DEEP LEARNING FOR BUILDING AND VALIDATING GEOMETRIC AND SEMANTIC MAPS

A Dissertation
Presented to
The Academic Faculty

By

John W. Lambert

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing
College of Computing

Georgia Institute of Technology

May 2022

DEEP LEARNING FOR BUILDING AND VALIDATING GEOMETRIC AND SEMANTIC MAPS

Thesis committee:

Dr. James Hays, Co-Advisor
Georgia Institute of Technology & Argo AI

Dr. Frank Dellaert, Co-Advisor
Georgia Institute of Technology & Google Research

Dr. Zsolt Kira
Georgia Institute of Technology

Dr. Cedric Pradalier
Georgia Institute of Technology

Dr. Simon Lucey
University of Adelaide & Australian Institute for Machine Learning

Date approved: March 21, 2022

For my wife Hayoung

ACKNOWLEDGMENTS

This thesis would not have been possible without my PhD co-advisors, James Hays and Frank Dellaert. They have both invested a great deal of time in mentoring and teaching me, and my time spent as a teaching assistant for their courses has added layers of depth to my technical understanding.

I'm grateful to James, for his ability to look far ahead into the future when considering meaningful research directions. He's taught me to be very careful about choosing research topics, which often entail years of effort. I've learned a great deal from him about the value of open collaboration and team-building. Our lab meetings and get-togethers have been something I've always enjoyed, and his lab is a uniquely friendly and enjoyable place. Without his funding and support, this work simply would not have come about.

I'm grateful for Frank's return to Georgia Tech, and for his willingness to say difficult truths. He has always emphasized to me that without crystal-clear communication, technical depth and progress are lost in translation. This advice continues to be invaluable for me. His willingness to embrace and lean in to the bleeding edge of computer vision is also inspiring. Frank's focus on research that is both practical and theoretical has resounded with me from the first time I encountered his work, before coming to Georgia Tech. As a programmer, I continue to learn from Frank; his emphasis on test-driven design has proven invaluable for my research over the past few years.

Thank you also to my labmates in James' group – Samarth Brahmbhatt, Amit Raj, Cusuh Ham, Sean Foley, Patsorn Sangkloy, and Ben Wilson. They have been invaluable as friends for support throughout the PhD. Thanks to my other collaborators in Frank's lab: Akshay Krishnan, Ayush Baid, Fan Jiang, Jing Wu, Adi Singh, Sushmita Warrier, Xiaolong Wu, Travis Driver, and Ren Liu, who have taught me a great deal from their technical depth. Thank you to the other students I've been able to mentor at Georgia Tech – Alex Butenko, Vivek Vanga, Hemanth Chittanuru, Chen Liu, Andrey Pak, Jeevanjot Singh, Shubhangi

Upasani, and Anshul Ahluwalia – as we've explored challenging new problems.

Any success I've enjoyed has been a product of amazing collaborations and mentorship. Thank you to Ozan Sener, for getting me started on my research journey and teaching me the basics of successful research. I'm grateful to my collaborators from my time at Argo AI – Allie Chang, Jagjeet Singh, Tanmay Agarwal, Will Qi, Andrew Hartnett, Jhony Pontes, Peter Carr, Deva Ramanan, Simon Lucey, Rob Keelan, Matt Gilson, Xiaoyan Hu, Kunal Desai, Ben Xinjilefu, Guillaume Binet, Ersin Yumer, Arjuna Ariyaratne, and Josh Manela. I owe thanks to my collaborators from my time at Intel Labs – Zhuang Liu, Ozan Sener, and Vladlen Koltun. I enjoyed the friendships I built at Zillow with Sing Bing Kang, Ivo Boyadzhiev, Yuguang Li, Lambert Wixson, Manju Narayana, Will Hutchcroft, Ethan Wan, and Naji Khosravan, and am grateful for the deep domain knowledge they shared with me in order to help mentor my project.

I would like to thank the members of my thesis committee for their help in preparation of this work – James, Frank, Simon, Cedric, and Zsolt. I am honored to have them serve on my committee.

Finally, I acknowledge the love and support of my wife Hayoung. She has been instrumental in my PhD journey and my best friend.

The author gratefully acknowledges the support for this work offered by Georgia Tech, Argo AI, Intel Labs, and Zillow Group. Any views and conclusions contained herein are those of the author, and do not necessarily represent the official positions, express or implied, of the funders.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Mapping the world is an essential tool for making spatial artificial intelligence a reality in our near future. Spatial AI, or embodied intelligence for 3D perception, enables awareness and understanding of our surroundings. Maps serve as a core workhorse of motion prediction and motion planning for modern autonomous vehicles. Maps also enable human users to interact with novel 3D spaces remotely via virtual reality (VR) or convey useful information about an environment through augmented reality (AR).

Current methods for building and validating geometric and semantic maps are limited in several ways. For example, floorplan maps constructed from sparse camera views within indoor environments generally suffer from low completeness. In other domains, such as city streets, the world is ever-changing, making online validation of high-definition (HD) maps a requirement for today's self-driving vehicles; however, many current map change detection methods suffer from high-storage costs or limited accuracy.

This dissertation research introduces new algorithms for building and validating geometric and semantic maps using deep learning, with three original contributions. I first develop a new learning-based algorithm, SALVe, for creating complete and accurate 2d geometric maps (floorplans) under very wide baselines and occlusion. Second, I explore the role of the deep "front end" in Structure-from-Motion (SfM), and analyze its use in GTSFM, a new system for global SfM. Finally, I introduce learning-based formulations for solving the HD map change detection task in a bird's eye view and ego-view. Because real map changes are infrequent and vector maps are easy to synthetically manipulate, we lean on simulated data to train such models. Perhaps surprisingly, we show that such models can generalize to real world distributions. Along the way, in order to satisfy the demands of these data-driven, deep learning approaches, I contribute several large-scale datasets towards solving these problems – the Argoverse 1.0 Datasets, the MSeg Dataset, the Trust but Verify (TbV) Dataset, and the Argoverse 2.0 Datasets.

CHAPTER 1
INTRODUCTION AND MOTIVATION

Maps bring order to a chaotic world. Computer vision offers the promise of both creating ubiquitous, accurate maps from low-cost, lightweight, low-power devices, and of validating such maps in real-time to make them dynamic. However, many significant challenges remain.

Mapping has more applications today than perhaps ever before, as mapping the world is a requirement for spatial intelligence applications [1]. Indoor geometric maps enable real estate professionals and virtual home buyers to create and experience immersive user experiences online. Outdoor semantic maps have wide applications in mobile robotics. Maps are especially important for safe robot operation for autonomous vehicles, where robots must reason with high accuracy about large portions of the world that are completely occluded or that are governed by strict laws that may be difficult to estimate on-the-fly. Safety is the key obstacle to deployment of self-driving vehicles, and detailed maps provide strong priors that can improve safety and adherence to laws during navigation, especially in cluttered urban environments. Even once created, such maps become stale quickly with out-of-date information. As the world is a highly dynamic place, if maps are used as hard priors, this could lead to confident but incorrect assumptions about the environment. Once built and validated, maps can be leveraged in myriad ways, from perception [2, 3, 4], to highly accurate motion forecasting [4, 5, 6, 7, 8], safe motion planning outdoors [9, 10] or indoors, and simulation [11, 12, 13]. Maps are key tools to unlocking the potential for robots to function effectively and safely in our world. Robotics applications that can benefit the lives of people all over the world are numerous, from home delivery of goods, to automation in agriculture, mining, infrastructure inspection, defense, warehouse automation, to much more. Perhaps the most impactful of all such applications will be autonomous transportation.

