity of the polygon can be easily controlled, these simplification processes do not take into account structural information contained in the input image. Consequently, output polygons are often imprecise, typically with edges that do not fit accurately the object silhouettes. Recent works on the partitioning of images into polygonal cells [AS17, BL18, DL15, FKF16] suggest that grouping cells from these partitions can produce more accurate results than traditional vectorization methods. This strategy however suffers from imprecise partitions, typically with some polygonal cells overlapping two different objects. Existing works in the field focus on merging polygonal cells only and omit the necessity of

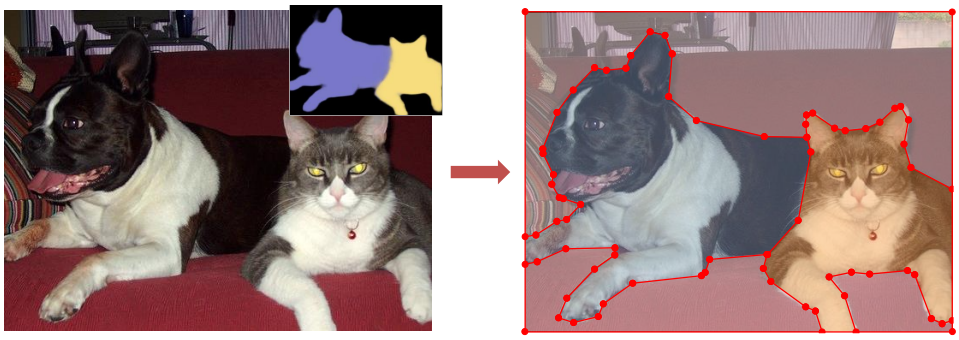


Figure 3.1: Goal of our approach. Our algorithm takes as input an image with a rough semantic probability map and outputs a set of low-complexity polygons capturing accurately the objects of interest, here dogs and cats.

splitting operations to deliver more precise results.

In this work, we propose an algorithm to capture objects by compact floating polygons. Inspired by mesh deformation techniques in Geometry Processing, the main idea consists in refining the geometry of an imprecise polygonal partition while labeling each cells by a semantic class. Our algorithm produces a compact representation for objects while offering geometric guarantees including self-intersection-free and overlap-free polygons. We demonstrate the potential of our method on different types of scenes, from organic shapes to man-made objects through aerial images and line-drawing sketches, and show its efficiency with respect to existing vectorization approaches.

3.2 Background on kinetic data structures

In this section, we introduce kinetic data structures, which are the fundamental data structures used for maintaining the geometry of polygonal partitions during the refinement process of the proposed algorithm.

A kinetic data structure [BGH99, Gui04] is defined as a data structure implemented to track an attribute of a geometric system consisting of a set of geometric primitives whose coordinates are continuous functions of time. In contrast to static geometric data structures that divide the continuous spatial domain into a discrete set of combinatorial structures of geometric objects, kinetic data structures divide the continuous time domain into a set of disjoint intervals. In each interval the combinatorial structure built

on top of the geometric primitives does not change. The validity of the combinatorial structure within an interval can be verified by checking a finite number of certificates of the geometric primitives.

As primitives move, events may occur when certificates become invalid, e.g. collision between primitives. Kinetic data structures maintain a global event queue which is ordered dynamically according to the times of occurrences of the events. When an event occurs, the geometric objects responsible for the certificate failures as well as the queue ifself are updated, therefore the combinatorial structure remains valid. A key insight of kinetic data structures is that, by looking at a small time interval into the future, the queued events correspond to possible combinatorial changes involving a constant and typically small number of geometric objects each.

Kinetic data structures have been applied to many applications including dynamic Delaunay triangulations for moving vertices [AGG+10], polygonal partitioning of images [BL18], and polyhedral surface reconstruction [BBPDM08, BL20]. As a part of this work, we design a kinetic data structure for the propagation of a collection of line-segments inside 2D nested polygons. Please refer to the appendix for a detailed description of the propagation and a pseudo-code.

3.3 Algorithm

The algorithm takes as input an image and an associated probability map that estimates the probability of each pixel to belong to the different classes of interest. This probability map is typically generated by state-of-the-art semantic segmentation methods or saliency detection algorithms.

The algorithm departs from a polygonal partition generated by kinetic propagation of line segments [BL18]. Each cell of this partition is enriched by a semantic label chosen as the class of interest with the highest mean over the inside pixels in the probability map. The goal of our algorithm is then to refine this *semantic polygonal partition* by splitting and merging cells in tandem. These refinement operations are guided by an energy that accounts for both fidelity to input data and complexity of output.

The algorithm ends when no splitting or merging operations can decrease the energy anymore. Each cell in the output is a polygon associated with a class of interest, as illustrated in Fig. 3.1. By construction, the set of output

polygons is guaranteed to recover the entire image domain without overlaps, to be closed and intersection-free, and does not contain edge-adjacent cells with the same semantic label.

We denote a semantic polygonal partition by $\mathbf{x} = (\mathbf{m}, \mathbf{l})$ where $\mathbf{m}$ defines a 2D polygon mesh on the image domain while $\mathbf{l}$ represents the semantic labels associated to the facets of $\mathbf{m}$. We denote by $\mathcal{F}_{\mathbf{x}}$ (respectively $\mathcal{E}_{\mathbf{x}}$) the set of facets (resp. non-border edges) of the polygon mesh $\mathbf{m}$.

3.3.1 Energy formulation

We measure the quality of a semantic polygonal partition $\mathbf{x}$ with an energy function U of the form:

$$U(\mathbf{x}) = (1-\lambda)U_{fidelity}(\mathbf{x}) + \lambda U_{complexity}(\mathbf{x}) \tag{3.1}$$

The first term $U_{fidelity}$ measures the coherence of the configuration $\mathbf{x}$ with the input data while $U_{complexity}$ encourages low-complexity outputs. These two terms, that are balanced by a model parameter $\lambda \in [0, 1]$, are typically expressed with local energies on the edges and facets of the mesh $\mathbf{m}$.

Fidelity term $U_{fidelity}$ has two objectives: (i) encouraging the semantic label of each facet to be coherent with the probability map, and (ii) encouraging edges to align with high gradients of the input image. These objectives are balanced by parameter β, set to 10^{-3} in our experiments:

$$U_{fidelity}(\mathbf{x}) = \sum_{f \in \mathcal{F}_{\mathbf{x}}} -w_f \log P_{map}(l_f) + \beta \sum_{e \in \mathcal{E}_{\mathbf{x}}} w_e A(e) \tag{3.2}$$

where w_f is the ratio of the area of facet f to the area of the whole image domain, $P_{map}(l_f)$ is the mean of the probability map for class l_f over the pixels inside facet f, and w_e is the inverse of the length of the image diagonal if the two adjacent facets f and f' of edge e have different labels $l_f \neq l_{f'}$, and 0 otherwise. Finally, $A(e)$ is a function measuring the alignment of edge e with image gradients:

$$A(e) = \sum_{i \in N_e} r_i \big[1 - \hat{F}(m_i) \exp\big(-\frac{{\Delta\theta_i}^2}{2\sigma^2}\big)\big] \tag{3.3}$$

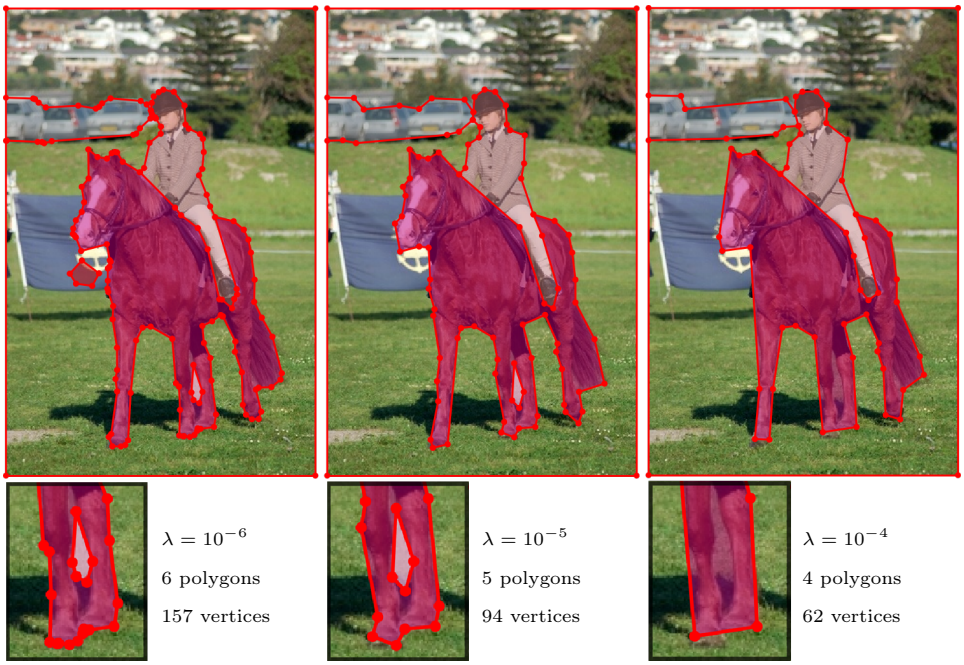
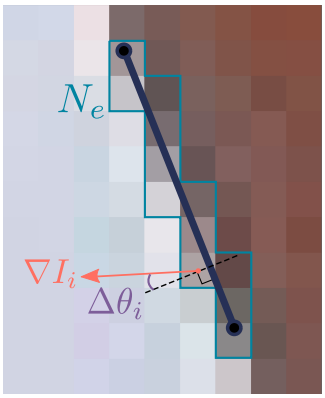


Figure 3.2: Trade-off between fidelity to data and complexity to output polygons. Increasing λ gives more compact, yet less accurate, output polygons. Objects of interest: horses, persons and cars.



where N_e is the set of pixels that overlap with edge e, r_i is the inverse of the number of edges that overlap pixel i, $\Delta\theta_i$ is the angular difference between the gradient direction at pixel i and the normal vector of edge e, and σ is a model parameter set to $\frac{\pi}{8}$ in our experiments. Denoting $\hat{F}$ the empirical cumulative density distribution of gradient magnitudes over the input image, $\hat{F}(m_i)$ is the probability that the gradient magnitude of a random pixel in the input image is smaller than the gradient magnitude m_i at pixel i. Note that, instead of image gradients, more sophisticated discontinuity maps such as [IZKA14, DZ13] could be used by modifying the density distribution $\hat{F}$ in Eq. (3.3).

Complexity term $U_{complexity}$ penalizes a complex polygon mesh with

the number of edges (the lower, the better):

$$U_{complexity}(\mathbf{x}) = |\mathcal{E}_{\mathbf{x}}| \tag{3.4}$$

As illustrated in Figure 3.2, the model parameter λ is a trade-off between fidelity to input data and complexity of the output polygons. Note that our data term measures data fidelity independently of polygon complexity. In particular, $A(e)$ is designed as a linear function so that, if an edge e is composed of two collinear edges e_1 and e_2, then $A(e) = A(e_1) + A(e_2)$. The linearity of $A(e)$ requires that each gradient pixel should not contribute multiple times to the total energy, which explains the factor r_i in Eq. (3.3).

3.3.2 Exploration mechanism

Both continuous variables for representing the polygon mesh and discrete semantic labels are involved in the minimization of the (non-convex) energy U. Inspired by edge contraction algorithms for simplifying triangle meshes [BKP+10, GH97a], we explore efficiently such a large solution space via an iterative mechanism based on local operators that split and merge facets of the polygon mesh $\mathbf{m}$. Starting from an initial configuration, we compute the energy variations for splitting each facet as well as the energy variations for merging each pair of adjacent facets. All the energy variations (values to add to the energy if performing the corresponding operation) are sorted and saved into a priority queue in ascending order, i.e., with more negative energy variations first. The exploration mechanism then consists in operating the splitting or merging at the top of the priority queue, i.e., the operation that gives the highest energy decrease. This modification is followed by an update of the priority queue. A pseudo-code of the exploration mechanism is given in Algorithm 1. We also provide visualization of the iteration process in Figure 3.3. The numbers of merging and splitting operated per one hundred iterations indicates that the splitting operations are dominating at the beginning and is taken over by merging as the process continues. We now detail the main components of this algorithm.

Initialization. Because the exploration mechanism finds a local minimum, a good initial configuration is required. In our experiments, we build the initial semantic polygonal partition using the kinetic partitioning method proposed in [BL18]. It produces in a fast and scalable manner a partition of

Algorithm 1 Pseudo-code of exploration mechanism

```
Initialize the semantic polygonal partition x
Initialize the priority queue Q
while the top operation i of Q decreases energy U do
    Update x with the merging or splitting operation i
    Update Q
end while
```

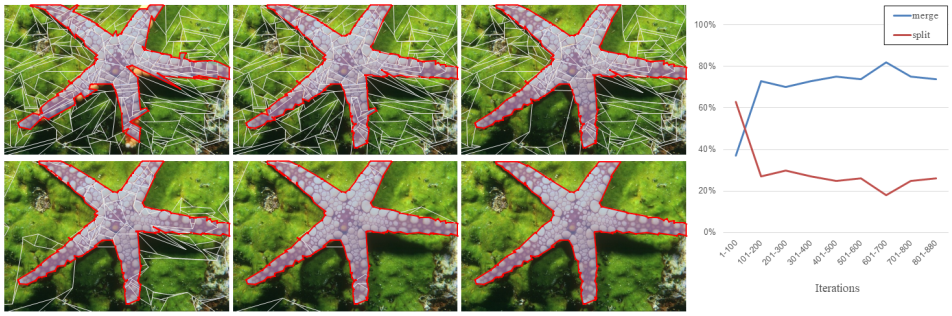


Figure 3.3: The iteration process on a foreground extraction example. The snapshots are taken at the initial configuration, then at every 200 iterations, and finally when the algorithm terminates. The boundary edges between the object of interest and the background is highlighted with red, and the other edges are in white. On the right, the plot records the portions of merging and splitting operations for each interval of 100 iterations.

polygonal cells that captures well the homogeneous regions in images. This partition is turned into a 2D polygon mesh. We then assign to each facet the semantic label that returns the highest mean over the inside pixels in the probability map. The impact of initialization is illustrated in Figure 3.4.

Merging operator. The merging operator merges two facets with at least one edge in common into a single facet. The update consists in removing all common edges in between the two original facets as illustrated in Figure 3.5. In case of the occurance of vertices of valence 2 whose incident edges are collinear, these vertices is also removed. The semantic label of the new, merged facet is chosen as the most probable label with respect to the probability map.

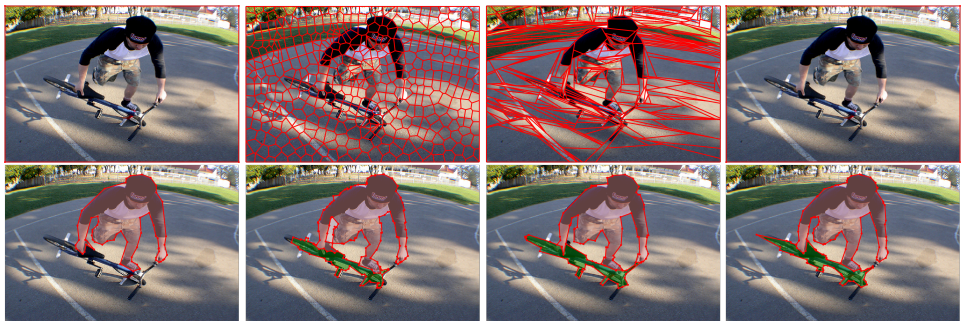
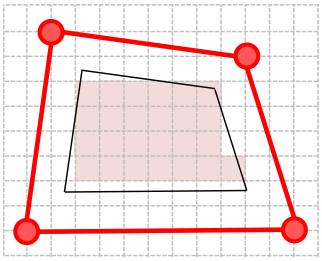


Figure 3.4: Initialization. The top (resp. bottom) row shows the initial partitions (resp. output polygons). Objects of interest are persons and bikes. Starting the exploration mechanism from a partition composed of one rectangular facet (column 1) typically produces results with missing objects such as the bike. An initial Voronoi partition [DL15] (column 2) is too fragmented to output low complexity polygons. Our algorithm performs best from kinetic partitions [BL18] (column 3) with a good trade-off between accuracy and polygon complexity. This option returns similar results than a simulated annealing exploration (column 4) but with processing times reduced by two orders of magnitude. For clarity reasons, here and in the following figures, we do not display the background polygons (at the image border) in the visual results.

Splitting operator. The splitting operator divides a facet into a number of new facets by inserting new edges and vertices. We first detect a set of cutting directions inside the original facet. These directions are found by fitting line segments to the input image with a region growing algorithm [OVJ+18]. To avoid detecting line segments overlapping the edges of the facet, only pixels inside the facet shrunk by 2 pixels are considered for the fitting (see the set of pink pixels inside the red facet in the inset). The detected line segments are then propagated until they collide with each other or the outside edges of the original facet, as illustrated in Figure 3.5. The geometric structure throughout the propagation inside arbitrary polygonal boundaries is maintained by adapting a kinetic data structure [BGH99]. The collision points



(respectively the prolonged line segments) cor-
respond to new vertices (resp. edges) inserted in the 2D polygon mesh. For each new facet, we associate the most probable semantic label with respect to the probability map. If two new adjacent (sub)facets have the same semantic label, they are immediately merged, as part of the splitting operation.

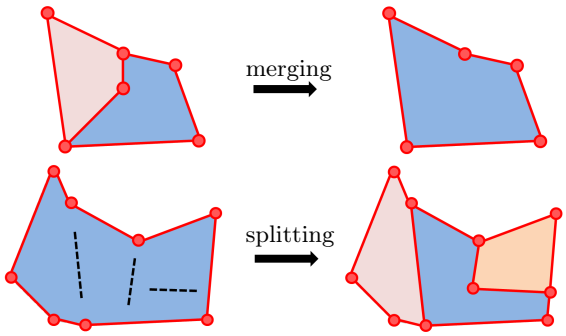


Figure 3.5: Merging and splitting operators. The merging operator merges two adjacent facets with different semantic labels by removing the common edges (top). The splitting operator divides one facet into multiple facets that have different semantic labels (bottom). The black dashed lines indicates the cutting directions detected in the input image (bottom left).

Priority queue. After a configuration $\mathbf{x}$ is modified, the priority queue must be updated. We first remove from the priority queue the current operation and all the merging or splitting operations concerning the modified facets. We then compute the energy variations of all possible operations that can affect the new facets and insert them in the priority queue, appropriately sorted. Because the energy is formulated as the sum of local terms and a global complexity term, these variations are not costly to compute. When a split occurs, only the parent facet, its new split facets and the edges composing these facets are involved in the energy updates of the priority queue. These updates are fast and local; they do not propagate through the whole mesh. In our experiments, the average number of facets created per split is 2.1 and the average number of updated edges is 7.2.

Stopping criterion. The exploration mechanism ends when the energy variations sorted in the priority queue become all positive, i.e., when no first-order operation can decrease the energy anymore. Note that this criterion

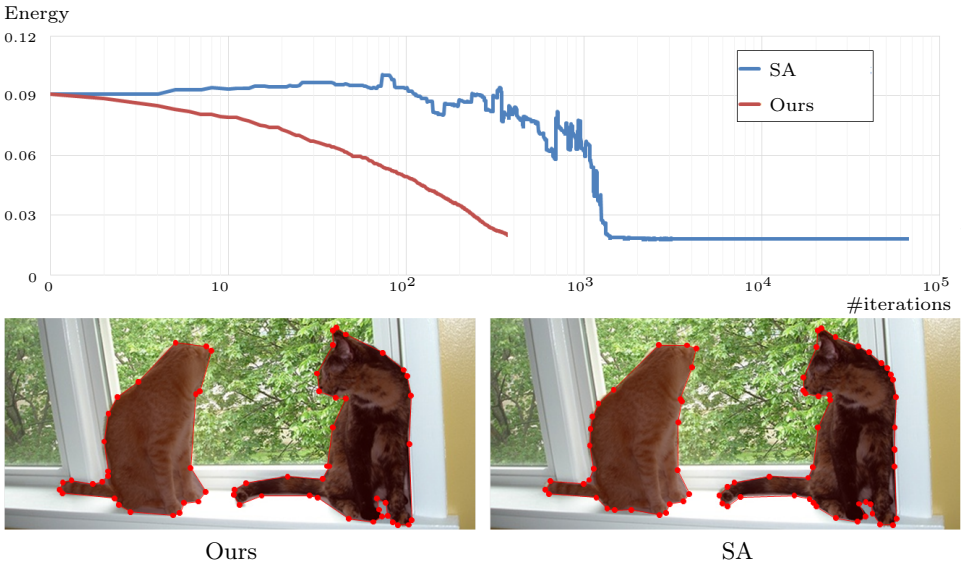


Figure 3.6: Evolution of energy U during our exploration mechanism (red curve) and a simulated annealing optimization (SA, blue curve). While the two optimization techniques converge towards a similar energy, our exploration mechanism requires two orders of magnitude less iterations than the simulated annealing.

guarantees the exploration mechanism to converge quickly without bumping effects or getting trapped in loops. Besides, the final solution cannot contain two edge-adjacent polygons with the same semantic class, as merging them necessarily decreases the energy (lower $U_{fidelity}$, thanks to the convexity of $-\log$, and lower $U_{complexity}$).

Details for speeding-up the exploration. The exploration mechanism is local. This choice is motivated by low running time and the presence of good initial configurations. (An alternative could be to use a non-local optimization algorithm such as the simulated annealing, cf. Figure 3.6.) Observing that a complex initial partition often over-segments the probability map, we initially (before exploration) merge all adjacent facets that contain only pixels classified with the same label. This highly reduces the processing time without affecting the results.

To reduce the time for detecting line segments when new splitting operations are considered, we allow a merged facet to inherit the already-detected line segments of its parent facets. We detect new line segments only in the

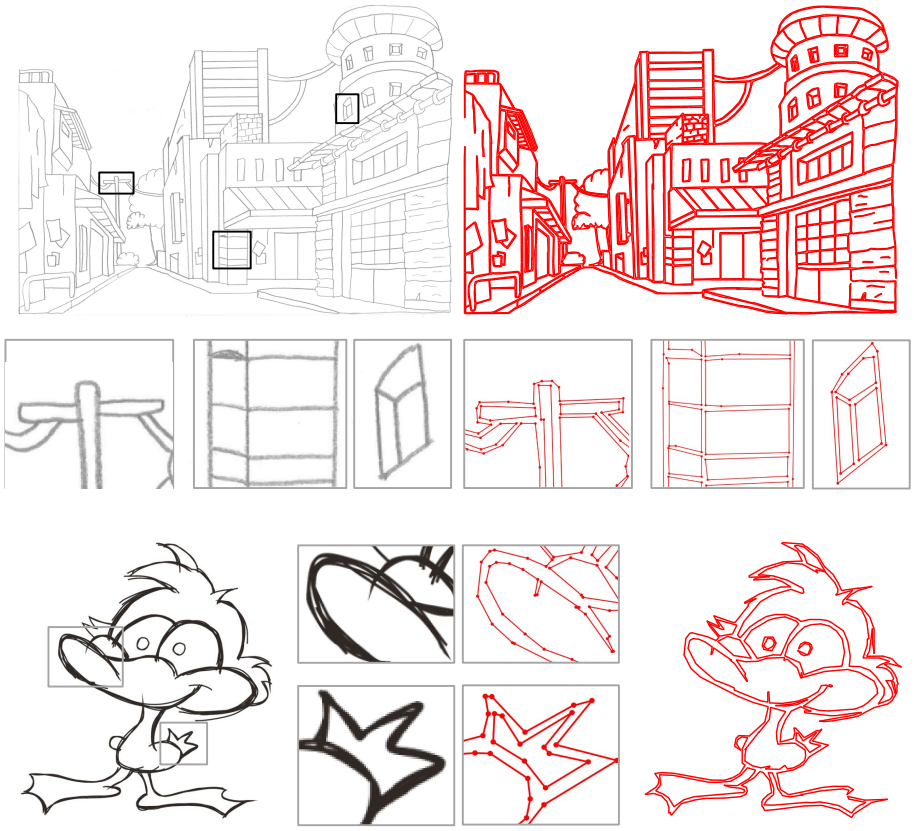


Figure 3.7: Vectorization of linear structures. Our algorithm can be used to vectorize line-drawings. While thin, these linear structures can be captured by compact polygons with a good accuracy (see closeups).

area around the removed edges. In addition to time savings, this allows us to refine the edges between two adjacent facets by operating a merging and then a splitting on the same facet.

3.4 Experiments

Our algorithm has been implemented in C++ using the Computational Geometry Algorithms Library (CGAL) [The]. All experiments have been done on a single computer with Intel Core i7 processor clocked at 2.4GHz.

Parameters. We have 3 model parameters λ, β, σ, that are set respectively to 10^{-5}, 10^{-3}, $\frac{\pi}{8}$ in all experiments, despite the dataset variety. (Note that our algorithm does not need any threshold to stop the exploration.) The

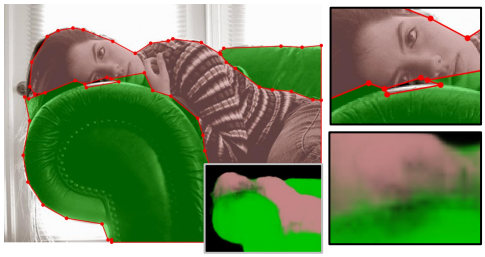
values of λ and β were chosen based on a grid search; σ was set to roughly model the standard deviation of gradient directions.

3.4.1 Flexibility and robustness.

Our algorithm has been tested on different types of scenes and objects. Piecewise-linear structures such as buildings are captured with fine details as long as probability maps have a good accuracy. Organic shapes such as humans and animals are approximated by low complexity polygons. In addition to the silhouettes of objects in images, our algorithm can also be used to vectorize line-drawing sketches. This applications usually require the use of specialized methods to detect, filter and connect strokes into a network of parametric curves [FLB16]. In contrast, our algorithm finely reconstructs these linear structures, as illustrated in Figure 3.7. Our algorithm offers a good robustness to imprecise probability maps thanks to the second part of the data term that favors the alignment of edges with image discontinuities. As illustrated in Figure 3.8, the output polygons can accurately capture the silhouette of objects even if the probability map is ambiguous where different objects meet.

3.4.2 Ablation study

As semantic maps sometimes suffer from ambiguities at object boundaries, using only the first data term yields polygons that do not contour well the objects, as illustrated in the inset. This result, obtained with β=0, must be compared with the results obtained in Figure 3.8 in the same initial conditions but with $\beta \neq 0$.



3.4.3 Quantitative evaluation.

We compared our algorithm to state-of-the-art methods on three different datasets.

We first tested our algorithm on the HKU-IS dataset [LY16] designed to evaluate salient object detection methods. We computed the probability

Figure 3.8: Vectorization of multi-class objects. Probability maps are often ambiguous and only roughly indicate the shape of the objects (see colors for different classes). Our algorithm captures the silhouette of theses objects with low-complexity polygons with a good precision. Note in particular how the polygons nicely delineate close objects, such as the lady face and the couch (see closeups). A failure case is shown in the bottom right example where the quality of the probability map is too poor to capture the underlying object. Images are from the PASCAL VOC2012 dataset.

map for each image using the algorithm of Li and Yu [LY16]. We compared our algorithm to two vectorization pipelines in which the same saliency maps [LY16] are binarized before chaining and simplifying the pixels on the object contours, either by the popular Douglas-Peucker algorithm [WM03] or by polyline decimation [DDS09]. We also compared to two cell grouping algo-

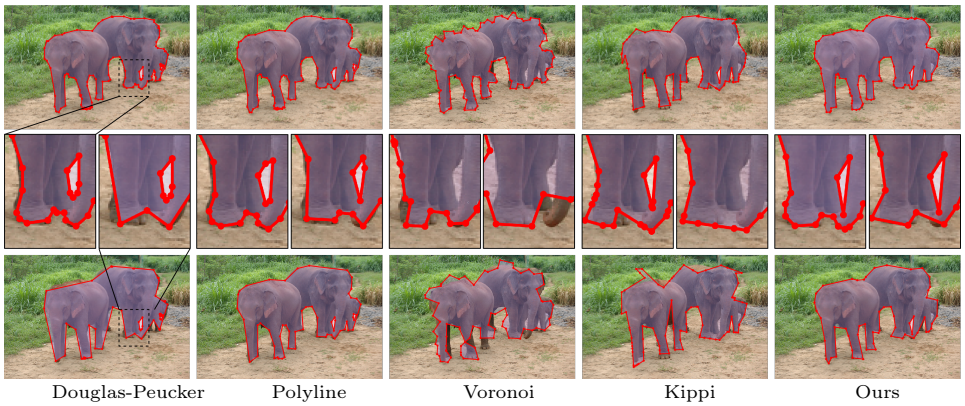


Figure 3.9: Visual comparisons at two compression ratios: 10 (top) and 33 (bottom). While the vectorization pipelines Polyline and, to a lesser extent, Douglas-Peucker yield accurate polygons at low compression, their precision drops at high compression, with polygons not aligning well with silhouettes anymore (cf. closeups). The cell-grouping algorithms Voronoi and Kippi are less accurate on such free-form shapes where cells often overlap several object classes. In contrast, we accurately capture the elephants at both compression ratios.

rithms that generate a polygonal partition by Voronoi diagram construction [DL15] or by kinetic propagation of line segments [BL18]. The polygons are then extracted from these partitions by thresholding the saliency map averaged over each cell. We denote these methods respectively by Voronoi and Kippi. The accuracy is measured using Intersection-over-Union of our pixelized output polygons against the ground truth. We also measure compression as the ratio of the number of pixels of the ground truth region boundary to the number of polygon vertices. In practice, we produce polygons at different complexity by varying λ, as shown in Figure 3.2.

Table 3.1 shows the evolution of accuracy against compression on the HKU-IS dataset. While all methods exploit the same saliency maps, only our algorithm maintains high accuracy at high compression ratios, i.e., when the output polygons have a very low number of vertices. Fig. 3.9 shows visual comparisons of the methods at low and high compression. At low compression, the vectorization pipelines Douglas-Peucker and Polyline produce accurate polygons, similarly to our algorithm. Because these pipelines simplify the geometry of polygons without taking into account consistency

Method	Compression ratio					
	10	15	20	25	33	50
Voronoi [DL15]	77.7	75.2	71.6	68.2	64.1	57.5
Kippi [BL18]	79.2	77.1	72.8	69.5	65.6	62.1
Douglas-Peucker [WM03]	83.8	83.3	81.2	79.4	76.0	65.7
Polyline [DDS09]	83.9	83.7	82.5	81.2	77.5	69.0
Ours	**84.1**	**84.0**	**83.7**	**83.1**	**81.3**	**77.0**

Table 3.1: Accuracy (%) vs compression on HKU-IS.

Method	Compression ratio					
	10	15	20	25	33	50
Voronoi [DL15]	87.9	86.4	83.6	81.3	77.7	74.3
Kippi [BL18]	88.9	87.6	85.4	83.0	79.5	75.2
Douglas-Peucker [WM03]	91.2	90.9	90.1	88.8	86.6	79.8
Polyline [DDS09]	91.2	91.1	90.6	89.9	88.1	85.8
Ours	**91.7**	**91.6**	**91.5**	**91.4**	**91.2**	**89.7**

Table 3.2: Accuracy (%) vs compression on PASCAL VOC2012.

the image, their accuracy significantly drops for higher compression ratios, typically from 25. Cell grouping methods Voronoi and Kippi suffer from imperfect polygonal partitions where cells often overlap several types of objects. In contrast, the merging and splitting operations of our algorithm allow us to refine cells with respect to the probability map and the input image.

We also tested our algorithm on the Pascal VOC2012 dataset [EVGW+] designed for multi-class segmentation tasks. This dataset contains 20 object classes and 1 background class. The evaluation was performed on the validation set. We compared our algorithm to the same four methods (Douglas-Peucker, Polyline, Voronoi and Kippi) with the same accuracy and compression metrics. Probability maps were generated by the DeepLab algorithm [CZP+18] by taking the output layer before the final argmax operation over class channels. Table 3.2 shows the evolution of accuracy against compression for the five algorithms. Similarly to the quantitative results obtained on the HKU-IS dataset, our algorithm outclasses the other methods, in partic-

Method	AP	AP_{50}	AP_{75}	AR	AR_{50}	AR_{75}
R-CNN [HGDG17]	41.9	67.5	48.8	47.6	70.8	55.5
PANet [LQQ+18]	50.7	73.9	62.6	54.4	74.5	65.2
PolyMapper[LDWL19]	55.7	86.0	65.1	62.1	88.6	71.4
Ours	**65.8**	**87.6**	**73.4**	**78.7**	**94.3**	**86.1**

Table 3.3: Performance on the CrowdAI mapping challenge dataset. Average precision (AP) and average recall (AR) in %.

ular with a significant accuracy gain at high compression. Figure 3.8 shows visual results obtained by our algorithm on different object classes.

We finally tested our algorithm on the CrowdAI mapping challenge dataset [Moha] which is composed of ∼60k satellite images of urban landscapes. Probability maps were generated using a U-Net variant [CKPT18]. We followed the same experimental protocol than in [LDWL19] for extracting the contours of buildings from this dataset. In particular, we used the same average precision (AP) and average recall (AR) metrics. We compared our algorithm with the deep learning methods PolyMapper [LDWL19], Mask R-CNN [HGDG17] based on the implementation of [Mohb], and PANet [LQQ+18]. Table 3.3 presents the quantitative results on these four methods. Our algorithm obtains the best average precision and average recall scores. In particular, our algorithm outclasses Polymapper with significant gains. This difference is partly explained by the iterative mechanism of vertex insertion of Polymapper whose efficiency decreases for complex shapes. By refining polygonal cells on a topologically-valid partition, our algorithm does not suffer from this problem. Figure 3.10 shows visual results on an urban scene of the Inria Aerial Image Labeling dataset [MTCA17].

3.4.4 Performance.

Figure 3.6 shows that our exploration mechanism reaches similar energies as a non-local simulated annealing while being two orders of magnitude faster. Our exploration mechanism is inspired by edge contraction algorithms for mesh simplification. While local, greedy and old, such algorithms, e.g., [BKP+10, GH97a], are still very popular and commonly used in the field. As shown in the inset, our algorithm typically requires a few seconds for a

Figure 3.10: Extraction of buildings from satellite images with our algorithm: 1,178 buildings of a half square kilometer area of Chicago, USA, are extracted with low complexity polygons (8,683 vertices). While compact, the polygons capture some fine details (see closeups).

100K-pixel image and about 2 min for a 10M-pixel image. Note that our code has not been optimized (beyond the general strategy expressed at the end of Sect. 3.3.2). In particular, the exploration mechanism runs sequentially on CPU (no parallelization). The most time-consuming operation is the update of the priority queue, and especially the simulation of splitting operations for the new large facets. If the initial partition contains N_f facets and N_e non-border edges, the priority queue is constructed by sorting the energy variations of the N_f possible splits and N_e possible merges; the running time for this is negligible ($< 0.1\%$ of total time). Last, the computation of cutting directions depends on the number of image pixels. It is very fast and performed only once at priority queue initialization. Getting split directions from the input image lowers the dependency on the initial partition and allows larger solution space explorations.

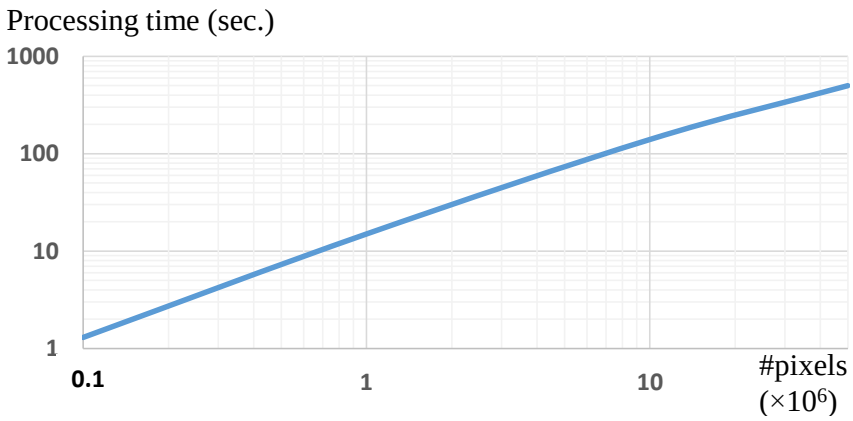


Figure 3.11: Processing time against number of pixels in the input image.

3.4.5 Limitations.

As energy $U(\mathbf{x})$ is not convex and as our exploration mechanism is local, results depend on the quality of the initial partition. As shown in Fig. 3.4, splits provide robustness to a range of under-segmentations; yet, an initial partition that over-segments well the image leads to more accurate results. If a good initial partition cannot be provided or guaranteed, simulated annealing can be a better choice regarding accuracy, but not running time (cf. Fig. 3.6).

Thanks to the gradient alignment term in $U_{fidelity}$, our algorithm is robust to some level of error or ambiguity in semantic maps, in particular at object border; see, e.g., the polygons capturing the lady's face and the couch from the blurry semantic map in Fig. 3.8. Yet, the class probability of most pixels has to be correct, as is also the case for shape grammar parsers. Note that depending on external methods (initial partition, semantic map) is a strength: our performance will improve along with the related state of the art.