1.1 Unsolved Problems in Mapping

1.1.1 Challenges in Building 2d Indoor Geometric Maps

Today, creation of 2d floorplans, a type of 2d, overhead geometric map of an indoor environment, requires costly hardware or very dense sampling of images. Availability of 2d floorplans has become one of the dominant factors in the modern real estate market, with 60% of buyers said that viewing a schematic floor plan was very or extremely important in choosing a home [1]. In order to support virtual tours with paired maps, precise localization of panoramas to this type of map is required, moving indoor mapping from a side project for consumers or amateur enthusiasts, to a high-stakes, competitive endeavor. While the market for very high-end homes can support high costs, the market for low- and mid-range homes cannot support expensive capture, and the development of lower cost techniques are

[1] https://www.zillow.com/z/3d-home/floor-plans/

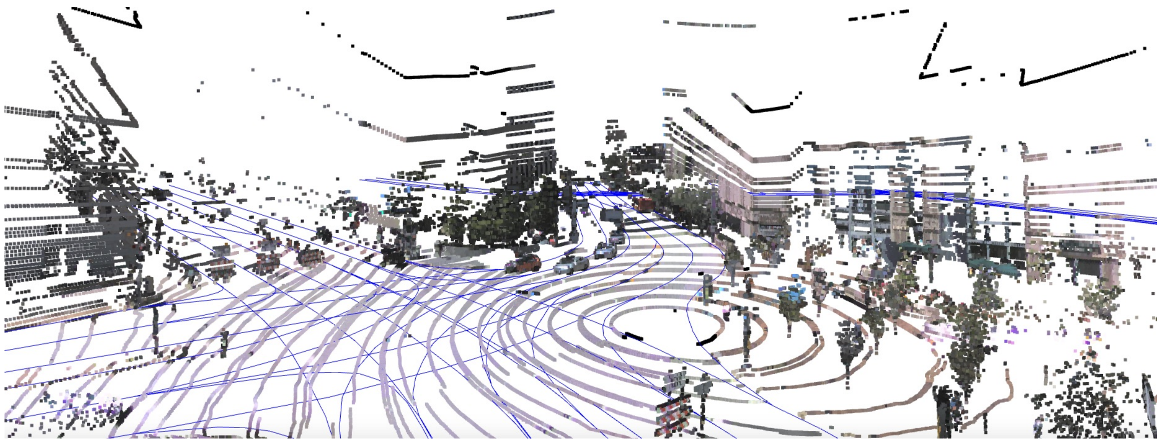


Figure 1.1: A scene of downtown Pittsburgh, USA, as featured in the Argoverse 2.0 Sensor Dataset, as captured by LiDAR and cameras, with an overlaid vector map.

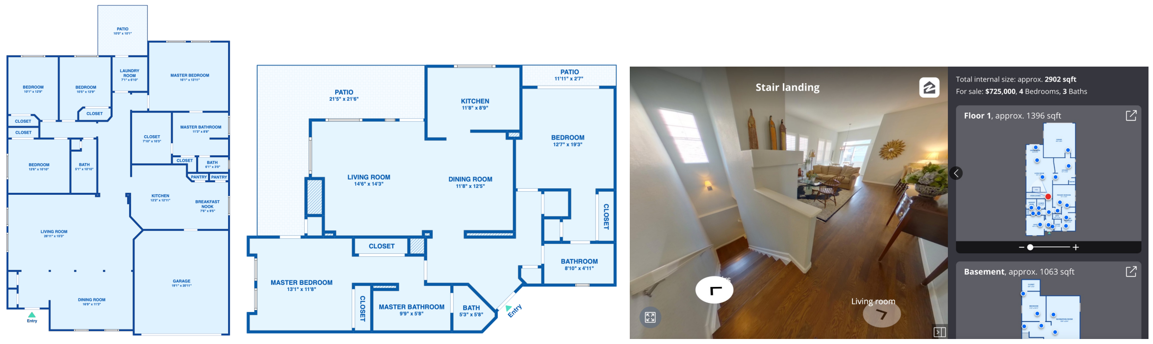


Figure 1.2: Example ground truth floorplans from the Zillow Indoor Dataset (ZInD) [14], and the Zillow *3D Home* user interface.

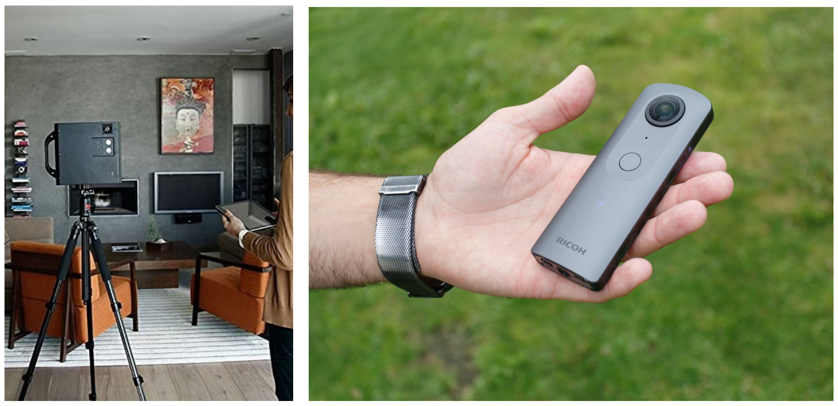


Figure 1.3: Two modern hardware options for indoor reconstruction: the Matterport Pro2 3D camera **(left)** and the Ricoh Theta V 360 camera **(right)**.

required.

Current techniques for building 2d indoor maps, i.e. floorplans, require expensive, heavyweight hardware such as Matterport cameras and protracted, tedious captures to fulfill spatial density requirements needed for registration. For example, reconstruction using Matterport Pro2 requires scanning with a tripod placed every 1.5 to 2.5 meters throughout a home, easily requiring 50-75 separate scans, and the hardware itself costs thousands of dollars [15]. On the other hand, consumer-grade cameras cost just hundreds of dollars, and offer the promise of inexpensive, immersive capture. However, reconstruction and localization from a sparse set of views with very wide-baselines, using consumer-grade panoramic cameras, is yet an unsolved problem [14]. Single-family homes present several unique challenges, among them highly variable lighting conditions between scenes, repetitive features such as windows and doors [16], and omnipresent occlusion.

1.1.2 Challenges in Creating Outdoor 3d Geometric Maps via SfM

The ability to create 3d structure for maps with image-based methods is still limited in both accuracy and scale today [17]. While self-driving applications can support LiDAR hardware for mapping that may cost a few thousand dollars, for other domains this is not true. For example, image-based reconstruction is increasingly attractive from a power consumption and hardware weight perspective, especially in the era of consumer and delivery drones. While visual SLAM is a mature field, without loop closures, it leads to inevitable drift.