Also, while parameter λ balances data fidelity and output complexity, it does not allow to control the exact number of output vertices, contrary to vectorization pipelines.

3.5 Conclusion

We proposed an algorithm for extracting and vectorizing objects in images with low-complexity polygons. Our algorithm refines the geometry of an

initial polygonal partition while labeling its cells by a semantic class. Based on local merging and splitting of cells, the underlying mechanism is simple, efficient and guaranteed to deliver intersection-free polygons. We demonstrated the robustness and the flexibility of our algorithm on a variety of scenes from organic shapes to man-made objects through floor maps and line-drawing sketches. We also showed on different datasets that it outperforms the state-of-the-art vectorization methods.

Chapter 4

3D registration of multi-modal geometry

4.1 Introduction

3D registration of multi-modal data is a long-standing challenge when working with real-world 3D objects. Geometric data obtained from different acquisition modalities (e.g. laser scans, multi-view stereo reconstruction) or created by modeling tools are represented in various forms, i.e. as point clouds or meshes, and exhibit different geometric properties in terms of noise, resolution or the scale. Classical problems in multi-modal registration involve registering a low-quality point cloud to a high-quality mesh, and registering a dense point cloud to a simplified mesh model.

Challenges in multi-modal registration arise from several aspects. Imperfection in data acquisition includes occlusions and non-uniform sampling density. Different surface representations, i.e. meshes and point clouds, often have different levels of detail and accuracy, making both traditional feature-based methods [SWK07b, RBB09, HIT+15, ZPK16] and deep learning architectures [AGASL19, WS19a, WS19b, CDK20, LZX+20] unsuited for this task. Variation in acquisition modalities can lead to scale ambiguity, e.g. multi-view stereo generates data in an unknown scale, which further complicates the problem. The majority of existing methods [BM92, YLJ13, MAM14, PBCE+16, CPK+19, ZYD21] focus on aligning 3D models to depth scans under the assumption that the model and the depth scan are already at the same scale. This is not the case for many real-world scenarios, where either the collected data or the 3D object model may have no absolute scale associated. Simple pre-processing by estimating and correcting the scale before calling the registration step often fails for non-uniformly sampled data or partially overlapping data. Several works [CDG+13, GAGM15, MDS16, EKK18, IBA+20] have considered relative

scale estimation. These methods treat the scale estimation as a separate step, therefore there lacks a unified formulation that simultaneously solve for the scale, rotation and translation.

In this work, we present a global registration algorithm from multi-modal geometric data, typically 3D point clouds and meshes. Existing feature-based methods and recent deep learning based approaches typically rely upon point-to-point matching strategies that often fail to deliver accurate results from defect-laden data. In contrast, we reason at the scale of planar shapes whose detection from input data offers robustness on a range of defects, from noise to outliers through heterogeneous sampling. The detected planar shapes are projected into an accumulation space from which a rotational alignment is operated. A second step then refines the result with a local continuous optimization which also estimates the scale. We demonstrate the robustness and efficacy of our algorithm on challenging real-world data. In particular, we show that our algorithm competes well against state-of-the-art methods, especially on piece-wise planar objects and scenes.

4.2 Background on Lie groups for Sim(3)

In this section, we introduce Lie groups and Lie algebras for 3D transformations, which are the fundamental representation that enables our continuous optimization step.

A Lie group is a continuous group that is also a differentiable manifold, in which the group operations of multiplication and inversion are smooth maps. Associated with every Lie group is a Lie algebra, which is the tangent space around the identity element of the group. It has become the building block for SLAM frameworks [Sel04, ZXB+19, DS20], as it addresses essential operations including composition, inversion, differentiation and interpolation for spatial transformations. A complete introduction of Lie groups and Lie algebras can be found in [Kir08].

Transformations in 2D and 3D space can be represented as Lie groups. In particular, the similarity transform matrix belongs to the Sim(3) Lie group. Similarity transformations are combinations of rigid transformation and scaling. When the scale $s = 1$, the transformation becomes rigid and belongs to the SE(3) group. The associated Lie algebra sim(3) of the Sim(3) group is a vector space of dimension 7, which is the same as the number of de-

grees of freedom of similarity transformations. The basis element of the Lie algebra are called generators. The generators of sim(3) correspond to differential translations (G_1, G_2, G_3), derivatives of rotation around the each of the standard axes (G_4, G_5, G_6), and the derivative of scale change (G_7):

$$G_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$
$$G_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$
$$G_7 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.1}$$

An element of sim(3) is represented as a linear combination of generators,

$$u_1G_1 + u_2G_2 + u_3G_3 + \omega_1G_4 + \omega_2G_5 + \omega_3G_6 + \lambda G_7 \in \mathrm{sim}(3) \quad \text{where } (\boldsymbol{u};\ \boldsymbol{\omega};\ \lambda) \in \mathbb{R}^7. \tag{4.2}$$

From now on we denote $\boldsymbol{\xi} = (\boldsymbol{u};\ \boldsymbol{\omega};\ \lambda) \in \mathrm{sim}(3)$ for convenience. The Lie algebra is linked to the Lie group by an exponential map which is a bijective function converting any element in the tangent space to exactly one transformation in the group. Specifically, the exponential map from sim(3) to Sim(3) is the matrix exponential given by

$$\begin{aligned} \exp(\boldsymbol{\xi}) &= \exp \begin{bmatrix} \boldsymbol{\omega}^\wedge & \boldsymbol{u} \\ \mathbf{0}^T & -\lambda \end{bmatrix} \\ &= \mathbf{I} + \begin{bmatrix} \boldsymbol{\omega}^\wedge & \boldsymbol{u} \\ \mathbf{0}^T & -\lambda \end{bmatrix} + \frac{1}{2!} \begin{bmatrix} (\boldsymbol{\omega}^\wedge)^2 & \boldsymbol{\omega}^\wedge \boldsymbol{u} - \lambda \boldsymbol{u} \\ \mathbf{0}^T & \lambda^2 \end{bmatrix} + \ldots \\ &= \begin{bmatrix} \exp(\boldsymbol{\omega}^\wedge) & \boldsymbol{v} \\ \mathbf{0}^T & \exp(-\lambda) \end{bmatrix} \in \mathrm{Sim}(3) \end{aligned} \tag{4.3}$$

where $\boldsymbol{\omega}^\wedge$ is the skew-symmetric matrix associated to vector $\boldsymbol{\omega}$, $\exp(\boldsymbol{\omega}^\wedge)$ is the rotation matrix, $\boldsymbol{v}$ denotes the translation component and $\exp(-\lambda)$ is

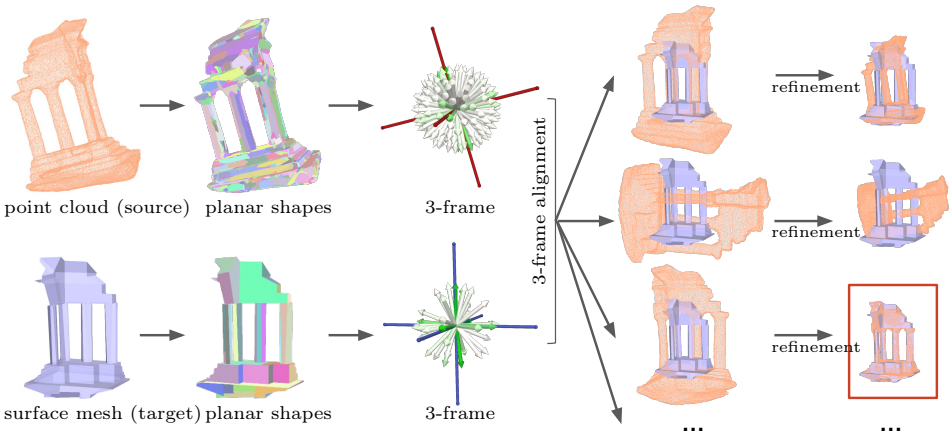


Figure 4.1: Overview of the proposed method. Planar shapes are first extracted from the input point cloud and surface mesh. From the surface-normal distribution of planar shapes (green corresponds to a high portion of planar area with normal pointing towards the arrow direction), three dominant directions are estimated, called a 3-frame. The 3-frames are aligned between the source and the target, leading to 24 possible rotations (only three are represented here). The refinement step takes each candidate rotation and estimate a final similarity transform. The alignment with the minimal loss is kept as the final result (see red frame).

the scale change. For $S \in \mathrm{Sim}(3)$ and a point $\mathbf{p} \in \mathbb{R}^3$, differentiation of $S\mathbf{p}$ by $\boldsymbol{\xi}$ is performed by implicitly left multiplying the transformation by the generators representing infitesimal perturbations.

4.3 Algorithm

We consider as input a pair of 3D data composed of a surface mesh and a point cloud which we denote by the source and the target respectively. The relative scale between them is unknown and the overlap can be partial. The goal is to determine the parameters of a similarity transformation S which best aligns the source against the target,

$$S = \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{4.4}$$

where $s \in \mathbb{R}$, $R \in \mathbb{R}^{3\times 3}$ and $\mathbf{t} \in \mathbb{R}^3$ are the scale factor, the rotation matrix and the translation vector respectively.

The application of distance function representation removes the need for explicitly solving for correspondences. At first glance, it is intuitive to formulate the task as a least squares problem in the same way as rigid registration [Fit01] using the distance field. Let $\{\mathbf{d}_i\}_1^{n_d}$ be a set of n_d points from the source, and $D_m \colon \mathbf{p} \in \mathbb{R}^3 \mapsto d \in \mathbb{R}$ be the distance field of the target surface, which maps a 3D point $\mathbf{p}$ to its Euclidean distance d to the closest point on the surface. Simple adaptation of the rigid registration formulation leads to a loss function given by

$$U(S) = \sum_{i=1}^{n_d} |D_m(S\mathbf{d}_i)|^2 \tag{4.5}$$

where conversion from homogeneous coordinates to Cartesian coordinates is omitted for simplicity of notations. The above formulation, however, has an infinite number of global minima $U = 0$ at scale factor $s = 0$, where the source simply shrinks to a single point on the target. These undesirable global minima result from the difference between Euclidean transformation and similarity transformation.

We propose an improved formulation by considering also the distance field D_d of the underlying surface of the source. Let $\{\mathbf{m}_i\}_1^{n_m}$ denote a set of n_m points sampled from the target. The proposed loss is given by

$$U(S) = \frac{1}{n_d} \sum_{i=1}^{n_d} |D_m(S\mathbf{d}_i)|^2 + \frac{1}{n_m} \sum_{j=1}^{n_m} |sD_d(S^{-1}\mathbf{m}_j)|^2 \tag{4.6}$$

where S^{-1} is the inverse of S. The distance field D_d is rescaled by the scale factor s. The purpose of rescaling is to balance the losses contributed by the two distance fields. The final loss is a symmetric measure of fit between the source and the target, which eliminates undesirable global minima. In order to minimize the proposed loss, we propose a two-step pipeline which consists in a rotational alignment followed by a local refinement. Figure 4.1 shows an overview of our method.

4.3.1 Planar shape based alignment

The first step of our method consists in aligning the orientations of the source and target in a simple yet effective manner. The method generates a set of candidate rotation matrices, which will be refined in the later refinement step. First, a set of planar shapes are detected on the point cloud via region

growing [LM12], with fixed parameters for all experiments. This step helps filtering out noisy points in the point cloud and yields an as clean as possible representation of the actual shapes, similar to the idea of Corsini et al. [CDG+13] who uses Variational Shape Approximation [CSAD04] to partition the point cloud into planar regions. From now on, we will discard the original point cloud and use the clean subset instead.

We propose to initialize the rotation matrix by aligning surface normals of the planar shapes of the source and the target. The alignment of normal vectors is invariant to scale and translation, which offers a more robust estimation. Our approach shares similarity with Stata Center World (SCW) [TBD05] and the Manhattan Frame [SRF+14], which analyze the surface-normal distributions of a single input. In our setup, we focus on the relationship between the surface-normal distributions of the source and target. As shown in Figure 4.1, we first cluster the normal vectors of each set of planar shapes to find 3 major axes (not necessarily orthogonal), which from now on will be called a 3-frame. A 3-frame represents the component means of a weighted-data Gaussian mixture model. In case of the point cloud, the data points are the alpha-shapes of the detected planes, weighted by their areas. In case of the surface mesh, they are the polygonal facets of the mesh, weighted by their areas. The distance metric of data points is defined on the unit sphere, where each axis includes both positive and negative directions. More specifically, a 3-frame can be represented as columns in

$$A = \begin{bmatrix} \mathbf{u}_1 & -\mathbf{u}_1 & \mathbf{u}_2 & -\mathbf{u}_2 & \mathbf{u}_3 & -\mathbf{u}_3 \end{bmatrix}, \ \mathbf{u}_i \in \mathbb{R}^3. \tag{4.7}$$

We use the absolute cosine similarity metric instead of the Euclidean distance for measuring the distance between two normal vectors on the spherical surface, i.e.

$$d(\mathbf{v}_1, \mathbf{v}_2) = \frac{|\mathbf{v}_1 \cdot \mathbf{v}_2|}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|}. \tag{4.8}$$

We use the weighted Expectation-Maximization (EM) algorithm [GAPFH16] to solve for the cluster means and variances. Same is done to the normal vectors of the target surface. The proposed approach is based on the assumption that, for the same underlying scene, there exists some column permutation P such that the 3-frames of different representations are aligned via a rotation R,

$$A_2 = RP(A_1). \tag{4.9}$$

For a given permutation P_i, the rotation matrix is computed as the solution to the orthogonal Procrutes problem

$$\begin{aligned} R_i =& \text{argmin}_{\Omega} \|\Omega P_i(A_1) - A_2\|_F \\ & \text{subject to } \ \Omega^T\Omega = I \text{ and } \det(\Omega) = 1, \end{aligned}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The solution for R is obtained by only allowing orthogonal matrices with determinant 1, and is given by $R_i = U\Sigma' V^T$, where $A_2 P_i(A_1)^T = U\Sigma V^T$ is the singular value decomposition and Σ' is a modified Σ with the smallest singular value replaced by $\det(UV^T)$, and other singular values replaced by 1. The output of the rotational alignment step is a list of rotation matrices $\{R_i\}_{i=1}^{24}$. This is done by aligning the 3-frames with all permutations of the three axes (3! in total) and combinations of the orientation of each axis (2^3 in total). Simple application of the right-hand rule (or left-hand rule) for a consistent orientation of axes can eliminate half of the candidates, giving a final number of 24. Figure 4.2 shows the complete list of rotational initilization.

4.3.2 Refinement

The results of the rotational alignment are now refined using local continuous optimization. For the input point cloud, the detected alpha shapes from the previous step are used to generate its distance field D_d. In order to solve the minimization problem locally, we rewrite the transformation matrix as $S = \exp(\boldsymbol{\xi})$ where $\boldsymbol{\xi} = (\boldsymbol{u};\ \boldsymbol{\omega};\ \lambda) \in \mathbb{R}^7$ is the corresponding element in Lie algebra. We denote $\exp(\boldsymbol{\xi})$ as $S_{\boldsymbol{\xi}}$ from now on, and let $s_{\boldsymbol{\xi}}$ be the associated scale factor. The optimization objective becomes

$$\min_{\boldsymbol{\xi}} U(\boldsymbol{\xi}) = \frac{1}{n_d}\sum_{i=1}^{n_d} |D_m(S_{\boldsymbol{\xi}}\mathbf{d}_i)|^2 + \frac{1}{n_m}\sum_{j=1}^{n_m} |s_{\boldsymbol{\xi}} D_d(S_{\boldsymbol{\xi}}^{-1}\mathbf{m}_j)|^2. \tag{4.10}$$

The derivative of the loss is thus

$$\begin{aligned} \frac{dU}{d\boldsymbol{\xi}} =& \frac{2}{n_d}\sum_{i=1}^{n_d} D_m(S_{\boldsymbol{\xi}}\mathbf{d}_i)\nabla D_m(S_{\boldsymbol{\xi}}\mathbf{d}_i)\frac{dS_{\boldsymbol{\xi}}\mathbf{d}_i}{d\boldsymbol{\xi}} + \\ & \frac{2}{n_m}\sum_{j=1}^{n_m} s_{\boldsymbol{\xi}} D_d(S_{\boldsymbol{\xi}}^{-1}\mathbf{m}_j)\big(\frac{ds_{\boldsymbol{\xi}}}{d\boldsymbol{\xi}} D_d(S_{\boldsymbol{\xi}}^{-1}\mathbf{m}_j) + s_{\boldsymbol{\xi}}\nabla D_d(S_{\boldsymbol{\xi}}^{-1}\mathbf{m}_j)\frac{dS_{\boldsymbol{\xi}}^{-1}\mathbf{m}_j}{d\boldsymbol{\xi}}\big) \end{aligned} \tag{4.11}$$

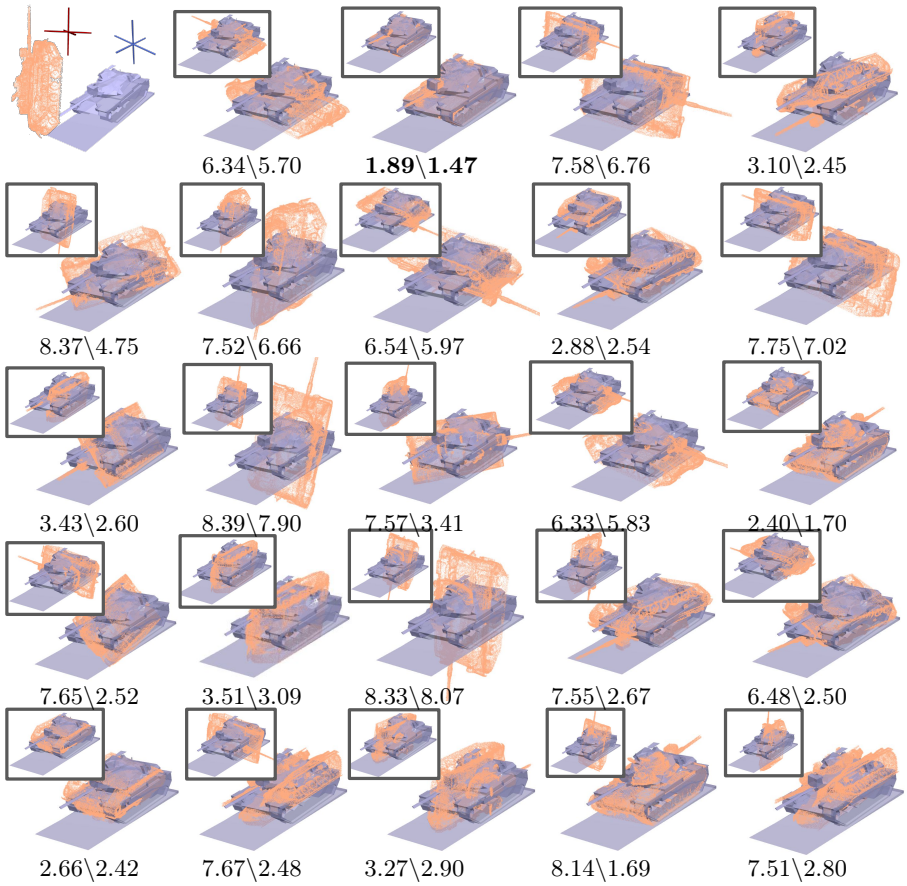


Figure 4.2: Demonstration of the 24 initial rotations and the corresponding output after local refinement. The input point cloud and the mesh are shown on the left, together with their 3-frames (red and blue, respectively). For each initial rotation, the alignment after applying the rotation matrix is shown on the top-left frame, and the final refinement output is shown at the bottom right. The RMSE ($\times 10^{-2}$) values before and after refinement are shown below. The best path is indicated by a red arrow.

where $\nabla D_m(\mathbf{p})$ is the gradient vector of the distance field at point $\mathbf{p}$. We have, for any point,

$$\frac{dS_{\boldsymbol{\xi}}\mathbf{p}}{d\boldsymbol{\xi}} = \begin{bmatrix} I & -\mathbf{q}'^{\wedge} & \mathbf{q}' \\ \mathbf{0}^T & \mathbf{0}^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times 7}, \tag{4.12}$$

$$\frac{dS_{\boldsymbol{\xi}}^{-1}\mathbf{p}}{d\boldsymbol{\xi}} = \frac{dS_{-\boldsymbol{\xi}}\mathbf{p}}{d\boldsymbol{\xi}}, \tag{4.13}$$

$$\frac{ds_{\boldsymbol{\xi}}}{d\boldsymbol{\xi}} = \begin{bmatrix} \mathbf{0}^T & \exp(\lambda) \end{bmatrix} \in \mathbb{R}^{1\times 7} \tag{4.14}$$

where $\mathbf{q}'$ denotes the Cartesian representation of the homogeneous coordinates $S_{\boldsymbol{\xi}}\mathbf{p}$, and $\mathbf{q}'^{\wedge}$ is the skew-symmetric matrix associated with vector $\mathbf{q}'$. Note that $S_{-\boldsymbol{\xi}} = \exp(-\boldsymbol{\xi}) = S_{\boldsymbol{\xi}}^{-1}$. Trust region methods, such as Levenberg-Marquardt and Dogleg, can be used for optimization. In our experiments, we use the Dogleg algorithm.

In our implementation, note that AABB tree is used for fast distance queries against sets of plane objects. Each query returns a closest point $\mathbf{p}$ on the set of planes to the query point $\mathbf{q}$. It also allows efficient computation of distance field gradient, as the gradient of a distance field always has magnitude 1 and has the same direction as $\mathbf{q} - \mathbf{p}$ whenever only one closest point exists.

4.4 Experiments

Our algorithm is implemented in C++ using the Computational Geometry Algorithms Library (CGAL) [The21] and the Ceres library [AMO]. For all experiments, the parameters of the region growing step for shape detection are fixed and set as follows: The Euclidean distance threshold is set to 4μ, where μ is the mean of K nearest neighbor distances of the point cloud. The normal threshold is set to 35 degrees. The minimum number of points per planar shape is 40.

4.4.1 Dataset and error metrics

Dataset. Existing datasets for rigid registration consist of point cloud pairs obtained from the same acquisition modality, which does not offer differences in terms of levels of detail and defects. Also, there is often no associated mesh data of the captured scene. Synthetic range data from meshes do not simulate well defects of real-world acquisition systems. To this end, we evaluate and compare our approach with state-of-the-art methods on a collection of 13 real-world point sets that differ in terms of shape complexity, size, and acquisition characteristics, provided in [BL20], together with their corresponding simplified 3D models. The point sets are acquired via different modalities, i.e. multi-view stereo, and laser scanner. Additionally, we include synthetic data consisting of point sets sampled from 6 shapes, and their simplified models computed using [BL20]. The simplified 3D models are

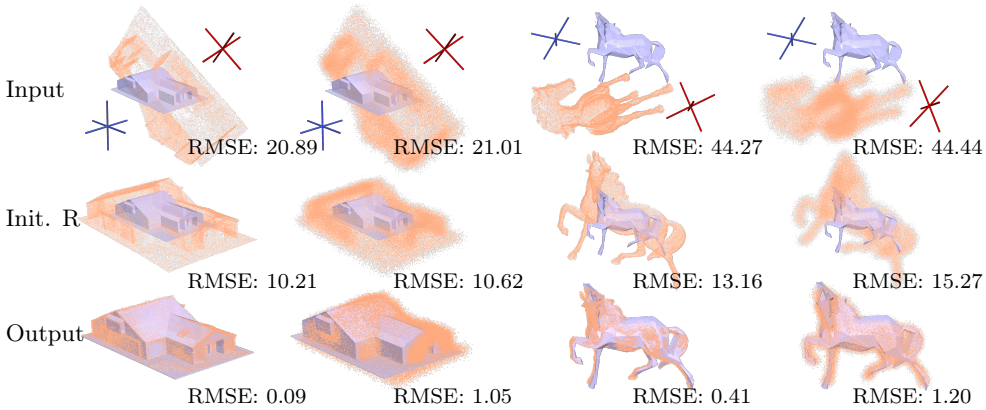


Figure 4.3: Visualization of our registration results on a regular object (barn) and a free-form object (horse). For each object, Gaussian noise is added to create a noisy version. The top row shows the mesh and the point cloud to be aligned, as well as their 3-frames (drawn in blue and red, respectively). The best initial rotation is shown in the middle row, with the aligned result at bottom. The RMSE ($\times 10^{-2}$) values are shown.

compact mesh representations of the objects, which differ from the point sets in terms of detail, noise and outliers. The models are set to different scales in the experiments. The dataset is divided into two categories: free-form objects and regular objects. We consider an object to be regular if it exhibits a high degree of organization in the form of large planar structures, e.g. buildings and furniture. The rest are considered as free-form objects. Figure 4.4 visualizes our results on all 19 pairs of objects in the dataset.

Metrics. We use two metrics for quantitative evaluation: root mean square error (RMSE) and α-recall. We compute the RMSE between the estimated transformation $S = (s, R, t)$ from the ground-truth transformation $S^* = (s^*, R^*, t^*)$:

$$\varepsilon = \sqrt{\frac{1}{n_d} \sum_{i=1}^{n_d} \min_j \|sR\mathbf{d}_i + t - s^* R^* \mathbf{d}_j - t^*\|_2^2} \tag{4.15}$$

It is worth noting that, the distance is computed against the nearest neighbor in the ground-truth. Unlike previous works, we do not directly compare against the ground-truth transformation nor on the distance between ground-truth correspondences in order to eliminate ambiguity for shapes

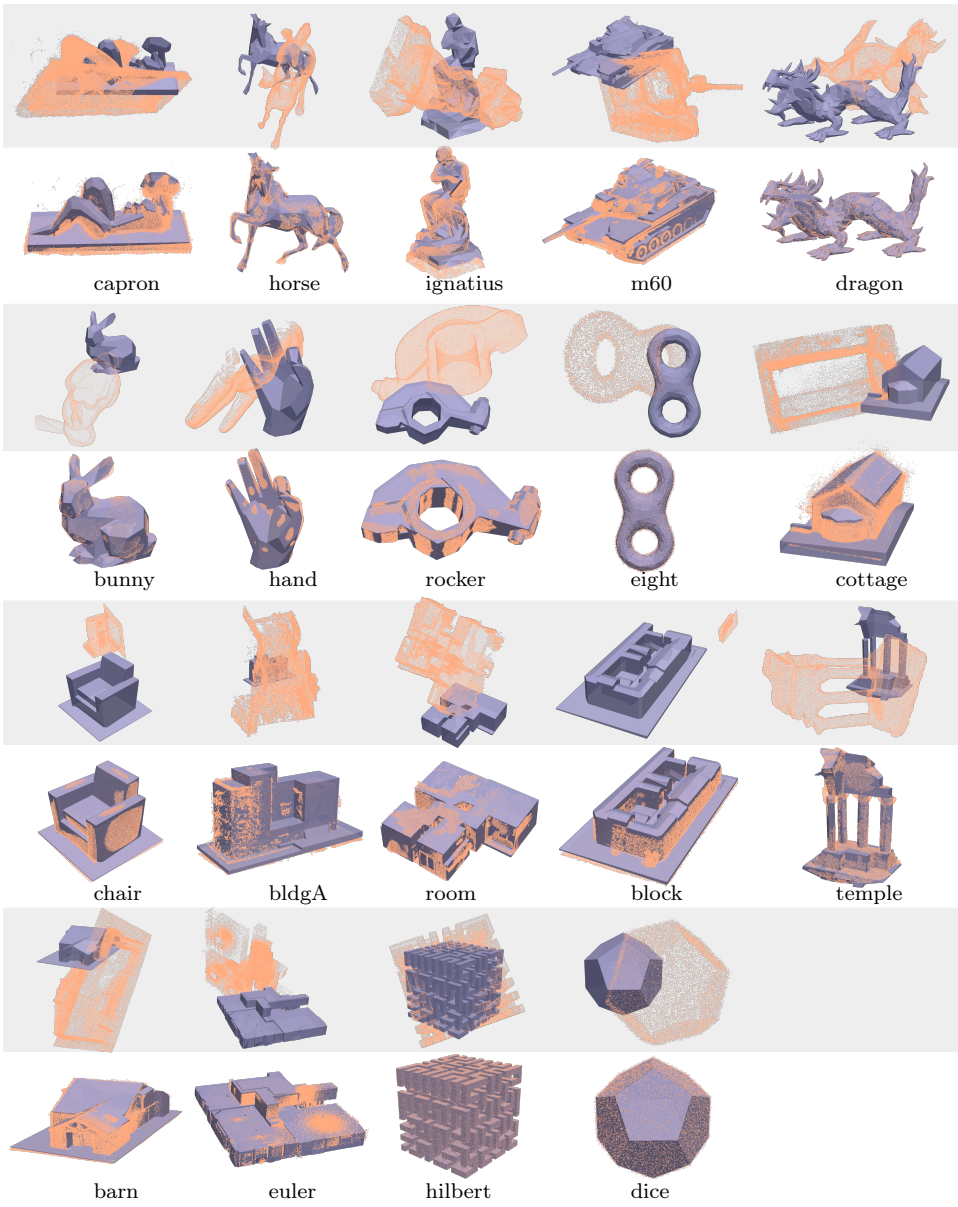
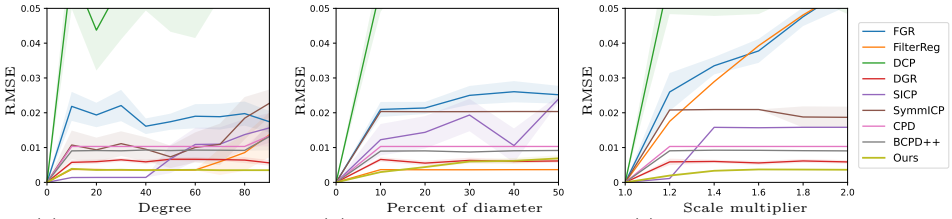


Figure 4.4: Visualization of our results on 19 examples from the dataset, with random perturbations in rotation between 90° and180 °, in translation between 0 and 100% of the diameter, and in scaling between 0.25 and 2. Regular objects are shown after free-form objects. For each example, the input point cloud and mesh are shown on the top (gray region), with the alignment output at the bottom.

with rotational symmetry, where multiple ground truths exist. α-recall is defined as the ratio of successful pairwise registrations, where a registration is considered successful if its RMSE is smaller than a certain threshold α [ZPK16]. For both metrics, the RMSE unit is the diameter of the target.



(a) Rotation perturbation (b) Translation perturbation (c) Scale perturbation

Figure 4.5: The mean (bold curve) and standard deviation (shaded region) of the RMSE of each method on different perturbations to the ground-truth alignment. Lower is better.

4.4.2 Robustness

Figure 4.3 shows our registration results for regular and free-form objects. To demonstrate robustness to noise, 3D Gaussian noise is added to the MVS point cloud of each object (2nd and 4th columns). Comparing the 3-frames of point clouds with and without added noise, it can be seen that estimation of 3-frames is robust to noise, especially for the case of regular objects. Thus our method is able to generate good initial rotations, which are refined to recover the final alignment.

We also investigate robustness to each type of perturbation (rotation, translation and scale). The results are shown in Figure 4.5. In the first two experiments, the source is perturbed from the ground-truth alignment with varying degrees of rotation (or translation, resp.), keeping the ground-truth translation (or rotation, resp.) and scale. In the third experiment, the source is resized by a specific amount each time and undergoes randomly generated small rotations and translations. As illustrated in Figure 4.5, our algorithm show competitive stability to all three types of perturbations.

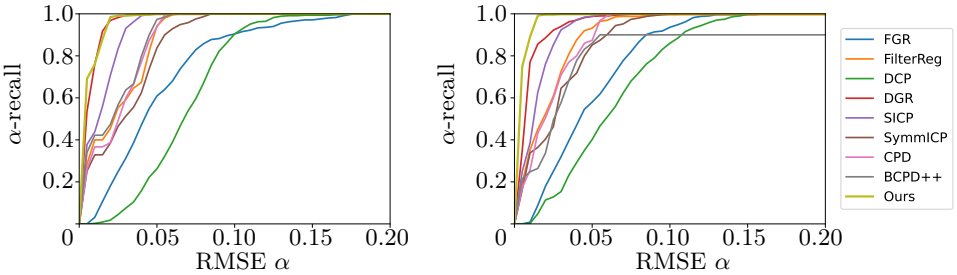
4.4.3 Comparisons

We compare with both local and global methods. The local methods include SICP [ZSN05], SymmICP [Rus19], CPD [MS10], BCPD++ [Hir20a] and Fil-

terReg [GT19], while the global methods are FGR [ZPK16], DCP [WS19a], and DGR [CDK20]. In our experiment, all local methods are combined with a RANSAC initialization. Some methods have a built-in scale estimation component and are labeled as *joint* in Table 4.1. The others assume a given scale as input and are labeled as *two-step*. For these two-step methods, we provide an estimated scale factor from a bounding box based estimation method which can provide a good estimate given sufficient overlap between input [CDG+13, MAM14]. For all data tested in Table 4.1, the error is around 2.5% from the ground truth scale. Details of the scale estimation method can be found in appendix A.3.

Visual comparison. Figure 4.7 visualizes the registration results of each method on two types of examples. The input point clouds are significantly perturbed from the ground-truth alignment. For local methods, we show both results with and without an FPFH-based RANSAC initialization. Classical feature-based global methods like FGR do not offer enough robustness to variations in the shape characteristics. On the free-form object (top), the family of probabilistic approaches (CPD, BCPD++, and FilterReg) benefits more from the RANSAC initialization than the ICP approaches. Point-based learning method DCP fails to register point clouds with large numbers of points, due to its high memory consumption. Volumetric learning-based approach DGR is able to produce more accurate correspondences, but requires accurate estimation of the relative scale as a pre-processing step. Our method recovers well the alignment for both types of objects, achieving satisfying accuracy.

Quantitative comparison. Our algorithm performs best on regular objects and scenes as they are well described by piecewise-planar geometry. As shown in the right part of Table 4.1, our method reaches significantly lower average RMSE in 8 out of 10 objects, while retaining errors reasonably close to the best baseline in the remaining 2 cases. In addition, our method achieves the lowest maximal RMSE for almost half of the tested objects, exhibiting a low failure rate comparable to the others. On the contrary, the best performing baseline, DGR, is less robust as its maximal RMSE tends to be off by a large amount when it fails, e.g. on cottage, hilbert and dice. Figure 4.6 (b) shows the α-recall rate of all methods on all tests done on the



(a) Free-form objects (b) Regular objects

Figure 4.6: α-recall curve of each method on free-form objects (a) and regular objects (b). In particular, our approach outperforms existing methods on regular objects, and achieves competitive results as compared to the best baselines on free-form objects.

free-form objects. Our algorithm achieves a 0.02-recall of 99%, significantly higher than the other algorithms, with DGR reaching 88%. For free-form objects, as indicated by the left part of Table 4.1 as well as Figure 4.6 (a), our method matches the accuracy achieved by state-of-the-art methods.

4.4.4 Performances

The planar shape-based alignment typically requires a few seconds to one minute depending on the size of the input point cloud (that ranges from 150K to 3M points). This corresponds to the processing time for detecting planar shapes, clustering being negligible. The refinement step is also a few seconds for each rotational initialization from our non-optimized sequential implementation of the algorithm.

4.4.5 Limitations

Our algorithm, which is designed to perform on regular scenes, is less competitive on free-form objects. The detection of planar shapes on such objects often gives a rough and arbitrary approximation of their curved surfaces. Our method is also not designed to the registration of 3D data with a very low overlap ratio.

4.5 Conclusion

We proposed a global registration algorithm for multi-modal geometric data which differs in terms of noise, detail, and scales. Our algorithm performs a planar shape based alignment to recover candidate rotations independent of scale and translation, followed by a refinement step with a local continuous optimization. We demonstrated the robustness and efficacy of our algorithm on defect-laden real-world data, as well as it competitiveness against state-of-the-art methods, especially on objects and scenes that can be described with a piece-wise planar geometry.