The current state of affairs for SfM suggests a surprising paradox within the computer vision community. State-of-the-art SfM systems [18, 19, 20] today *still* use SIFT feature detectors and descriptors [21, 22], developed in 1999, coupled with RANSAC verifiers, despite *over 20 years* of subsequent research dedicated towards improving feature detectors, feature descriptors, and correspondence verifiers [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,

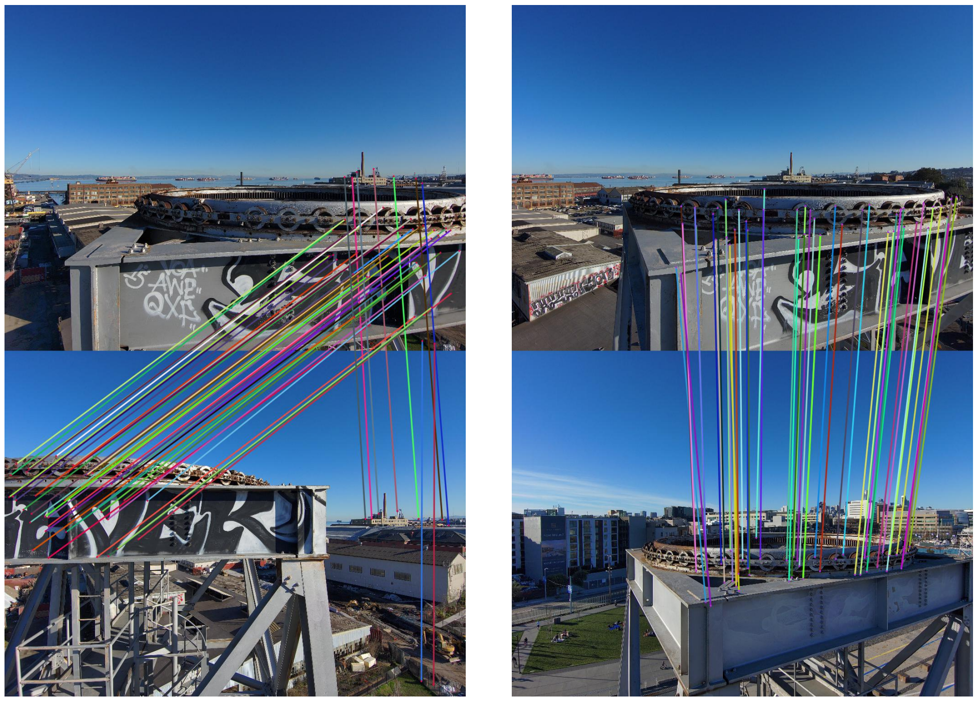
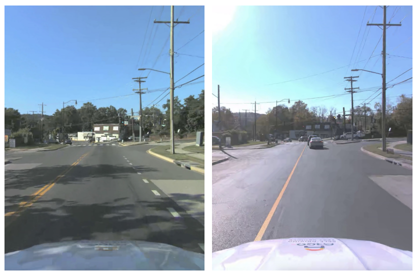


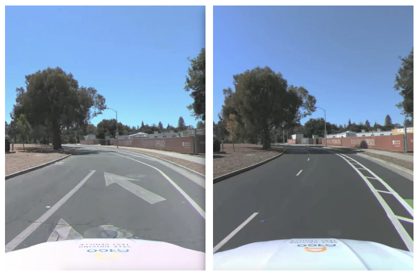
Figure 1.4: Examples of good **(left)** and poor **(right)** matches from a deep feature detector/descriptor (SuperPoint [42]) and deep matcher (SuperGlue [76]) for two image pairs, from Skydio's Crane Mast dataset. False positives **(right)** occur because of repetitive structures.

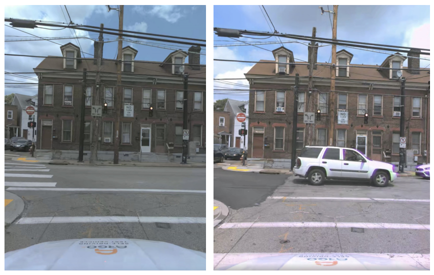
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102]. At a first glance, this trend suggests deep learning is not useful for SfM.

Incremental SfM represents the state-of-the-art [17] and is the accepted practice for image-based reconstruction (e.g. COLMAP [18]), but is slow and relies upon greedy "initial view selection" and "next best view selection" decisions when adding image pairs to a scene using PnP. Accordingly, it also offers no guarantees against drift over city-scale reconstruction, limiting its scale. Such methods require running bundle adjustment repeatedly. On the other hand, Global SfM, with implementations available in open-source libraries such as OpenMVG [19] and Theia [20], is known to be fast, but less accurate. Perhaps this is not surprising, however, as wide-baseline stereo matches are known to suffer from low signal-to-noise ratio, leaving batch optimization vulnerable to the impact of even a few punishing outliers (see Figure 1.4). These outliers can often overpower one of the benefits global SFM provides – the ability to exploit redundancy in measurements. For a dataset of N images, there can be up to $\frac{N(N-1)}{2}$ pairs for which the relative motions can be estimated, potentially providing a highly redundant set of observations can be efficiently averaged [103]. However, the community has yet to find techniques to use this redundancy to provide an advantage in accuracy.

Identifying trustworthy and dense correspondences, i.e. the "front-end" of SfM and SLAM, is widely considered to be the most challenging part of any such SfM system, and is generally unsolved in the wide-baseline case, with state-of-the-art approaches achieving only 60% mean average accuracy at a generous 10° pose error threshold [104], and achiev-


(a) Washington, D.C., USA


(b) Palo Alto, CA, USA


(c) Pittsburgh, PA, USA

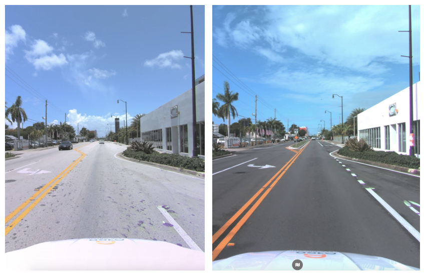

(d) Miami, FL, USA

Figure 1.5: Examples of HD map changes, i.e. scenarios when sensor data and map data are no longer in agreement with one another due to real-world changes.

ing just 22% AUC on indoor scenes with low-texture at a 5° error threshold [82]. For a real-world mobile robot system, 10° pose error could lead to disastrous consequences.

1.1.3 Challenges in Validating Outdoor HD Maps

Semantic maps are defined as geometric maps endowed with semantic labels. However, building and validating highly-accurate semantic maps in the wild, at-scale, without expensive human annotation, is yet an unsolved problem [105, 14, 17]. A particular type of semantic map, named a *high-definition map* (HD map), is particularly important for self-driving vehicles. The most important semantic labels of HD maps describe lane-level geometry, which allows such vehicles to take advantage of city infrastructure. These maps may also include possible attributes like semantic attributes, pedestrian crosswalks, all of which exhibit broad variation [4, 106, 107].

Generating vector map data in urban environments, such as lane-level geometry, is well-known to be an unsolved problem in the mapping community [105], although progress is being made (HDMapNet [108], Tesla AI [109], STSU [110]).