Table 4.1: Quantitative comparisons. Average (and maximal) RMSE ($\times 10^{-2}$) is computed over 50 random perturbations on scaling (between 0.25 and 4), rotation (between 60° and 180°), and translation (between 0 and 100% of the diameter) for each of the 19 models.

		free-form objects									regular objects									
		capron	horse	ignatius	m60	dragon	bunny	hand	rocker	eight	cottage	chair	bldgA	room	block	temple	barn	euler	hilbert	dice
two-step	FGR[ZPK16]	3.532	3.509	3.829	4.173	1.280	6.992	5.358	5.169	11.762	4.999	6.482	8.437	5.099	7.873	4.012	3.414	3.213	3.443	1.435
		(9.471)	(8.427)	(9.266)	(7.949)	(2.415)	(8.472)	(10.280)	(12.768)	(17.103)	(8.307)	(8.502)	(13.041)	(8.194)	(15.813)	(11.484)	(7.434)	(6.776)	(4.832)	(4.261)
	FilterReg[GT19]	1.680	3.250	2.568	1.732	2.696	3.591	2.861	2.568	0.720	1.695	3.627	2.625	2.007	42.352	3.770	1.832	1.566	0.853	1.048
		(2.576)	(5.232)	(5.297)	(2.873)	(5.374)	(6.153)	(4.583)	(4.245)	(0.735)	(2.907)	(6.125)	(4.348)	(3.286)	(>100)	(9.126)	(3.236)	(2.068)	(3.228)	(1.537)
	DCP[WS19a]	6.439	9.028	5.174	6.977	5.985	6.933	5.098	8.666	7.095	4.362	6.200	8.902	4.919	9.509	8.966	8.294	6.551	3.088	1.410
		(9.791)	(13.373)	(10.540)	(9.148)	(10.837)	(9.761)	(7.834)	(11.323)	(16.750)	(6.875)	(7.757)	(13.061)	(8.367)	(15.283)	(14.689)	(12.157)	(10.537)	(4.149)	(1.561)
	DGR[CDK20]	1.377	0.339	0.259	**0.718**	0.247	0.498	0.265	1.107	1.414	1.738	0.770	**0.679**	1.190	**0.370**	0.372	0.154	0.529	3.025	0.758
		(2.129)	(0.449)	**(0.418)**	**(1.084)**	(0.400)	**(0.569)**	**(0.488)**	(3.651)	(8.287)	(8.294)	(1.147)	**(0.860)**	(4.120)	**(0.511)**	**(0.506)**	**(0.255)**	(1.188)	(7.451)	(1.149)
joint	SICP[ZSN05]	**1.059**	**0.096**	1.644	1.231	0.045	1.937	1.837	2.075	1.674	0.495	2.322	2.583	1.723	1.409	1.560	0.696	1.024	1.345	0.786
		(1.904)	**(0.122)**	(3.024)	(3.203)	**(0.063)**	(4.591)	(3.665)	(3.288)	(2.745)	(2.578)	(5.192)	(6.738)	(3.114)	(1.962)	(2.891)	(3.164)	(1.409)	(1.437)	(1.514)
	SymmICP[Rus19]	2.344	3.699	2.618	2.053	3.176	4.889	3.374	3.097	0.421	2.022	4.685	3.582	2.714	1.755	5.152	3.519	1.887	0.864	0.475
		(3.055)	(5.800)	(6.722)	(4.124)	(5.670)	(8.530)	(5.807)	(5.200)	(0.575)	(4.533)	(7.238)	(7.535)	(4.651)	(2.783)	(10.979)	(6.157)	(3.008)	(2.651)	(0.488)
	CPD[MS10]	1.723	2.670	2.615	3.186	2.299	3.224	2.970	2.180	0.405	1.967	4.981	3.031	2.223	1.349	3.055	1.740	1.829	1.221	1.100
		(2.352)	(4.334)	(5.344)	(3.825)	(5.429)	(5.888)	(4.610)	(3.680)	(0.575)	(3.089)	(6.287)	(4.232)	(5.971)	(2.557)	(5.275)	(3.112)	(2.367)	(1.316)	(1.632)
	BCPD++[Hir20a]	1.568	2.862	2.581	1.522	2.196	2.851	2.728	2.169	0.435	2.086	3.201	2.595	2.039	1.544	3.128	1.807	1.710	2.363	21.311
		(2.609)	(5.087)	(5.577)	(2.970)	(4.387)	(5.136)	(4.405)	(3.970)	(0.592)	(3.688)	(4.495)	(3.942)	(4.211)	(2.851)	(5.493)	(3.393)	(2.588)	(2.406)	(21.321)
	Ours	1.338	0.383	**0.136**	1.720	**0.042**	**0.314**	**0.170**	**0.095**	**0.314**	**0.427**	**0.268**	1.188	**0.348**	0.920	**0.170**	**0.079**	**0.253**	**0.039**	**0.487**
		(1.842)	(5.035)	(0.941)	(2.258)	(1.332)	(4.000)	(2.144)	**(1.300)**	**(0.562)**	**(1.211)**	**(0.696)**	(5.417)	**(1.047)**	(1.316)	(0.850)	(0.286)	**(0.742)**	**(0.513)**	**(0.504)**

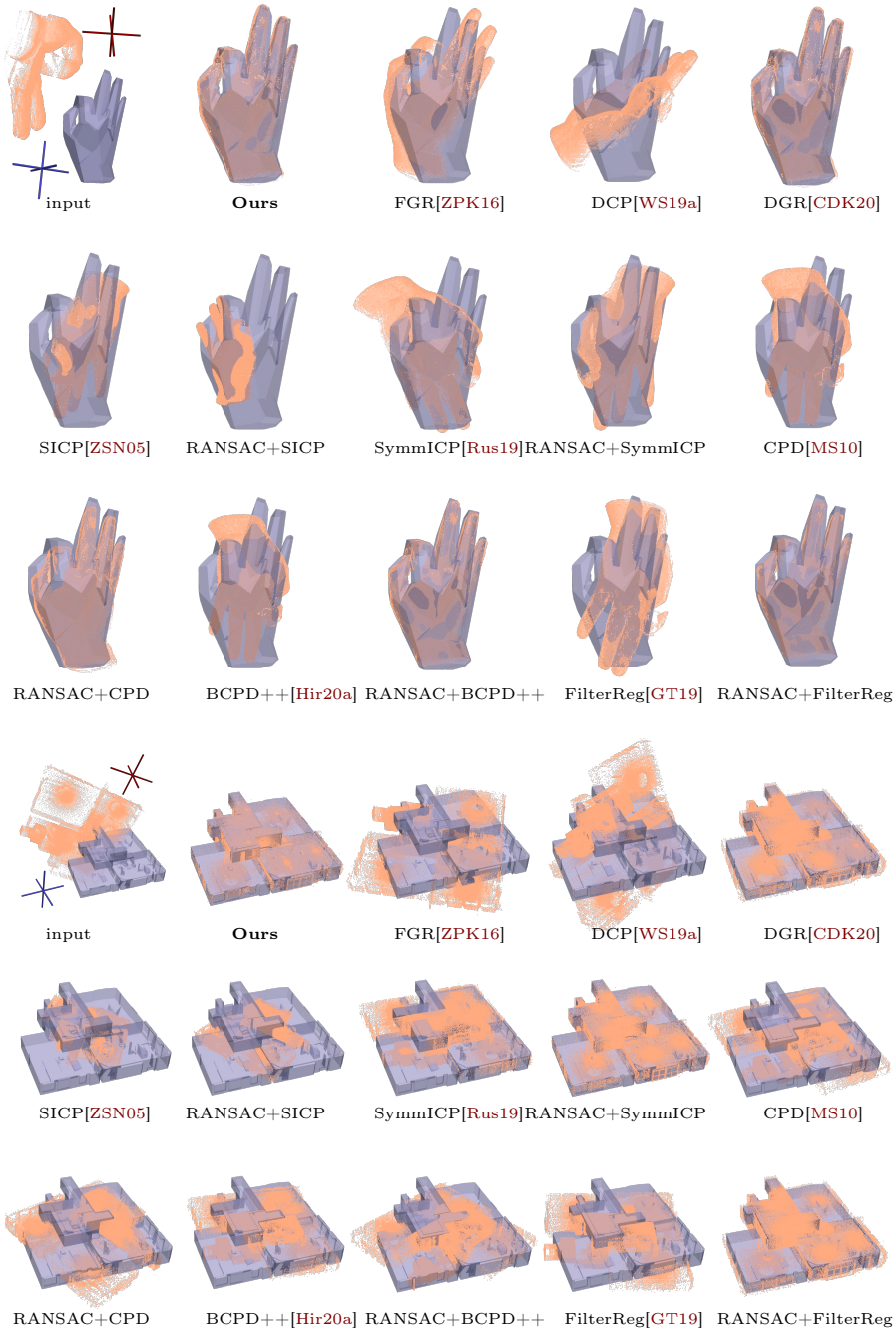


Figure 4.7: Visualization of results registered by each method on a free-form object (top) and a regular object (bottom). The 3-frames of the input point cloud and the mesh are shown (red and blue, respectively). Our method achieves satisfying alignment for both types of objects under large transformation.

Chapter 5

Application to floor modeling

In this chapter, we show a potential application of our algorithms presented in the previous two chapters, by combining them into a vectorization-then-registration pipeline for generating a compact model for building interiors. While existing geometric modeling approaches mainly focus on generating a 3D model of the target scene from a single type of data (e.g. RGB-D scans [DST+21], a point cloud [STM+21], a single image [RPJT13] or panorama [YWP+19], or multi-view images [CF14]), the proposed pipeline explores the possibility of utilizing multi-modal data in the intermediate steps.

5.1 Principle

Our pipeline receives as input a floor plan image, and a multi-view image sequence of the building interior. The output is a 2.5D floor model with cameras aligned. An overview of the pipeline is shown in Figure 5.1. We assume that the point cloud (and camera parameters relative to the point cloud) is already computed by off-the-shelf reconstruction systems such as COLMAP [SF16, SZPF16], as the reconstruction of point cloud is not the focus of this thesis. Now we detail main components that constitute the pipeline.

First, our polygonal image segmentation algorithm presented in chapter 3 can be used to vectorize floor map images. Existing pipeline for this application usually requires the use of specialized methods to detect, filter and connect corner points into a floor map [LWKF17]. In contrast, our algorithm finely approximates linear structures with polygons. The input is a rasterized floor plan. The probability map is estimated using color cues in the floor plan. For a grayscale floor plan where walls are typically black, the probability map is simply computed by inverting and rescaling the pixel values to the $[0, 1]$ range. Erosion and dilation are used to remove thin lines representing doors and windows. For a color floor plan, a naive two-

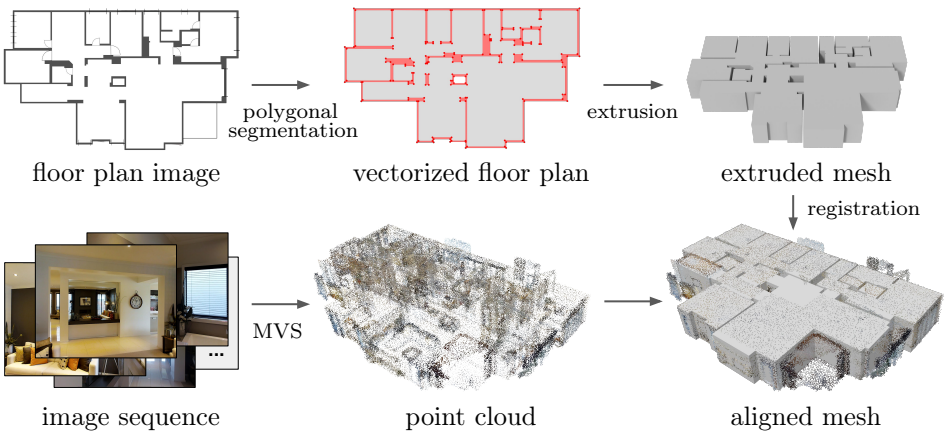


Figure 5.1: Overview of our pipeline. The floor plan image is first converted to a vectorized form, where the interior region is indicated in gray. The 2.5D model is generated by extruding polygons corresponding to the interior region given an estimated height. Finally, the model is registered to the point cloud, computed from the image sequence, to output an aligned model.

step method is used for estimation of the probability map. First, the pixel colors are clustered by k-means clustering, given a user-specified number of desirable clusters. In our experiment, the number of clusters is set to 20. For each of the resulting clusters, a class label (wall, room type 1, room type 2, other etc) is assigned by the user. Second, following a similar idea of [FLBA20], the probability of a pixel i belonging to class m_f is assigned as its normalized RGB distance to the closest color in the set of cluster centers that corresponds to m_f, that is,

$$P(i|m_f) = \frac{\min_{j \in S_{m_f}} \|I(i) - C_j\|}{\min_{j \in S_{m_f}} \|I(i) - C_j\| + \min_{j \notin S_{m_f}} \|I(i) - C_j\|} \tag{5.1}$$

where $I(i)$ is the color at pixel i, S_{m_f} is the set of clusters labeled as m_f, and C_j is the center of cluster j. Note that more sophisticated approaches could be used for predicting the probability map, but are beyond the scope of this thesis.

Our algorithm is capable of processing floor plans of different styles and complexity, as demonstrated in Figure 5.2. In contrast to the previous method [LWKF17] which relies on integer programming for recovering a geometrically consistent result, our iterative approach is scalable for complex floor plans and captures well the detailed structures. Figure 5.3 shows

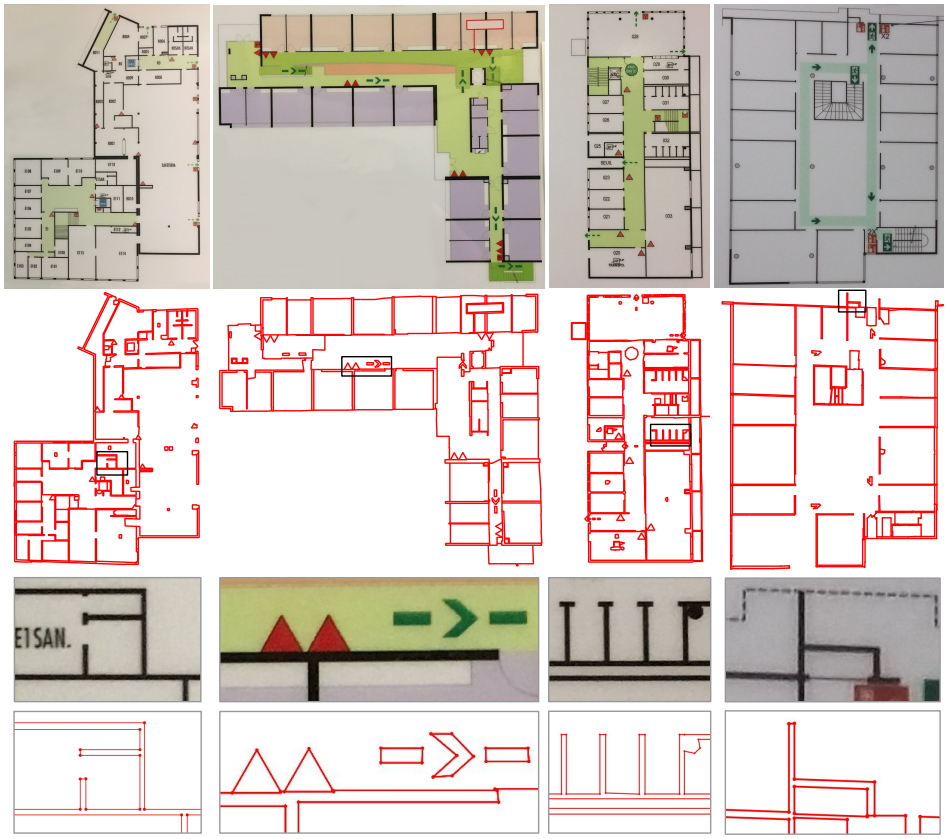


Figure 5.2: Vectorization of floor plans. The first row shows the input floor plan photographs, and the second row shows the vectorized output. Our algorithm captures well the thin linear structures by compact polygons (see closeups in the last two rows).

an example of generating a 2.5D model from the vectorized output by simply extruding wall polygons. Alternatively, we could extrude the interior polygons representing the rooms.

The second step is the alignment of the point cloud with respect to the vectorized floor plan. In contrast to existing geometric modeling pipelines which directly estimates a surface mesh from the point cloud, we are able to generate a compact 2.5D representation of the building interior, thanks to the floor plan. The model is created from the vectorized floor plan by extruding the interior region. It is done by inserting boundaries (possibly nested) into a constrained Delaunay triangulation and marking the triangle facets

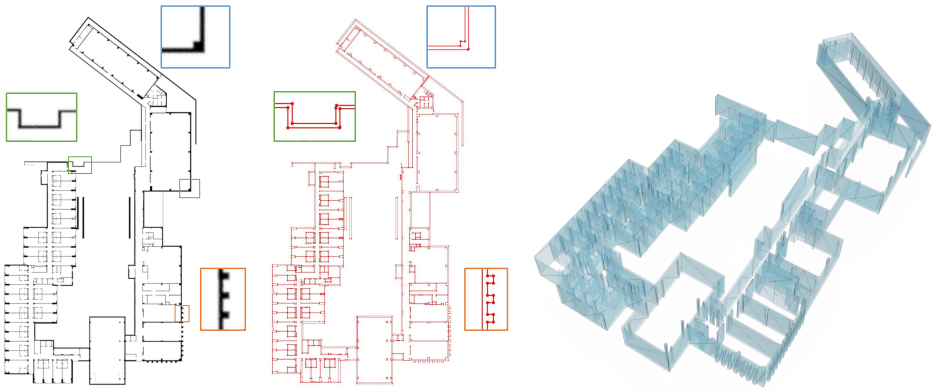


Figure 5.3: Illustration of generating a 2.5D model from a rasterized floor plan. The rasterized floor plan is first converted into a vectorized form by our algorithm. The mesh is created by extruding wall polygons from the vectorized floor plan.

of the region of interest, followed by an extrusion step. A height value is provided to the extrusion algorithm as a ratio against the diagonal of the 2D bounding box of the vectorized floor plan. The height can be user-specified, or estimated from the oriented bounding box of the point cloud. In our experiments, we estimated the height by computing the oriented bounding box of the point cloud after first filtering out noise and outliers. The mesh generated by extruding the interior region is a more accurate approximation to the point cloud than the one created by extruding wall polygons, as the latter contains no part corresponding to the floor or the ceiling. The registration step is done by the multi-modal registration algorithm presented in chapter 4. The algorithm is suited for registering point clouds to compact mesh models for buildings and indoor scenes, as both are well described by planar shapes.

We color the aligned model for visualization purposes. For a point on the mesh, the color is simply determined by the color of the nearest point in the point cloud, constraint by a distance threshold. The surface regions with no neighboring points inside the threshold sphere are painted black. Note that more advanced texturing method [WMG14] could be deployed to create a better texture map.