Semantic maps, especially HD maps, are usually valid for only a limited period of time before they become stale due to real-world changes (see Figure 1.5). Accordingly, validating such maps is in general an unsolved problem and little is known about how to perform it without massive human effort. To solve the task in a data-driven fashion, a large training dataset is necessary. However, capturing data before and after a real-world change occurs is difficult because changes occur as part of a stochastic process, making it difficult to know when and where to record data. Accordingly, a large fleet of vehicles operating continuously in the real world is required to capture a sufficient quantity of training data. In addition, no public data has ever been released for the problem, leaving the research community without the ability to fully understand and approach the problem. Some researchers have presented heuristics for data association between map elements and real world entities captured on-the-fly, but they rely upon many heuristics, making these methods brittle and unable to generalize [111, 112, 113]. Other approaches store huge amounts of past sensor data in the form of a high-definition infrared image [114], requiring gigabytes, if not terabytes, at city scale, and heuristics, with a difficult trade-off between recall and precision. Accordingly, humans are required to manually sift through petabytes of data to identify locations where maps may have been stale. This is an error-prone process, as it is hard even for humans to recognize these changes. Validating HD maps requires comparing dozens of semantic map entities with dozens of map entities in the real world, to see if any have changed.

1.2 Thesis Statement

In this work, we demonstrate how deep learning enables substantial improvements in completeness and accuracy over the state-of-the-art for building and validating maps from image input, across three domains: 2d geometric indoor maps (i.e. floorplans), 3d geometric outdoor maps, and 3d semantic HD maps.

Machine learning today is fundamentally a data engineering problem [115]. It can be well-defined as "programming by data". To make meaningful progress in machine learning research for autonomous robotics today, sourcing vast amounts of diverse, real-world data is essential. One cannot simply write a few lines of code and train a neural network on a small-scale dataset, as one might have been able to do in an academic setting several years ago. Training data requirements often range in size from terabytes to petabytes, introducing large infrastructure challenges.

1.3 Thesis Contributions

Table 1.1: Thesis contributions towards building and validating maps.

Domain	Building Maps	Validating Maps
2D geometry indoors	SALVe: Semantic Alignment Verification for Floorplan Reconstruction from Sparse Panoramas	SALVe: Semantic Alignment Verification for Floorplan Reconstruction from Sparse Panoramas
3D geometry outdoors	Deep Front Ends and GTSFM	-
3D semantics outdoors	-	Trust, but Verify: Cross-Modality Fusion for HD Map Change Detection

My thesis work is focused on developing methods that can improve the accuracy and efficiency of semantic and geometric understanding of 3D environments by exploiting learned priors. In this thesis, I present a number of new problem formulations, algorithms, and deep-learning based methods for building and validating maps. I also carry out a number of experiments to demonstrate the advantages of these data-driven, deep-learning based methods. In my work, I build and validate maps in the following ways, which advances the state-of-the-art:

For creating 2d indoor geometric maps, e.g. floorplans, I introduce a new system for automatic 2D floorplan reconstruction that is enabled by *SALVe*, our novel pairwise learned alignment verifier. This verifier validates putative bird's eye view (BEV) submaps while building the floorplan. The inputs to our system are sparsely located 360° panoramas, whose semantic features (windows, doors, and openings) are inferred and used to hypothesize pairwise room adjacency or overlap. SALVe initializes a pose graph, which is subsequently optimized using GTSAM [116]. Once the room poses are computed, room layouts are inferred using HorizonNet [117], and the floorplan is constructed by stitching the most confident layout boundaries. We validate our system qualitatively and quantitatively on the Zillow Indoor Dataset (ZinD) [14], as well as through ablation studies, showing that it outperforms state-of-the-art SfM systems in completeness by over 200%, without sacrificing accuracy. Our results point to the significance of our work: poses of 81% of panoramas are localized in the first 2 CCs connected components (CCs), and 89% in the first 3 CCs.

For creating outdoor 3d structure, I show how deep-learning can be used in the "front-end" to improve the accuracy of global SfM, and to improve the runtime over incremental SfM [118, 119]. We call this system "Georgia Tech Structure from Motion" (GTSFM).

For validating outdoor semantic/HD maps, I show how deep models can be used to effectively predict and localize map changes. This requires building new datasets for the problem, which I did [120].

1.4 Conclusion

In conclusion, taking a deep-learning based approach to mapping yields a host of benefits. In the indoor scenario, our approach enables mapping with cheap hardware, few views, and very wide baselines, representing a significant breakthrough. In the outdoor case, our approach shows promising steps towards understanding the place of deep learning in a modern global SfM system. Finally, our approach to building and validating HD maps can make them dynamic, an important missing part of mapping research today, mitigating the danger associated with a host of unsafe scenarios for mobile robots and autonomous driving.

CHAPTER 2
BUILDING 2D INDOOR GEOMETRIC MAPS: SEMANTIC ALIGNMENT VERIFICATION (SALVE) FOR FLOORPLAN RECONSTRUCTION FROM SPARSE PANORAMAS

We propose a new system for automatic 2D floorplan reconstruction that is enabled by *SALVe*, our novel pairwise learned alignment verifier. The inputs to our system are sparsely located 360° panoramas, whose semantic features (windows, doors, and openings) are inferred and used to hypothesize pairwise room adjacency or overlap. SALVe initializes a pose graph, which is subsequently optimized using GTSAM [116]. Once the room poses are computed, room layouts are inferred using HorizonNet [117], and the floorplan is constructed by stitching the most confident layout boundaries. We validate our system qualitatively and quantitatively as well as through ablation studies, showing that it outperforms state-of-the-art SfM systems in completeness by over 200%, without sacrificing accuracy. Our results point to the significance of our work: poses of 81% of panoramas are localized in the first 2 connected components (CCs), and 89% in the first 3 CCs.

2.1 Introduction

Indoor geometry reconstruction enables a variety of applications that include virtual tours, architectural analysis, virtual staging, and autonomous navigation. There are solutions for image-based reconstruction based on inputs ranging from dense image capture to sparser capture using specialized imaging equipment (e.g., Matterport Pro2). For scalability of adoption, however, data bandwidth, equipment costs, and amount of labor must be considered.

We reconstruct floorplans from sparsely captured 360° panoramas, as provided by ZInD [14]. Currently, this problem is far from solved. Traditional Structure-from-Motion (SfM) [19, 121] suffers from very limited reconstruction completeness [14, 122]. Semantic SfM has been proposed [123, 124, 16], but accuracy is still limited, typically requiring a human in the loop [14].

Indoor floorplan reconstruction from unordered panoramas is a *discrete* instance of the wide-baseline SfM problem. Unlike traditional SfM, which is associated with a continuous estimation problem, for indoor residential floorplan reconstruction, discrete room pieces must align at specific junction points (such as doors and walls), similar to solving a jigsaw puzzle [125]. We show how objects with repetitive structure, such as windows and doors, can be used to hypothesize room adjacency or overlap. Each hypothesis, i.e. a matched semantic element, provides a relative 2D room pose. The main innovation of our work is *SALVe*, a learned pairwise room alignment verifier. Given a room pair alignment hypothesis, *SALVe* uses the bird's eye view (BEV) of floors and ceilings to predict the likelihood score of adjacency or overlap. Our use of a discrete combinatorial proposal step, followed by a learned deep verifier, is akin to recent trends in language models, for tasks requiring multi-step reasoning [126, 127], as "*Verifiers benefit both from their inherent optionality*

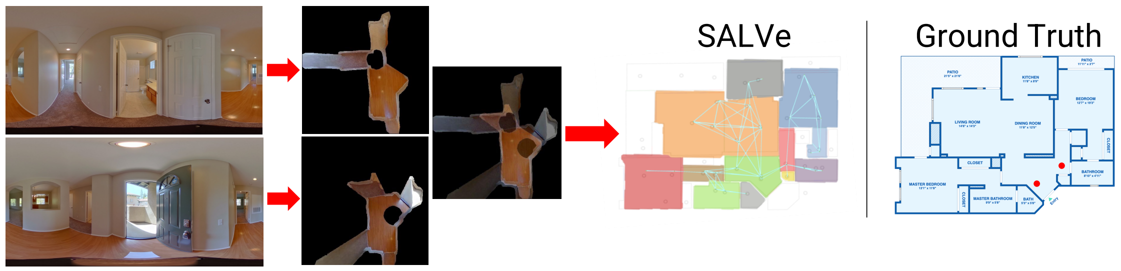


Figure 2.1: A challenging wide-baseline scenario where traditional SfM systems that rely upon keypoint feature matches struggle, but where we succeed by exploiting semantic features such as doors, windows, and openings, or W/D/O). We infer layout and hypothesize plausible pairwise relative poses, which are then accepted or rejected, by feeding top-down aligned renderings into our learned *SALVe* verifier. Our global pose estimation has high completeness, leading to dramatic improvements in floorplan reconstruction (indicated by colored regions) vs. state-of-the-art systems such as OpenMVG [19] and OpenSfM [121]. For this hallway/entryway pano pair, SALVe easily validates a relative pose that was generated by grounding on a hallway opening feature.

and from verification being a simpler task than generation in general." [126].

Once the relative poses are computed and verified, we perform global pose graph optimization using GTSAM [116]. Using the estimated poses and room layouts generated using HorizonNet [117], we construct the floorplan by stitching these layouts.

Our contributions are:

- To our knowledge, the first system for creating floorplans from unaligned panoramas with small to extremely wide baselines. These baselines can be so large that traditional SfM techniques fail.
- SALVe, a novel learning-based approach for validating discrete pairwise alignment proposals between panoramas in polynomial time.
- We show how our network verifies measurements with a high enough signal-to-noise ratio to directly apply global aggregation and optimization techniques.

2.2 Related Work

We briefly review approaches in floorplan reconstruction, SfM, and pose estimation under extreme baselines. While single-room layout estimation and depth estimation are also relevant, we do not claim novelty in these areas. Good surveys of such methods can be found in [128] and [129].

Floorplan Reconstruction. Early systems require a human in the loop [130, 131]. One notable manual approach is that of Farin *et al.* [131], which uses sparsely located 360° panoramas for joint floorplan and camera pose estimation.

For more automated solutions, SfM is used on densely captured perspective images [132] or 360° panoramas [133]. Both use SfM and MVS output to formulate graph optimization problems on a regular grid, through either graph cuts [132] or shortest-path problems [133], from which a rough 2D floorplan can be extracted. For sparser image inputs, semantic information such as floors, ceilings, and walls are used as additional cues [134]. Pintore *et* *al.* [135] cluster panoramas by room using photo-consistency at the central horizon line and plane sweeping with superpixel object masks to model clutter and floorplans in 3D. There are also methods on floorplan reconstruction from known camera poses [136, 137, 125, 138, 139] or RGBD data [136, 137, 125, 138, 139, 140, 141, 142, 143].

Structure from Motion (SfM). Much work has been done on SfM, and we refer readers to surveys such as [144]. Recently, deep learning with graph-based attention [76] or transformers [82] for deep, differentiable key point matching has been exploited to learn and match features from data. These "deep front-ends" offer a promise of less noisy input to back-end optimization [76]. Our system can be viewed as a deep verifier network (a deep front-end) that feeds measurements to global SfM [145, 20]; however, instead of requiring complex outlier rejection schemes typical of global SfM [146, 145, 19, 147, 20, 148], we show that outlier rejection can simply be based on predicted scores.

Semantic information has been used to overcome the limitation of keypoint matching for large baselines or scenes with little detail or repetitive textures [123, 149]. Cohen *et al.* [124] first introduced a combinatorial approach for 3D model registration by aligning semantic objects such as windows [16]. More recent work [14, 122] exploits this same idea to assemble floorplans from room layouts.

Extreme Pose Estimation. This refers to computing relative pose with little to no visual overlap. On localizing RGBD images, Yang *et al.* [150, 151] demonstrate scan completion to a $360°$ image, followed by feature-based registration can be useful. Chen *et al.* [152] introduce DirectionNet to estimate a distribution of relative poses in 5 DOF space, i.e., when scale is unknown. SparsePlanes [153] uses planar surface estimation from perspective views within a single room for relative pose estimation. Other CNN-based approaches on perspective image re-localization include [154, 155, 156].

In concurrent work, Shabani *et al.* [122] use semantic information to generate global pose hypotheses by synthesizing Manhattan-only floorplans. The hypotheses are then scored by ConvMPN [157] and used to produce plausible room layout arrangements along with camera poses. They assume each panorama is captured in separate but connected rooms. Another key difference from our work is that their learning-based verifier is trained to evaluate the *final floorplan arrangements*, after using heuristics to enumerate many possible solutions. This is *exponential* in the number of input panoramas. Their approach is expected to produce several layout arrangements. In contrast, *SALVe* matches semantic elements between pairs of panoramas in polynomial time. Our model is then trained to verify the individual pairwise arrangements, allowing our approach to be substituted as a front-end in any pose-graph optimization and producing a single reconstruction with higher reliability.

2.3 System Overview

We address the problem of global pose estimation of sparsely located panoramas, for the purpose of floorplan reconstruction. Formally, we define the global pose estimation problem as, given an unordered collection of n panoramas $\{\mathbf{I}_i\}$, determine poses $\{{}^w\mathbf{T}_i\}_{i=1}^n \in SE(2)$ of each panorama in global coordinate frame w. Similar to [139], we define the floorplan reconstruction problem as generating a *raster* (1) floor occupancy and (2) per-room masks.

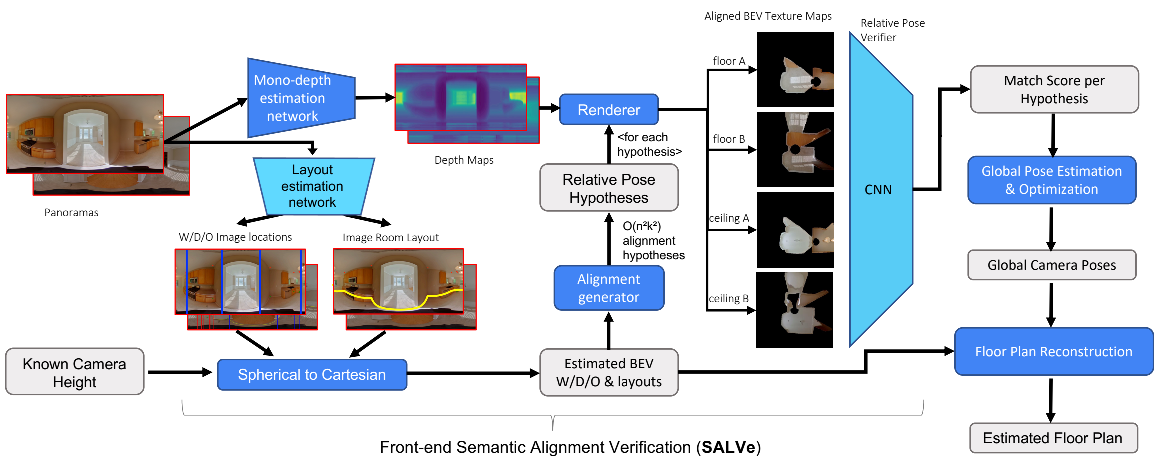


Figure 2.2: Overview of our floorplan reconstruction system. "BEV" = "bird's eye view". Blue boxes are processing components, gray boxes are data. Trapezoids denote components based on deep networks; lighter blue networks are trained by us. 'Image Room Layout' represents the image coordinates of the floor-wall boundary (at each panorama column). n is the number of panoramas and k is the average number of detected windows/doors/openings per panorama. We show rendered floor and ceiling texture maps for a consistently-aligned pair of panoramas.