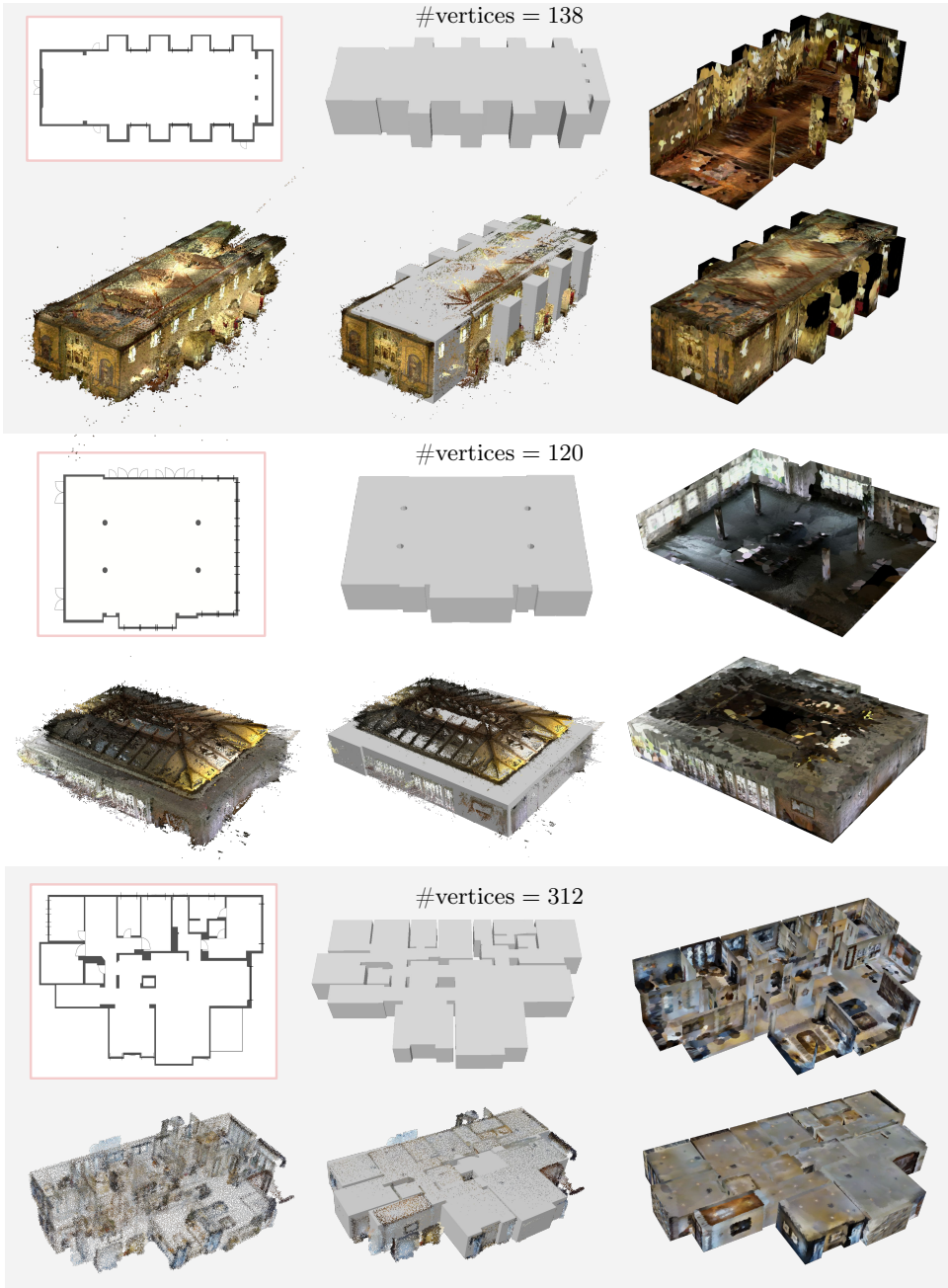


Figure 5.4: Results of our pipeline for building interiors. For each scene, the first column shows the input consisting of a rasterized floor plan and a point cloud. The second column visualizes the 2.5D floor model by extruding the vectorized floor plan, and the alignment result to the point cloud. The last column is the visualization of the aligned floor model.

5.2 Experiments

We demonstrate our pipeline on example scenes from the Tanks and Temples dataset [KPZK17], and the Matterport3D dataset [CDF+17].

Our method produces visually satisfactory results for simple floors where the ground truth can be well approximated by a 2.5D representation, such as the bottom examples in Figure 5.4. In the more complex case where the floor contains rooms with different heights or has a non-trivial ceiling structure, texturing the inconsistent region of the 2.5D model becomes challenging. For instance, the top example in Figure 5.4 contains side rooms which are lower than the central part of the floor, and the middle example in Figure 5.4 has a complicated ceiling which is in fact the interior part of the roof.

The proposed pipeline provides a compact model for the floor. Our output typically consists of a few dozens to a few hundreds of vertices, as indicated in Figure 5.4 and Figure 5.6, whereas Poisson reconstructed meshes could contain hundreds of thousands of vertices.

We test the impact of defects in the input point cloud, by simulating random noise and incomplete acquisition in the scene. As shown in Figure 5.5, the registration step is more robust to random noise on point locations, but tends to fail in case of low overlap between the input pair. In contrary, the simple coloring strategy is able to produce reasonable output in the case of incomplete point cloud, given sufficient amount of overlap with the mesh. It is more sensitive to noisy point clouds, resulting in undesirable artifacts in the texture even in the presence of a relatively low amount of noise. Using more sophisticated texturing techniques, such as methods based on registered raster images, could be the way to address the issue.

Our pipeline is also applicable to building exteriors. An illustrative example is shown in Figure 5.6. Ideally the overhead image should be an orthophoto of the building, however, due to the lack of data we are limited to the floor plan as input. As compared to the indoor case, the reconstructed point clouds from outdoor scenes may contain certain building structures that are absent from the floor plan. For instance, the roof structure cannot be inferred from the floor plan.

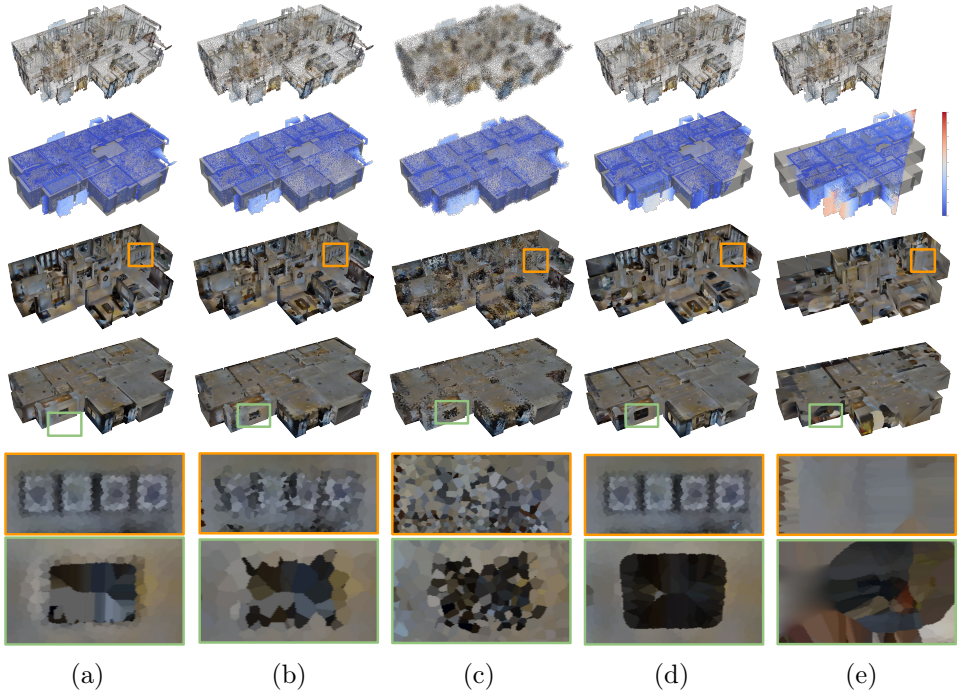


Figure 5.5: Impact of defects in the point cloud. (a) Original input. (b) Points perturbed by noise with std = 0.3% of the mesh diameter. (c) Points perturbed by noise with std = 1% of the mesh diameter. (d) Partial point cloud with 90% overlap. (e) Partial point cloud with 70% overlap. From top to bottom: input point cloud, alignment result with color-coded point-wise distance from the point cloud to the mesh, output model (inner view), output model (outer view), closeups.

5.3 Limitations

While simple grayscale floor plans can be vectorized in a fully automatic fashion, the vectorization of color floor plans still requires human in the loop (for assigning class labels to clusters). The quality of the vectorized floor plan is reliant on the accuracy of the input probability map. In particular, our naive approach for estimating the probability map is limited to floor plans with strong color cues.

The quality of the reconstructed floor model is affected by the presence of rooms with heterogeneous heights, and by the existence of sloped ceilings in the scene. One possible way of addressing this issue is to estimate a height

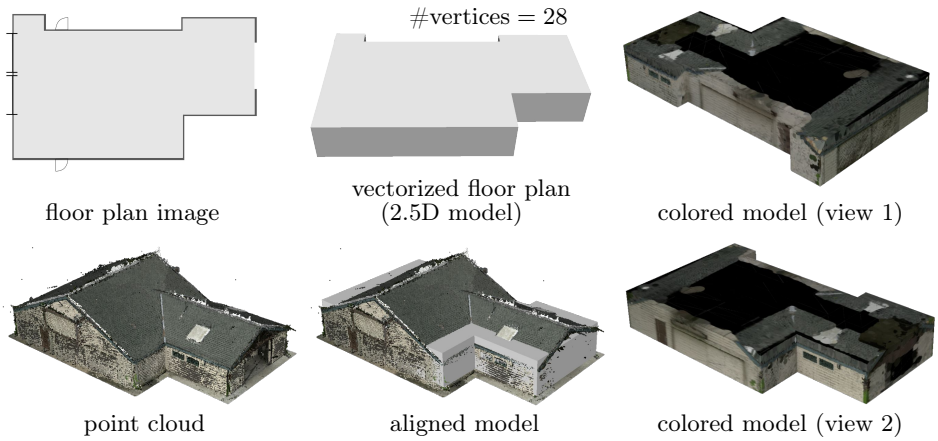


Figure 5.6: Result of our pipeline for a building exterior.

map for the floor. Such a height map can be utilized to guide the mesh extrusion step for creating models with more complex styles.

When extended to building exteriors, the absence of roof structures in the floor plan results in discrepancy between the extruded model and the point cloud. This indicates the need of adding to the pipeline a roof generation step, which can be realized by the weighted straight skeleton algorithm [BHH+15] for polyhedral roofs.

Chapter 6

Conclusion and perspectives

6.1 Conclusion

In this thesis, we investigated the problem of geometric approximation of urban objects from images in the form of compact and accurate meshes. We presented two generic algorithms, one for approximation of 2D object contours with floating polygons, another for 3D alignment of multi-modal geometric data. We also demonstrated the applicative potential of these algorithms by incorporating them into a pipeline for generating textured model for indoor scenes.

The key contribution of this thesis is the novel way of reasoning at the scale of geometric primitives for refining polygonal partitions of 2D domains to improve the quality of the object contouring results, and for robust estimation of the rotational transformation between the pair of input geometry. In chapter 3, we developed an iterative algorithm for refining image partitions through a sequence of merging and splitting operations that correct oversegmentation of objects, and misdetection or absence of primitives. Experiments on images demonstrate that our merging-and-splitting refinement strategy offers reduced segmentation errors, as well as more accurate object contours as compared to vectorization and cell grouping baselines. In chapter 4, we proposed to represent the surface-normal distributions of detected planar primitives with a mixture model, from which a set of hypotheses for the rotational alignment are produced and later refined to output a final transformation. Experiments on challenging real-world data demonstrate the robustness and efficacy of our algorithm. Our method is shown to compete well against existing feature-based approaches, especially on piece-wise planar objects and scenes. The use of geometric primitives also brings computational efficiency. Our algorithms are capable of processing megapixel images and large-scale point clouds with millions of points in a few minutes.

Secondary contributions of our work include the design of an energy func-

tion that measures the quality of polygonal partitions for guiding a discrete optimization scheme, and a continuous optimization formulation for accurate estimation of similarity transformations. For the polygonal segmentation algorithm, our proposed energy takes into account both the fidelity to the input data, by encoding semantic penalization on facets and image gradient penalization on edges, and the complexity of the output polygons. The design of the energy as the sum of local fidelity terms and a global complexity term makes it computationally efficient to evaluate its variations when local operations are applied. For the 3D registration algorithm, we designed a symmetric energy for avoiding undesirable global minima incurred due to object scale variations. Thus we could formulate an optimization problem that jointly updates rotation, translation and scale. Representing similarity transformation as elements in the Lie algebra, gradient based optimization can be used for minimizing the energy.

We also designed kinetic data structures for the partition of 2D domains with non-trivial boundaries, i.e. domains bounded by nested polygons, instead of a simple rectangle. The resulting partition supports complex output including nested polygons. Kinetic data structures for space partitioning offer fast computation and flexible termination criteria, as opposed to exhaustive approaches.

Our algorithms are applicable to different types of scenes, from organic shapes to man-made objects. In particular, we demonstrated a potential application in modeling of indoor scenes by introducing a vectorization-then-registration pipeline. Provided proper semantic information (e.g. by pretrained neural networks for semantic segmentation) and a point set reconstruction method, our pipeline achieves a high level of automation, requiring only a few user-specified parameters.

6.2 Perspectives

This thesis constitutes a tiny step towards the automation of object modeling with compact representations. Our work suffers from some weaknesses. Specifically, the accuracy of our algorithms significantly depend on how well the primitives approximate the underlying object. The quality of meshes generated by our system is still far from the ones created by human experts. User interactions are still required for vectorization of challenging

input (color floor plans without input semantic information). Now we discuss some perspectives that could be explored for future research.

Control of shape complexity. Our polygonal segmentation algorithm provides a parameter that implicitly affects the number of output vertices. Direct user control of the complexity of output shapes is desirable in some practical situations. One way could be to design an additional operator that removes and adds relevant vertices in the partition.

Refining a mesh to observations. As mentioned before, our current extrusion strategy is limited to produce 2.5D building models with uniform height. Several directions could be considered for generating refined meshes. The most straightforward improvement could be to introduce height map prediction for creating meshes with non-uniform height across the top surface. It can be achieved in conjunction with the 3D registration step in an iterative manner, where we could alternate between point-cloud-to-mesh alignment given the current mesh and height map prediction given the current alignment. Another possible extension to our system is the modeling of building components other than walls and floors. Roof model generation, for instance, is an important component for lifting the 2.5D representation into a realistic 3D building model. Finally, recent advances in differential rendering techniques open a new way for fitting a mesh to visual observations. Vertices of the mesh, once roughly registered to input images, can be relocated to optimize the visual consistency between the geometry and photometric information.

Generalization to free-form shapes. A natural extension of this work would be to support more complex objects including curved shapes. Our algorithms focus on linear geometric primitives, i.e. lines and planes, which describe well man-made structures. The introduction of Bezier cycles (polygons where two successive vertices are connected by a Bezier curve) or NURBS elements would bring more versatility to the shape representation. Detecting different types of primitives and designing a unified framework for analyzing their geometric properties and relationships is the key challenge to be addressed. These techniques would allow us to capture free-form shapes with a better complexity-distortion trade-off.

Generalization to incomplete data. Our system is proposed under the assumption that the object of interest is well captured by the set of images. In particular, our registration algorithm assumes a sufficient amount of overlap between the input pair of geometry. However, constraint by the data acquisition techniques, sometimes only a part of the object is available, e.g. an incomplete point cloud reconstruction. Adapting our algorithm to the partial-to-partial alignment case will open up new application fields for our pipeline. One way could be to design a confidence estimation method for weighing each data point. The weight can be assigned according to the likelihood of having the point also contained in the other input.

Attribute transfer. In this thesis, we presented a pipeline that generates a compact textured mesh by transferring color information from images acquired from different data sources. More advanced texture computation techniques could be explored for better quality of visualization. For instance, differential rendering provides an optimization framework for estimating the texture map given a set of images registered to the mesh. Moreover, our pipeline could be utilized for transferring other attributes from the image domain to the mesh surface, i.e. mapping and aggregating pixel-wise features (material traits, semantic labels etc.) to produce facets with rich properties. The key problem involves resolving ambiguities that arise from errors in camera parameters, and the inconsistency between high-resolution image data and the coarse mesh representation.

Semantic scene understanding. Our representation can be enriched by introducing semantic labels to the geometric model. A direct improvement to the current pipeline is to design more advanced methods for classifying pixels in floor plans for generating semantic maps. It could benefit the vectorization of floor plans with various styles. Another way could be to integrate an object detection component for separating objects from the building interior, before projecting pixels onto the mesh. This technique allows generating a cleaner texture for the floor model, with the option for adding furniture later.

Panoptic scene modeling. Automatic reconstruction of a whole building including exterior structures, interior layout and even furniture inside is one of the most challenging problems in small-scale urban modeling. This requires the design and integration of specialized pipelines for outdoor and

indoor scenes in a single consistent framework. The main challenge lies in the geometric and photometric differences between the indoor and the outdoor models. One possible direction is to detect structures (e.g. windows) that are present in both models, and estimate correspondences by utilizing these structures [DVM21].

Publications

This thesis is supported by the following publications:

- Muxingzi Li, Florent Lafarge, Renaud Marlet. Approximating shapes in images with low-complexity polygons. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [Oral presentation]
- Muxingzi Li, Florent Lafarge. Planar Shape Based Registration for Multi-modal Geometry. Accepted to *the British Machine Vision Conference (BMVC)*, 2021.

During the course of my PhD (and an internship at Alibaba), I have also published the following papers not within the scope of this thesis:

- Muxingzi Li*, Peihan Tu*, Wolfgang Heidrich. Robust Joint Image Reconstruction from Color and Monochrome Cameras. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2019.
- Xiaopeng Sun*, Muxingzi Li*, Tianyu He, Lubin Fan. Enhance Image as You Like with Unpaired Learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Appendix A

Appendix

This appendix provides pseudo-codes illustrating the kinetic propagation step of the polygon approximation algorithm presented in chapter 3, and the surface-normal clustering step of the registration algorithm presented in chapter 4. Supplementary experiment information for chapter 4 is also included.

A.1 Polygonal image segmentation: a pseudo-code for kinetic propagation inside a nested polygon

Notations. Let:

- P be a nested polygon, with outer boundary defined as a set of edges $\{e_i^{\text{out}}\}$, and inner boundary defined by another set of edges $\{e_i^{\text{in}}\}$.
- $\{\mathbf{s}_k\}_{k=1}^{N}$ be a set of line-segments inside the polygon.
- $\{\mathbf{r}_k(t)\}_{k=1}^{2N}$ be a set of rays created from the above line-segments. For each line-segment, two opposite rays are created, representing propagation from the two ends of the line-segment.

The algorithm partitions a nested polygon by propagating a set of rays. Note that the inner boundary and the outer boundary of the polygon are treated differently. A ray stops immediately when colliding with the outer boundary, whereas it continues to propagate when colliding with the inner boundary. This is to avoid the creation of sub-polygons with duplicated edges, therefore maintains a valid geometric structure of the polygonal partition. Stopping criteria of ray propagation can be defined by a maximal number of collisions. The pseudo-code is given in Algorithm 2.

Algorithm 2 Kinetic propagation of line-segments inside a nested polygon

Input: P, $\{\mathbf{s}_k\}_{k=1}^{N}$

Output: a partition of the input polygon P

```
Initialize a polygonal partition x_P with {e_i^out} and {e_i^in}
Initialize rays {r_i(t)}_{k=1}^{2N} from {s_k}_{k=1}^{N}
Initialize t_0 := −∞, t_1 := T
while rays not all stopped do
    Schedule events within time window [t_0, t_1], and store in a priority
    queue Q
    while Q not empty do
        Pop the top event i of Q
        Update x_P by building edges and vertices
        if collision with the outer boundary then
            Stop the ray
        else if collision between rays then
            Update status of rays according to the stopping criteria
        end if
        Update Q
    end while
    Update t_0 := t_1, and t_1 := t_1 + T
end while
```

A.2 3D registration of multi-modal geometry: a pseudo-code for clustering surface-normals of planar shapes

Notations. Let:

- $\mathbf{n}_i \in \mathbb{R}^3$ be the surface normal of the i-th planar shape.
- $\mathbf{N} = \{\mathbf{n}_1, ..., \mathbf{n}_m\}$ be the set of surface normals.
- $a_i \in \mathbb{R}$ be the surface area of the i-th planar shape.
- $w_i = f(a_i) \in \mathbb{R}$ be a non-negative weight assigned to the i-th planar shape. In this thesis, we simply define the function f as $f : x \in \mathbb{R} \mapsto x \in \mathbb{R}$ such that the weight is proportional to the shape area.
- $\mathbf{W} = \{w_1, ..., w_m\}$ be the set of weights.
- $\boldsymbol{\theta}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ be the parameters defining the k-th Gaussian component, with mean $\boldsymbol{\mu}_k \in \mathbb{R}^3$ and covariance $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3\times3}$.
- $p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{N}_g(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the probability density of a multivariate Gaussian distribution at $\mathbf{x}$, under a distance metric defined by the function $g : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$.
- $\{\pi_1, ..., \pi_K\}$ be mixture coefficients, where $\pi_k \leq 0$ is the coefficient for the k-th Gaussian component, satisfying $\sum_{k=1}^{K} \pi_k = 1$.

The algorithm clusters the surface-normals of a set of planar shapes into K clusters, under the Gaussian mixture model assumption for weighted data. The distance metric is the absolute cosine similarity metric (see Eq. 4.8) instead of the Euclidean distance for measuring the distance between two normal vectors on the spherical surface. The pseudo-code is given in Algorithm 3.

Algorithm 3 Weighted-data clustering with the EM algorithm

Input: $\mathbf{N} = \{\mathbf{n}_1, ..., \mathbf{n}_m\}$, $\mathbf{W} = \{w_1, ..., w_m\}$

Output: $\{\pi_1, ..., \pi_K\}$, $\{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\}$

1: Initialize π_k, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ for all k

2: **while** not converging **do**

3: $\quad p(z_i = k|\mathbf{n}_i; \boldsymbol{\theta}, w_i) := \frac{\pi_k \mathcal{N}_g(\mathbf{n}_i; \boldsymbol{\theta}_k, \frac{1}{w_i}\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}_g(\mathbf{n}_i; \boldsymbol{\theta}_j, \frac{1}{w_i}\boldsymbol{\Sigma}_j)}$

4: $\quad \pi_k := \frac{1}{m}\sum_{i=1}^{m} p(z_i = k|\mathbf{n}_i; \boldsymbol{\theta}, w_i)$

5: $\quad \boldsymbol{\mu}_k := \frac{\sum_{i=1}^{m} w_i p(z_i=k|\mathbf{n}_i;\boldsymbol{\theta},w_i)\mathbf{n}_i}{\sum_{i=1}^{m} w_i p(z_i=k|\mathbf{n}_i;\boldsymbol{\theta},w_i)}$

6: $\quad$ Normalize $\boldsymbol{\mu}_k$

7: $\quad \boldsymbol{\Sigma}_k := \frac{\sum_{i=1}^{m} w_i p(z_i=k|\mathbf{n}_i;\boldsymbol{\theta},w_i)(\mathbf{n}_i-\boldsymbol{\mu}_k)(\mathbf{n}_i-\boldsymbol{\mu}_k)^T}{\sum_{i=1}^{m} p(z_i=k|\mathbf{n}_i;\boldsymbol{\theta},w_i)}$

8: **end while**

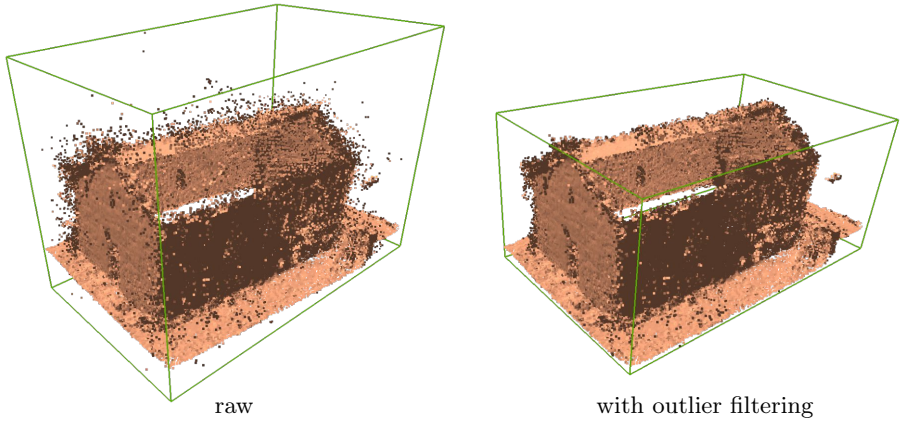


Figure A.1: Illustration of estimated bounding boxes without and with outlier filtering. Left: bounding box estimated on the raw point cloud. Right: bounding box estimated on the filtered point cloud.

A.3 3D registration of multi-modal geometry: scale estimation for two-step baselines

Existing two-step pipelines for model-to-scene alignment has shown that bounding boxes provide a good estimate for the relative scale between objects with sufficient overlap [CDG+13, MDS16], which can be used for a later 6-DoF alignment step [GAGM15, IS20]. We adopt the bounding box based method for estimating a relative scale, which consists of an outlier filtering step and a bounding box estimation step. Outlier filtering is done with the KNN algorithm as the following: First compute for each point the mean distance to its K nearest neighbors. Average KNN distances for all points give a distribution with a sample mean μ and sample standard deviation σ. All points with a mean KNN distance greater than $\mu + 2\sigma$ are classified as outliers, which approximately corresponds to an outlier ratio of 2.3% under the normal distribution assumption. The effectiveness of outlier filtering is illustrated in Fig. A.1. Optimal bounding boxes are estimated using the algorithm of Chang et al.[CGM11]. Let B_d and B_m denote the bounding boxes of the source and the target, respectively. The scale is computed as the ratio between the lengths of the bounding box diagonals,

$$s = \frac{\text{Diagonal}(B_m)}{\text{Diagonal}(B_d)}. \tag{A.1}$$

Bibliography

[AA96] Oswin Aichholzer and Franz Aurenhammer. Straight skeletons for general polygonal figures in the plane. In Jin-Yi Cai and Chak Kuen Wong, editors, *Computing and Combinatorics*, pages 117–126, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. (Cited on page 21.)

[AA19] Hamed Amini Amirkolaee and Hossein Arefi. Height estimation from single aerial images using a deep convolutional encoder-decoder network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 149:50–66, 2019. (Cited on page 21.)

[ADD+19] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Niessner. Scan2cad: Learning cad model alignment in rgb-d scans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. (Cited on pages 28 and 29.)

[AGASL19] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & efficient point cloud registration using PointNet. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. (Cited on pages 27 and 51.)

[AGG+10] P. Agarwal, J. Gao, L. Guibas, H. Kaplan, V. Koltun, N. Rubin, and M. Sharir. Kinetic stable delaunay graphs. In *Proc. of Symposium on Computational Geometry (SoCG)*, 2010. (Cited on page 33.)

[ALKF18] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018. (Cited on page 24.)

[AMCO08] Dror Aiger, Niloy J. Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust surface registration. *ACM Transactions on Graphics*, 27(3), 2008. (Cited on page 26.)

[AMO] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org. (Cited on page 59.)

[AS17] R. Achanta and S. Susstrunk. Superpixels and polygons using simple non-iterative clustering. In *CVPR*, 2017. (Cited on pages 25 and 31.)

[ASF+13] M. Arikan, M. Schwarzler, S. Flory, M. Wimmer, and S. Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *Trans. on Graphics*, 32(1), 2013. (Cited on page 19.)

[BB13] Heiko Bülow and Andreas Birk. Spectral 6DOF registration of noisy 3d range data with partial overlap. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):954–969, 2013. (Cited on page 27.)

[BB18] Heiko Bülow and Andreas Birk. Scale-free registrations in 3d: 7 degrees of freedom with fourier mellin soft transforms. *International Journal of Computer Vision*, 2018. (Cited on page 28.)

[BBPDM08] M. Brédif, D. Boldo, M. Pierrot-Deseilligny, and H. Maître. 3d building model fitting using a new kinetic framework. *arXiv:0805.0648*, 2008. (Cited on page 33.)

[BDLGM14] A. Boulch, M. De La Gorce, and R. Marlet. Piecewise-planar 3d reconstruction with edge and corner regularization. *Computer Graphics Forum*, 33(5), 2014. (Cited on page 19.)

[BGH99] J. Basch, L. Guibas, and J. Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31(1), 1999. (Cited on pages 32 and 38.)

[BHH+15] Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader. Weighted straight skeletons in the plane. *Computational Geometry*, 48(2):120–133, 2015. (Cited on pages 21 and 76.)

[BKP+10] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. *Polygon Mesh Processing*. AK Peters / CRC Press, 2010. (Cited on pages 36 and 46.)

[BL18] J.-P. Bauchet and F. Lafarge. Kippi: Kinetic polygonal partitioning of images. In *CVPR*, 2018. (Cited on pages 25, 31, 33, 36, 38, 44 and 45.)

[BL20] Jean-Philippe Bauchet and Florent Lafarge. Kinetic shape reconstruction. *ACM Trans. Graph.*, 39(5), June 2020. (Cited on pages 19, 20, 33 and 59.)

[BLS16] Filip Biljecki, Hugo Ledoux, and Jantien Stoter. An improved lod specification for 3d building models. *Computers, Environment and Urban Systems*, 59:25–37, 2016. (Cited on page 12.)

[BM92] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. (Cited on pages 26 and 51.)

[BSKK13] Erik Bylow, Jurgen Sturm, Christian Kerl, and Fredrik Kahl. Real-time camera tracking and 3d reconstruction using signed distance functions. In *RSS*, 2013. (Cited on page 26.)

[BTP13] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. In *Symposium on Geometry Processing (SGP)*, page 113–123, 2013. (Cited on page 26.)

[CC08] J. Chen and B. Chen. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision (IJCV)*, 78(2-3), 2008. (Cited on page 19.)

[CDF+17] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. (Cited on pages 4 and 74.)

[CDG+13] Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, Riccardo Gherardi, Andrea Fusiello, and Roberto Scopigno. Fully automatic registration of image sets on approximate

geometry. *International Journal of Computer Vision*, 102(1-3):91–111, March 2013. (Cited on pages 28, 51, 56, 63 and 89.)

[CDK20] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on pages 27, 51, 63, 66 and 67.)

[CF14] Ricardo Cabral and Yasutaka Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, page 628–635, 2014. (Cited on pages 22 and 69.)

[CGM11] Chia-Tche Chang, Bastien Gorissen, and Samuel Melchior. Fast oriented bounding box optimization on the rotation group so(3,r). *ACM Trans. Graph.*, 30(5), October 2011. (Cited on page 89.)

[CKPT18] J. Czakon, K. Kaczmarek, Andrzej P., and P. Tarasiewicz. Best practices for elegant experimentation in data science projects. In *EuroPython*, 2018. (Cited on page 46.)

[CKUF17] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017. (Cited on page 24.)

[CMH+15] M. Cheng, N. Mitra, X. Huang, P. Torr, and S.-M. Hu. Global contrast based salient region detection. *PAMI*, 37(3), 2015. (Cited on page 23.)

[CPK+19] Dylan Campbell, Lars Petersson, Laurent Kneip, Hongdong Li, and Stephen Gould. The alignment of the spheres: Globally-optimal spherical mixture alignment for camera pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11796–11806, 2019. (Cited on pages 27 and 51.)

[Cra07] A. Cracknell. *Introduction to remote sensing*. CRC press, 2007. (Cited on page 3.)

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914, August 2004. (Cited on page 56.)

[CSaLM13] Desai Chen, Pitchaya Sitthi-amorn, Justin T. Lan, and Wojciech Matusik. Computing and fabricating multiplanar models. *Computer Graphics Forum*, 32(2pt3):305–315, 2013. (Cited on page 18.)

[CSL13] Daniel R. Canelhas, Todor Stoyanov, and Achim J. Lilienthal. Sdf tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In *IROS*, 2013. (Cited on page 26.)

[CTZ20] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bspnet: Generating compact meshes via binary space partitioning. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (Cited on page 20.)

[CWP13] Sung-In Choi, Udaya Wijenayake, and Soon-Yong Park. Head pose tracking using gpu based real-time 3d registration. In *IEEE International Workshop on Robot and Human Interactive Communication*, 08 2013. (Cited on page 26.)

[CY01] James Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2001. (Cited on page 18.)

[CZP+18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. (Cited on page 45.)

[DDS09] Christopher Dyken, Morten Dæhlen, and Thomas Sevaldrud. Simultaneous curve simplification. *Journal of Geographical Systems*, 11(3), 2009. (Cited on pages 23, 31, 43 and 45.)

[DEGN98] Tamal K. Dey, Herbert Edelsbrunner, Sumanta Guha, and Dmitry V. Nekhayev. Topology preserving edge contraction.

Publ. Inst. Math. (Beograd) (N.S, 66:23–45, 1998. (Cited on page 18.)

[DGCSAD11] F. De Goes, D. Cohen-Steiner, P. Alliez, and M. Desbrun. An optimal transport approach to robust reconstruction and simplification of 2d shapes. *Computer Graphics Forum*, 30(5), 2011. (Cited on pages 23 and 31.)

[DL15] L. Duan and F. Lafarge. Image partitioning into convex polygons. In *CVPR*, 2015. (Cited on pages 25, 31, 38, 44 and 45.)

[DS20] Jérémie Deray and Joan Solà. Manif: A micro lie theory library for state estimation in robotics applications. *J. Open Source Softw.*, 5(46):1371, 2020. (Cited on page 52.)

[DST+21] Angela Dai, Yawar Siddiqui, Justus Thies, Julien Valentin, and Matthias Nießner. Spsg: Self-supervised photometric scene generation from rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021. (Cited on page 69.)

[DVM21] R. Djahel, B. Vallet, and P. Monasse. Towards efficient indoor/outdoor registration using planar polygons. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2021:51–58, 2021. (Cited on page 81.)

[DZ13] P. Dollar and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. (Cited on page 35.)

[DZY+07] Shaoyi Du, Nanning Zheng, Shihui Ying, Qubo You, and Yang Wu. An extension of the ICP algorithm considering scale factor. In *2007 IEEE International Conference on Image Processing (ICIP)*, volume 5, 2007. (Cited on page 28.)

[EH18] Georgios D. Evangelidis and Radu Horaud. Joint alignment of multiple point sets with batch and incremental expectation-maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. (Cited on page 26.)

[EKK18] Ben Eckart, Kihwan Kim, and Jan Kautz. Fast and accurate point cloud registration using trees of gaussian mixtures.

In *European Conference on Computer Vision (ECCV)*, 2018. (Cited on pages 27 and 51.)

[EKT+15] Ben Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. MLMD: Maximum likelihood mixture decoupling for fast and accurate point cloud registration. In *2015 International Conference on 3D Vision*, pages 241–249, 2015. (Cited on page 26.)

[EVGW+] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. (Cited on page 45.)

[FA20] Qiaojun Feng and Nikolay Atanasov. Fully convolutional geometric features for category-level object alignment. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8492–8498, 2020. (Cited on page 28.)

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. (Cited on page 18.)

[FCSS09a] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Manhattan-world stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1422–1429, 2009. (Cited on page 18.)

[FCSS09b] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *2009 IEEE 12th International Conference on Computer Vision*, pages 80–87, 2009. (Cited on page 22.)

[Fit01] Andrew Fitzgibbon. Robust registration of 2d and 3d point sets. In *BMVC*, volume 21, 2001. (Cited on pages 26 and 55.)

[FKF16] J. Forsythe, V. Kurlin, and A. Fitzgibbon. Resolution independent superpixels based on convex constrained meshes

without small angles. In *International Symposium on Visual Computing*, 2016. (Cited on pages 25 and 31.)

[FL20] Hao Fang and Florent Lafarge. Connect-and-slice: An hybrid approach for reconstructing 3d objects. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13487–13495, 2020. (Cited on page 19.)

[FLB16] J.-D. Favreau, F. Lafarge, and A. Bousseau. Fidelity vs. Simplicity: a Global Approach to Line Drawing Vectorization. *ACM Trans. on Graphics*, 35(4), 2016. (Cited on page 42.)

[FLBA20] J.D. Favreau, F. Lafarge, A. Bousseau, and A. Auvolat. Extracting geometric structures in images with delaunay point processes. *PAMI*, 42(4), 2020. (Cited on pages 24 and 70.)

[FLD18] Hao Fang, Florent Lafarge, and Mathieu Desbrun. Planar Shape Detection at Structural Scales. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, United States, 2018. (Cited on page 19.)

[FLPH21] Hao Fang, Florent Lafarge, Cihui Pan, and Hui Huang. Floorplan generation from 3d point clouds: a space partitioning approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 175:44–55, 2021. (Cited on page 20.)

[FPH21] Hao Fang, Cihui Pan, and Hui Huang. Structure-aware indoor scene reconstruction via two levels of abstraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 178:155–170, 2021. (Cited on page 18.)

[GAGM15] Saurabh Gupta, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4731–4740, 2015. (Cited on pages 28, 51 and 89.)

[GAPFH16] Israel Dejene Gebru, Xavier Alameda-Pineda, Florence Forbes, and Radu Horaud. Em algorithms for weighted-data clustering with application to audio-visual scene anal-

ysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(12):2402–2415, 2016. (Cited on page 56.)

[GH97a] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH*, 1997. (Cited on pages 36 and 46.)

[GH97b] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 209–216, USA, 1997. (Cited on page 18.)

[GKOM18] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNet: Learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85, 2018. (Cited on page 19.)

[GLCV19] S. Guinard, L. Landrieu, L. Caraffa, and B. Vallet. Piecewise-planar approximation of large 3d data as graph-structured optimization. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:365–372, 2019. (Cited on page 19.)

[GP02] Sébastien Granger and Xavier Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. In *European Conference on Computer Vision (ECCV)*, pages 418–432, 2002. (Cited on page 26.)

[GP12] G. Groger and L. Plumer. Citygml – interoperable semantic 3d city models. *Journal of Photogrammetry and Remote Sensing*, 71, 2012. (Cited on page 1.)

[GS97] T. Gevers and A. W. M. Smeulders. Combining region splitting and edge detection through guided delaunay image subdivision. In *CVPR*, 1997. (Cited on page 25.)