Global pose estimation inherently relies on methods that build up global information from local signals. In our work, these local signals are estimated relative poses between pairs of panoramas. Our system for generating the floorplan from sparsely located panoramas is shown in Figure 2.2. The system consists of a front-end designed to hypothesize and compute relative pairwise poses, and a back-end designed to optimize global poses using these measurements.

The front-end (*SALVe*, or Semantic Alignment Verifier) first generates hypotheses of relative pose between the input pair of panoramas using their estimated room layout and detected semantic objects (specifically windows, doors, and openings, or W/D/O).[1] A hypothesis consists of pairing the same type of object across the two panoramas. Each pair of hypothesized corresponding W/D/O detections generates two relative pose hypotheses, by solving for the 2D translation that aligns their centers (on the ground plane), and the two possible rotation angles $\alpha, 180^\circ + \alpha$ that align their extents. Each pairing allows us to compute the relative SE(2) pose.

A main novelty in this paper is how we test whether a hypothesis is plausible with *SALVe*. For a hypothesized relative pose, the system renders bird's-eye views of the floor and ceiling for both panoramas in the same BEV coordinate system, which produces overlapped top-down renderings. The rendering is computed with per-panorama depth distribution estimation using HoHoNet [158]. Then we use a deep CNN with a ResNet [159] backbone to generate a likelihood score that the overlapped images are a plausible match.

Implausible matches are discarded, and from the remaining plausible matches we construct a pose graph. The back-end then globally optimizes the constructed pose graph

[1]Openings are constructs that divide a large room into multiple parts [14].

using GTSAM [116]. Finally, floorplans are created by clustering the panoramas by room, extracting the most confident room layout given predicted panorama poses, and finally stitching these room layouts.

2.4 Approach

In this section, we detail the steps taken to generate a 2D floorplan from sparsely distributed 360° panoramas. The first step is to generate alignment hypotheses between pairs of panoramas.

2.4.1 Assumptions

We assume the inputs are a set of unordered 360° panoramas, captured from an indoor space. The images cover the entire space and the connecting doors between different rooms. Neighboring images may or may not have visual overlap. We assume the panoramas are in equirectangular form, i.e., their fields of view are 360° (horizontal) and 180° (vertical). The camera is assumed to be of known height and fixed orientation parallel to the floor[2], so pose is estimated in a 2D bird's-eye view (BEV) coordinate system.

2.4.2 Generating Alignment Hypotheses

Since our floorplan is 2D, alignment between pairs of panoramas has 3 DOFs (horizontal position and rotation). Scale is not a free parameter, assuming known, fixed camera height and a single floor plane (see [161] or our Appendix for a derivation). To handle wide baselines, we use semantic objects (windows, doors, and openings, or W/D/O) to generate alignment hypotheses. While this is similar to the W/D/O-based room merge process in [14], we additionally make use of estimated room layout. Each room layout is estimated using a modified HorizonNet model [117]; it is trained with partial room shape geometry to predict both the floor-wall boundary with an uncertainty score and locations of W/D/O.

Each alignment hypothesis is generated with the assumption that W/D/O being aligned are in either the same room or different rooms. The outward surface normals of W/D/O are either in the same or opposite directions; we assume a window can only be aligned in the direction of its interior normal, while a door or opening could be aligned in either direction. The hypothesis for rotation is refined using dominant axes of the two predicted room layouts.

Exhaustively listing pairs of W/D/O can produce many hypotheses for alignment verification. We halve the combinatorial complexity by ensuring that each pair of matched W/D/O have widths with a ratio within $[0.65, 1.0]$, i.e. a door that is 2 units wide cannot match to a door that is 1 units wide. Once the alignment hypotheses are found, they need to be verified.

[2]We achieve this orientation assumption via pre-processing that straightens the panoramas using vanishing points [160].

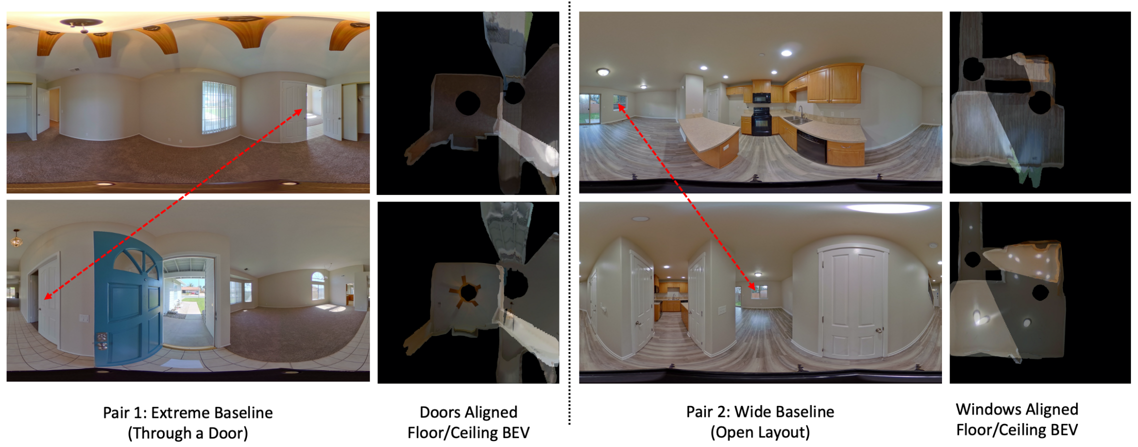

Pair 1: Extreme Baseline (Through a Door)
Doors Aligned Floor/Ceiling BEV
Pair 2: Wide Baseline (Open Layout)
Windows Aligned Floor/Ceiling BEV

Figure 2.3: Generating training samples. Orthographic BEVs of given panoramas, after semantic alignment proposal. Red arrows indicate the W/D/O, used to generate the pose proposals. ***Column 1:*** Example of extreme baseline pair. ***Column 2:*** overlaid floor **(top)** and ceiling **(bottom)**. ***Column 3:*** Example of a wide baseline pair. ***Column 4:*** overlaid floor **(top)** and ceiling **(bottom)**.

2.4.3 SALVe: Semantic Alignment Verifier

While domain knowledge of indoor space such as room intersections and loop closure can be helpful in constructing the floorplan [14], visual cues can also be used to verify pairwise panorama overlap [162]. We use bird's eye views (BEVs, which are orthographic) of the floor and ceiling as visual cues for alignment verification. Given the significant variation in lighting and image quality across panoramas, traditional photometric matching techniques may not be very effective. Instead, we train a model to implicitly verify spatial overlap based on these aligned texture signals.

We extract depth using HoHoNet[158], which is used to render the BEVs. Example views can be found in Figure 2.3. Given an alignment hypothesis, we map the BEVs of the floor and ceiling for both panoramas to a common image coordinate system. The four stacked views are then fed into our deep-learning based pairwise alignment verification model to classify 2-view alignment. Given n panoramas, each with k W/D/O, $\mathcal{O}(n^2k^2)$ alignments are possible and thus need to be verified.