[GT18] N. Girard and Y. Tarabalka. End-to-End Learning of Polygons for Remote Sensing Image Classification. In *IGARSS*, 2018. (Cited on page 24.)

[GT19] Wei Gao and Russ Tedrake. Filterreg: Robust and efficient probabilistic point-set registration using gaussian filter and twist parameterization. In *Conference On Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 26, 63, 66 and 67.)

[Gui04] L. Guibas. Kinetic data structures. In *Handbook of Data Structures and Applications*, 2004. (Cited on page 32.)

[HDG21] Sofiane Horache, Jean-Emmanuel Deschaud, and François Goulette. 3d point cloud registration with multi-scale architecture and self-supervised fine-tuning. *CoRR*, abs/2103.14533, 2021. (Cited on page 28.)

[HEH07] Derek Hoiem, Alexei Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75:151–172, 07 2007. (Cited on page 22.)

[HGDG17] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. (Cited on page 46.)

[HHF09] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1849–1856, 2009. (Cited on page 22.)

[Hir20a] Osamu Hirose. Acceleration of non-rigid point set registration with downsampling and gaussian process regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. (Cited on pages 28, 62, 66 and 67.)

[Hir20b] Osamu Hirose. A bayesian formulation of coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. (Cited on page 28.)

[HIT+15] Dirk Holz, Alexandru E. Ichim, Federico Tombari, Radu B. Rusu, and Sven Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics Automation Magazine*, 22(4):110–124, 2015. (Cited on pages 26 and 51.)

[HJS+14] Jin Huang, Tengfei Jiang, Zeyun Shi, Yiying Tong, Hujun Bao, and Mathieu Desbrun. l1-based construction of polycube maps from complex shapes. *ACM Trans. Graph.*, 33(3), June 2014. (Cited on page 18.)

[HZF+17] Xiaoshui Huang, Jian Zhang, Lixin Fan, Qiang Wu, and Chun Yuan. A systematic approach for cross-source point cloud registration by preserving macro and micro structures. *IEEE Transactions on Image Processing*, 26(7), 2017. (Cited on page 28.)

[IBA+20] Vladislav Ishimtsev, Alexey Bokhovkin, Alexey Artemov, Savva Ignatyev, Matthias Niessner, Denis Zorin, and Evgeny Burnaev. Cad-deform: Deformable fitting of cad models to 3d scans. In *ECCV*, pages 599–628, Cham, 2020. Springer International Publishing. (Cited on pages 28 and 51.)

[IKH+11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 559–568. Association for Computing Machinery, 2011. (Cited on page 6.)

[IS20] Hamid Izadinia and Steven M. Seitz. Scene recomposition by learning-based icp. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 927–936, 2020. (Cited on pages 28 and 89.)

[IYF15] Satoshi Ikehata, Hang Yang, and Yasutaka Furukawa. Structured indoor modeling. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1323–1331, 2015. (Cited on page 18.)

[IZKA14] P. Isola, D. Zoran, D. Krishnan, and E.H. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, 2014. (Cited on page 35.)

[JV11] Bing Jian and Baba C. Vemuri. Robust point set registration using gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2011. (Cited on page 26.)

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, 2006. (Cited on page 17.)

[KE12] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012. (Cited on page 6.)

[Kir08] A. Kirillov. *An Introduction to Lie Groups and Lie Algebras*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2008. (Cited on page 52.)

[KPZK17] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *Trans. on Graphics*, 36(4), 2017. (Cited on pages 4 and 74.)

[KSA06] Yosi Keller, Yoel Shkolnisky, and Amir Averbuch. Volume registration using the 3-d pseudopolar fourier transform. *IEEE Transactions on Signal Processing*, 54(11):4323–4331, 2006. (Cited on page 27.)

[KvLS07] R. Kluszczyński, M. N. M. van Lieshout, and T. Schreiber. Image segmentation by polygonal markov fields. *Annals of the Institute of Statistical Mathematics*, 59(3), 2007. (Cited on page 24.)

[KW11] Tom Kelly and Peter Wonka. Interactive architectural modeling with procedural extrusions. *ACM Trans. Graph.*, 30(2), April 2011. (Cited on page 21.)

[LA13] F. Lafarge and P. Alliez. Surface reconstruction through point set structuring. In *Computer Graphics Forum*, volume 32, 2013. (Cited on page 19.)

[LBMR17] Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz, and Andrew Rabinovich. Roomnet: End-to-end room layout estimation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4875–4884, 2017. (Cited on page 22.)

[LD03] R. G. Laycock and A. M. Day. Automatically generating large urban environments based on the footprint data of buildings. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, page 346–351, 2003. (Cited on page 21.)

[LDWL19] Z. Li, J. Dirk Wegner, and A. Lucchi. Topological map extraction from overhead images. In *ICCV*, 2019. (Cited on pages 24 and 46.)

[LGK+19] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019. (Cited on page 24.)

[LHK09] David C. Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2136–2143, 2009. (Cited on page 22.)

[LKK+20] Chao-Jung Liu, Vladimir A. Krylov, Paul Kane, Geraldine Kavanagh, and Rozenn Dahyot. Im2elevation: Building height estimation from single-view aerial imagery. *Remote Sensing*, 12(17), 2020. (Cited on page 21.)

[LM12] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: A robust approach with hybrid representation. *Int. J. Comput. Vision*, 99(1):69–85, 2012. (Cited on page 56.)

[LQQ+18] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. (Cited on pages 24 and 46.)

[LSD10] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Optimal contour closure by superpixel grouping. In *ECCV*, 2010. (Cited on page 23.)

[LSD+19] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. Guibas. Supervised Fitting of Geometric Primitives to 3D Point Clouds . In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on page 19.)

[LT98] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 279–286, 1998. (Cited on page 18.)

[LTR+13] Baowei Lin, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda, and Koji Ichii. Scale ratio ICP for 3d point clouds with different scales. In *2013 IEEE International Conference on Image Processing (ICIP)*, pages 2217–2221, 2013. (Cited on page 28.)

[LWKF17] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-vector: Revisiting floorplan transformation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2214–2222, 2017. (Cited on pages 22, 69 and 70.)

[LY16] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016. (Cited on pages 23, 42 and 43.)

[LZMC12] Z. Li, Wu Z.-M., and S.-F. Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *CVPR*, 2012. (Cited on page 25.)

[LZX+20] Jiahao Li, Changhao Zhang, Ziyao Xu, Hangning Zhou, and Chi Zhang. Iterative distance-aware similarity matrix convolution with mutual-supervised point elimination for efficient point cloud registration. In *European Conference on Computer Vision (ECCV)*, 2020. (Cited on pages 27 and 51.)

[MAM14] Nicolas Mellado, Dror Aiger, and Niloy J. Mitra. Super 4PCS fast global pointcloud registration via smart indexing.

Computer Graphics Forum, 33(5):205–215, 2014. (Cited on pages 26, 51 and 63.)

[MDS16] Nicolas Mellado, Matteo Dellepiane, and Roberto Scopigno. Relative scale estimation and 3d registration of multi-modal geometry using growing least squares. *IEEE Transactions on Visualization and Computer Graphics*, 22(9):2160–2173, 2016. (Cited on pages 28, 51 and 89.)

[ML15] Arun Mallya and Svetlana Lazebnik. Learning informative edge maps for indoor scene layout prediction. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 936–944, USA, 2015. IEEE Computer Society. (Cited on page 22.)

[MLM01] D. Marshall, G. Lukacs, and R. Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 23(3), 2001. (Cited on page 19.)

[MMP16] C. Mura, O. Mattausch, and R. Pajarola. Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. *Computer Graphics Forum*, 35(7), 2016. (Cited on page 19.)

[Moha] S. Mohanty. Crowdai dataset: the mapping challenge. https://www.aicrowd.com/challenges/. (Cited on page 46.)

[Mohb] S.P. Mohanty. Crowdai mapping challenge : Baseline with mask r-cnn. https://github.com/crowdAI/crowdai-mapping-challenge-mask-rcnn/. (Cited on page 46.)

[MPBF20] Jisan Mahmud, True Price, Akash Bapat, and Jan-Michael Frahm. Boundary-aware 3d building reconstruction from a single overhead image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. (Cited on page 21.)

[MS10] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis*

and Machine Intelligence, 32(12):2262–2275, 2010. (Cited on pages 28, 62, 66 and 67.)

[MTCA17] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark. In *IGARSS*, 2017. (Cited on page 46.)

[Nis03] Nister. Preemptive ransac for live structure and motion estimation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 199–206 vol.1, 2003. (Cited on page 19.)

[NW17] L. Nan and P. Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *Proc. of International Conference on Computer Vision (ICCV)*, 2017. (Cited on pages 4 and 19.)

[NXS12] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2012)*, 31, 11 2012. (Cited on page 28.)

[OLA14] S. Oesau, F. Lafarge, and P. Alliez. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90:68–82, 2014. (Cited on page 19.)

[OLA16] S. Oesau, F. Lafarge, and P. Alliez. Planar shape detection and regularization in tandem. In *Computer Graphics Forum*, volume 35, 2016. (Cited on page 19.)

[OVJ+18] S. Oesau, Y. Verdie, C. Jamin, P. Alliez, F. Lafarge, and S. Giraudot. Point set shape detection. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition, 2018. (Cited on page 38.)

[PBCE+16] Álvaro Parra Bustos, Tat-Jun Chin, Anders Eriksson, Hongdong Li, and David Suter. Fast rotation search with stereographic projections for 3d registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2227–2240, 2016. (Cited on pages 27 and 51.)

[PHDV15] Danda Pani Paudel, Adlane Habed, Cédric Demonceaux, and Pascal Vasseur. Robust and optimal sum-of-squares-based point-to-plane registration of image sets and structured scenes. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2048–2056, 2015. (Cited on page 28.)

[PRR02] Nikos Paragios, Mikaël Rousson, and Visvanathan Ramesh. Matching distance functions: A shape-to-area variational approach for global-to-local registration. In *ECCV*, 2002. (Cited on page 26.)

[QZF21] Yiming Qian, Hao Zhang, and Yasutaka Furukawa. RoofGAN: Learning to generate roof geometry and relations for residential houses. In *CVPR*, 2021. (Cited on pages 3 and 21.)

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009. (Cited on pages 26 and 51.)

[RKB04] C. Rother, V. Kolmogorov, and A. Blake. Grabcut -interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graphics*, 23(3), 2004. (Cited on page 23.)

[RL93] G. Roth and M.D. Levine. Extracting geometric primitives. *CVGIP: Image Understanding*, 58(1):1–22, 1993. (Cited on page 18.)

[RL01] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 145–152, 2001. (Cited on page 26.)

[RPJT13] Srikumar Ramalingam, Jaishanker K. Pillai, Arpit Jain, and Yuichi Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3065–3072, 2013. (Cited on pages 22 and 69.)

[RS13] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *CVPR*, 2013. (Cited on page 25.)

[Rus19] Szymon Rusinkiewicz. A symmetric objective function for ICP. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), July 2019. (Cited on pages 26, 28, 62, 66 and 67.)

[RVDHV06] T. Rabbani, F. Van Den Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraint. *ISPRS*, 36(5), 2006. (Cited on page 19.)

[SCF14] X. Sun, M. Christoudias, and P. Fua. Free-shape polygonal object localization. In *ECCV*, 2014. (Cited on pages 23 and 24.)

[Sel04] J. M. Selig. Lie groups and lie algebras in robotics. In Jim Byrnes, editor, *Computational Noncommutative Algebra and Applications*, pages 101–125, Dordrecht, 2004. Springer Netherlands. (Cited on page 52.)

[SF16] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on page 69.)

[SHSC19] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1047–1056, 2019. (Cited on page 22.)

[SHT09] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. Generalized-icp. In *Proc. of Robotics: Science and Systems*, 2009. (Cited on page 26.)

[SK13] Kenichi Sugihara and Junne Kikata. Automatic generation of 3d building models from complicated building polygons. *Journal of Computing in Civil Engineering*, 27(5):476–488, 2013. (Cited on page 21.)

[SK18] Kenichi Sugihara and Youry Khmelevsky. Roof report from automatically generated 3d building models by straight skeleton computation. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–8, 2018. (Cited on page 20.)

[SKNI16] Miroslava Slavcheva, Wadim Kehl, Nassir Navab, and Slobodan Ilic. Sdf-2-sdf: Highly accurate 3d object reconstruction. In *ECCV*, 2016. (Cited on page 26.)

[SLA15] D. Salinas, F. Lafarge, and P. Alliez. Structure-aware mesh decimation. *Comput. Graph. Forum*, 34(6):211–227, September 2015. (Cited on page 18.)

[SRF+14] J. Straub, G. Rosman, O. Freifeld, J. J. Leonard, and J. W. Fisher. A mixture of manhattan frames: Beyond the manhattan world. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3770–3777, 2014. (Cited on page 56.)

[ST20] E. Saiti and T. Theoharis. An application independent review of multimodal 3d registration methods. *Computers & Graphics*, 91:153 – 178, 2020. (Cited on page 28.)

[STM+21] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. RetrievalFuse: Neural 3D Scene Reconstruction with a Database. *arXiv e-prints*, page arXiv:2104.00024, March 2021. (Cited on page 69.)

[SWF11] F. Schindler, W. Worstner, and J.-. Frahm. Classification and reconstruction of surfaces from point clouds of man-made objects. In *Proc. of International Conference on Computer Vision (ICCV) Workshops*, 2011. (Cited on page 19.)

[SWK07a] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007. (Cited on page 19.)

[SWK07b] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007. (Cited on pages 26 and 51.)

[SWWK08] R. Schnabel, R. Wessel, R. Wahl, and R. Klein. Shape recognition in 3d point-clouds. 2008. (Cited on page 19.)

[SX14] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, pages 634–651, 2014. (Cited on page 28.)

[SZPF16] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016. (Cited on page 69.)

[TBD05] R. Triebel, W. Burgard, and F. Dellaert. Using hierarchical em to extract planes from 3d range scans. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4437–4442, 2005. (Cited on page 56.)

[The] The CGAL Project. CGAL, computational geometry algorithms library. (Cited on page 41.)

[The21] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.2.1 edition, 2021. (Cited on page 59.)

[TZ00] P.H.S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. (Cited on page 19.)

[VGJMR10] R. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *PAMI*, 32(4), 2010. (Cited on page 25.)

[VKVLV11] M. Van Kreveld, T. Van Lankveld, and R. Veltkamp. On the shape of a set of points and lines in the plane. *Computer Graphics Forum*, 30, 2011. (Cited on page 19.)

[VKVLV13] M. Van Kreveld, T. Van Lankveld, and R. Veltkamp. Watertight scenes from urban lidar and planar surfaces. In *Proc. of tSymposium on Geometry Processing (SGP)*. Eurographics Association, 2013. (Cited on page 19.)

[VLA15] Yannick Verdie, Florent Lafarge, and Pierre Alliez. Lod generation for urban scenes. *ACM Trans. Graph.*, 34(3), May 2015. (Cited on pages 2, 12, 18 and 19.)

[VWF+21] Madhawa Vidanapathirana, Qirui Wu, Yasutaka Furukawa, Angel X. Chang, and Manolis Savva. Plan2scene: Converting floorplans to 3d scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10733–10742, June 2021. (Cited on page 23.)

[WCD+19] Zongze Wu, Hongchen Chen, Shaoyi Du, Minyue Fu, Nan Zhou, and Nanning Zheng. Correntropy based scale icp algorithm for robust point set registration. *Pattern Recognition*, 93:14 – 24, 2019. (Cited on page 28.)

[WM03] S. T. Wu and M. R. G. Marquez. A non-self-intersection Douglas-Peucker algorithm. In *IEEE Symposium on Computer Graphics and Image Processing*, 2003. (Cited on pages 23, 31, 43 and 45.)

[WMG14] M. Waechter, Nils Moehrle, and M. Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *ECCV*, 2014. (Cited on page 72.)

[WS19a] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. (Cited on pages 27, 51, 63, 66 and 67.)

[WS19b] Yue Wang and Justin M. Solomon. PRNet: Self-supervised learning for partial-to-partial registration. In *33rd Conference on Neural Information Processing Systems (NIPS)*, 2019. (Cited on pages 27 and 51.)

[WWL+16] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan. Saliency detection with recurrent fully convolutional networks. In *ECCV*, 2016. (Cited on page 23.)

[YEK+20] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. DeepGMR: Learning latent

gaussian mixture models for registration. In *European Conference on Computer Vision (ECCV)*, pages 733–750. Springer, 2020. (Cited on page 27.)

[YLJ13] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-ICP: Solving 3d registration efficiently and globally optimally. In *2013 IEEE International Conference on Computer Vision (CVPR)*, pages 1457–1464, 2013. (Cited on pages 27 and 51.)

[YWP+19] Shang-Ta Yang, Fu-En Wang, Chi-Han Peng, Peter Wonka, Min Sun, and Hung-Kuo Chu. Dula-net: A dual-projection network for estimating room layouts from a single RGB panorama. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3363–3372, 2019. (Cited on pages 22 and 69.)

[YZT18] Toshihiko Yamasaki, Jin Zhang, and Yuki Takada. Apartment structure estimation using fully convolutional networks and graph model. In *Proceedings of the 2018 ACM Workshop on Multimedia for Real Estate Tech*, page 1–6, 2018. (Cited on page 23.)

[ZCSH18] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2051–2059, 2018. (Cited on page 22.)

[ZFW+12] Z. Zhang, S. Fidler, J. Waggoner, Y. Cao, S. Dickinson, J. Siskind, and S. Wang. Superedge grouping for object localization by combining appearance and shape informations. In *CVPR*, 2012. (Cited on page 24.)

[ZLYF19] Zhiliang Zeng, Xianzhi Li, Ying-Kin Yu, and Chi-Wing Fu. Deep floor plan recognition using a multi-task network with room-boundary-guided attention. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. (Cited on page 23.)

[ZPK16] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV*, pages 766–782, Cham, 2016. Springer International Publishing. (Cited on pages 26, 51, 62, 63, 66 and 67.)

[ZSN05] Timo Zinßer, Jochen Schmidt, and Heinrich Niemann. Point set registration with integrated scale estimation. In *International Conference on Pattern Recognition and Image Processing (PRIP)*, 2005. (Cited on pages 28, 62, 66 and 67.)

[ZSTX14] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 668–686, Cham, 2014. Springer International Publishing. (Cited on page 22.)

[ZXB+19] Yong Zhao, Shibiao Xu, Shuhui Bu, Hongkai Jiang, and Pengcheng Han. Gslam: A general slam framework and benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2019. (Cited on page 52.)

[ZYD21] Juyong Zhang, Yuxin Yao, and Bailin Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. (Cited on pages 26 and 51.)

[ZYY+17] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. (Cited on page 19.)