SALVe uses a ResNet [159] ConvNet architecture as the backbone for verification. Its input is a stack of 4 aligned views (2 from each panorama), with a total of 12 channels. It is trained with softmax-cross entropy over 2 classes, representing the "mismatch" and "match" classes. We generate these classes by measuring the deviation of generated relative poses (alignments from window-window, opening-opening, or door-door pairs) against the ground truth poses. Those below a certain amount of deviation are considered "matches", and all others are considered "mismatches".

2.4.4 Global Pose Estimation and Optimization

SALVe is used to generate a set of pairwise alignments, which are used to construct a pose graph; its nodes are panoramas and edges are estimated relative poses. The pose graph has

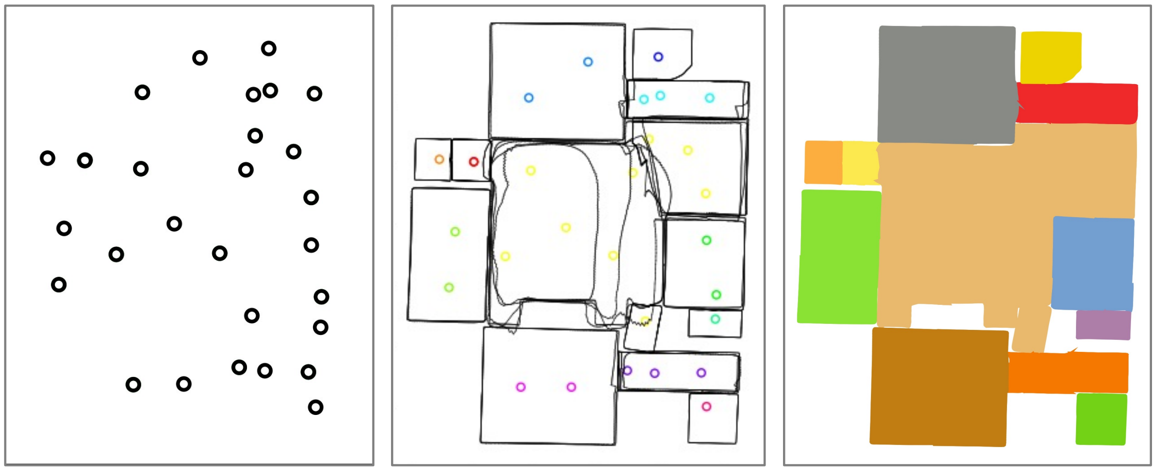


Figure 2.4: An example of different stages of floorplan reconstruction: ***Left***: Estimated positions of panorama centers. ***Center***: Grouped panoramas with estimated dense room layouts. Panorama centers with the same color are part of the same group. Notice that each open space is grouped together. Distinct groups correspond largely to physical rooms separated by doors. ***Right***: The final floorplan after highest-confidence contour extraction is applied to each group. Each contour is filled with a unique color.

an edge between any two panoramas $\mathbf{I}_{i_1}$ and $\mathbf{I}_{i_2}$ where pairing a detection $\mathbf{d}_{k_1}^{i_1}$ with detection $\mathbf{d}_{k_2}^{i_2}$ yields a plausible (according to SALVe) alignment. A detection may participate in multiple edges e.g., pairing $(\mathbf{d}_{k_1}^{i_1}, \mathbf{d}_{k_2}^{i_2})$ may add an edge between i_1 and i_2, and pairing $(\mathbf{d}_{k_1}^{i_1}, \mathbf{d}_{k_3}^{i_3})$ may add an edge between panos i_1 and i_3. Although conflicting relative pose hypotheses are possible, in practice SALVe is a sufficiently accurate verifier that they are quite rare.

When multiple disjoint graphs result, we only consider the largest connected component. We experiment with two algorithms for global localization: spanning tree pose aggregation and pose graph optimization (PGO) with a robust noise model, detailed in the Appendix.

2.4.5 Floorplan Reconstruction

Figure 2.4 shows the progression of floorplan reconstruction, from estimated panorama poses and room layouts to the output. There are three steps: panorama room grouping, highest confidence room contour extraction, and floorplan stitching. To refine a room layout, we first identify all the panoramas within that room; this is done using 2D IoU. Since each panorama has its own layout with local shape confidence (subsection 2.4.2) within a room, we extract a single global layout by searching for the most confident contour points. The search is done by raycasting from panorama centers and voting for the most confident contour point along each ray. The final floorplan is found by taking the union of (stitching) all room layouts. Details are in the Appendix.

2.5 Experimental Results

In this section, we explain why we use ZInD [14], provide implementation details, and describe our metrics before showing results for different global pose estimation techniques. We also describe ablation studies that show how different types of inputs affect the results.

2.5.1 Use of ZInD [14]

In order to evaluate every part of our approach, as well as the entire system, we use the recently released Zillow Indoor Dataset (ZInD) [14]. ZInD has all the required components: (1) *large scale* with $67,448$ panoramas taken in $1,575$ real homes; (2) *multiple localized panoramas per-room* with 42 panoramas over 15 rooms per-home on average; (3) *layout and W/D/O* annotations including complex, non-Manhattan layouts and (4) *2D floor-plans* with 1.8 number of floors per-home on average. We use the official train, val, and test splits that contain 1260, 157, and 158 homes, and 2168, 278, 291 floors respectively. We acknowledge that in ZInD most rooms are unfurnished, but this is a frequent scenario in the domain of real estate floor plan reconstruction. While there are other real [163, 164, 122] and synthetic [165, 166] indoor datasets, none of them have all the required components. Structured3D [165] is a synthetic dataset with only one panorama per room and doors in almost all rooms are closed (uncommon in real estate capture scenarios); these factors result in a significant change of modality.

2.5.2 Implementation Details

Layout and W/D/O estimation. We use a modified version of HorizonNet [117], trained to jointly predict room layout as well as 1D extents of W/D/O. We trained the joint model on ZInD, and will share the predictions upon publication.
Verifier supervision. We consider a pair-wise alignment to be a "match" if ground truth relative pose $(x, y, \theta) \in SE(2)$ and generated relative pose $(\hat{x}, \hat{y}, \hat{\theta}) \in SE(2)$ differ by less than 7° (θ) for doors and windows, and less than 9° for openings. A larger threshold is used for openings because there is more variation in their endpoints. We also require that $\left\|[x, y]^\top - [\hat{x}, \hat{y}]^\top\right\|_\infty < 0.35$ in normalized room coordinates (i.e., when camera height is scaled to 1).
Texture mapping. When texture mapping an orthographic view using the monocular estimated depth map from [158], we use all 3D points $\geq$ 1m below the camera for rendering the floor, and all points $\geq$ 0.5m above the camera for rendering the ceiling. We render a 10×10m region, using a resolution of 0.02 m/pixel, creating a 500×500 image.
Verifier data augmentation. We resize BEV texture maps to 234×234 resolution, sample random 224×224 crops, randomly flip them, and then normalize crops using the ImageNet mean and standard deviation.
Verifier training. We use a ResNet-152 architecture with ImageNet-pretrained weights, training for 50 epochs, with an initial learning rate of 1×10^{-3}, polynomial learning rate decay with a decay factor of 0.9 per iteration, and a weight decay of 1×10^{-4}. We use a batch size of 256 examples on 3 NVIDIA Quadro RTX 6000 GPUs.

2.5.3 Evaluation Metrics

In order to evaluate our entire system, we measure the performance of each portion of the system separately.

Layout estimation and W/D/O detection accuracy. To evaluate the quality of the layout estimation, we report 2D IoU between the predicted and ground truth room layouts per panorama. Because we project 1D W/D/O on the predicted layout, we use 1D IoU to measure the accuracy of those semantic elements, with F1 score evaluated at a true positive 1D IoU threshold of 70%.

Relative pose classification accuracy. We report intermediate metrics of the system, such as how accurate the model is at discerning between correct and inaccurate alignments. We use mean accuracy over two classes, as well as precision, recall, and F1 score.

Global pose estimation accuracy and completeness. We first align an estimated pose graph $\{\hat{\mathbf{T}}_i\}_{i=1}^{M}$ to a ground truth pose graph $\{\mathbf{T}_i\}_{i=1}^{N}$ where $\mathbf{T}_i \in SE(2)\ \forall i \in 1, \dots, N$, by estimating a $\mathrm{Sim}(2)$ transformation between them, where $M \leq N$, since not all poses may be estimated. To reduce the influence of outliers for mostly-correct global pose estimates, we perform pose graph alignment in a RANSAC loop, with a randomly selected subset ($^2/_3$ of the M estimated poses) used to fit each alignment hypothesis, over 1000 hypotheses. We then measure the distance between the predicted and true i'th camera location $\|t_i - \hat{t}_i\|_2$, and difference between true and predicted i'th camera orientation $|\theta_i - \hat{\theta}_i|$. Completeness is essential to floorplan reconstruction, so we also report the percent of panoramas localized in the largest connected component.

Floorplan reconstruction accuracy and completeness. We measure the 2D IoU between a rasterized binary occupancy map of the ground truth and the predicted floorplans. This metric measures the quality of our end-to-end system, as it encapsulates the *accuracy* of our *pair-wise relative pose proposal* in combination with the *accuracy* and *completeness* of the global pose estimation and the fusion of the room layouts (see Appendix for more details).

2.5.4 Layout and W/D/O Estimation Accuracy

The layout estimation module used in the system yields an average of 85% IoU with ground truth shape. W/D/O detection is accurate; at a 70% 1D IoU threshold, we correctly identify W/D/O with F1 scores of 0.91, 0.89, and 0.67, respectively. Our model is the least accurate in predicting openings. As discussed in [14], there are issues with annotator error and possibly ambiguous tagging of rooms in open spaces that cover different room types, making locations of openings less clear. We speculate that these contribute to the errors, especially for openings. In the Appendix, we provide qualitative examples of the various types of failure modes of the model.

2.5.5 Relative Pose Classification

We first measure the performance of the SALVe "front-end". These trained models achieve 92-95% accuracy on the test split (see Appendix). We show that a larger capacity model than ResNet-50 (i.e. ResNet-152) further improves performance. We also note that the accuracy is limited by noisily-generated 'ground truth'. We train on 587 number of tours

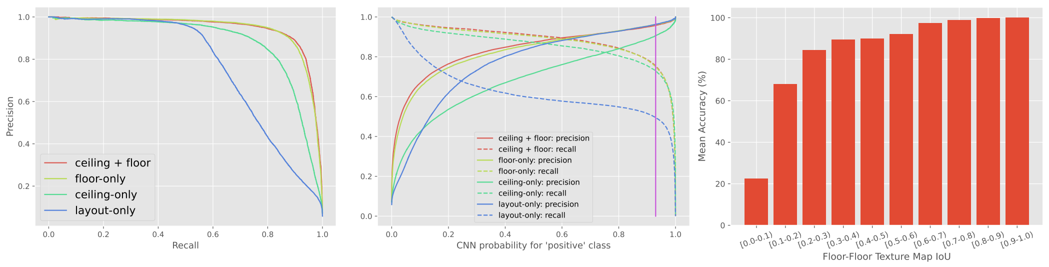


Figure 2.5: Precision-recall analysis of SALVe. ***Left***: curve for SALVe under different inputs ('layout-only' refers to a model with access only to estimated room geometry, but no floor or ceiling texture). ***Center***: Comparison of confidence thresholds versus their effect on precision and recall. The purple line indicates our operating point (93% confidence). ***Right***: Classification accuracy vs. visual overlap for the GT **positive** class only from SE(2) alignments generated from predicted W/D/O's. Small visual overlap often corresponds to "extreme" baselines.

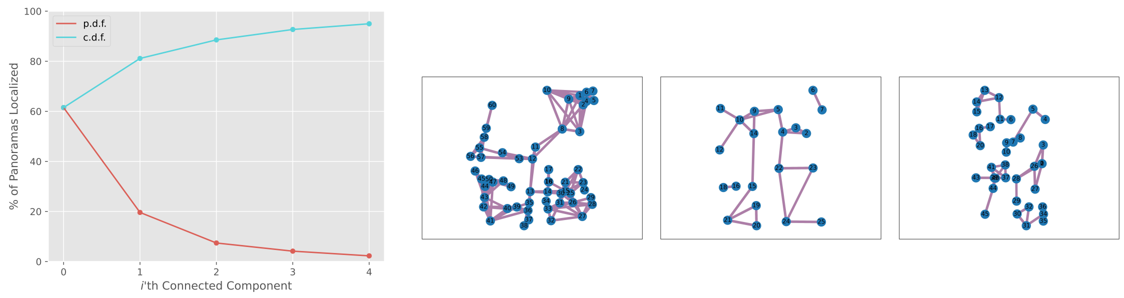


Figure 2.6: ***Left:*** Distribution of localization percentage in the first 5 connected components, averaged over all test tours. ***Right:*** Topology of global pose graphs for various different homes.

from ZInD, and use the official train/val/test splits.

In Figure 2.5, we show a PR curve, indicating the precision of the model at different recall thresholds. We choose a 93% confidence threshold as our operating point, as it maximizes precision just before a precipitous drop in recall.

How does the amount of visual overlap affect relative pose classification accuracy? More overlap yields higher accuracy for the ground truth positive class, but lower accuracy for the ground truth negative class. In Figure 2.5, we analyze the performance of our relative pose classification method under varying amounts of visual overlap. 100% overlap would indicate that two panoramas were captured in exactly the same position, with the scene unchanged between the two captures. On the other hand, 0% overlap would indicate that the panoramas were captured in completely different locations, i.e. in two rooms, on opposite sides of a closed door (an example of an "extreme" baseline). We use a proxy metric, IoU of the texture map generated using HoHoNet-estimated [158] monocular depth, which introduces some amount of noise.

2.5.6 Global Pose Estimation Results

Next, we measure performance of both the "front-end" along with some form of global aggregation ("back-end"). We compare with two baselines from state-of-the-art structure

Table 2.1: Results of global pose estimation on the ZinD test set. Two global aggregation methods are evaluated: spanning tree ('ST'), and pose graph optimization ('PGO'), with axis-alignment ('AA'). ST and PGO both use the same largest connected component of $\mathcal{G}$ as input, and thus localize an equal number of panoramas.

METHOD	LOCALIZATION %		TOUR AVG. ROTATION ERROR (DEG.)		TOUR AVG. TRANSLATION ERROR (METERS)	
	MEAN	MEDIAN	MEAN	MEDIAN	MEAN	MEDIAN
OPENSFM [121]	27.62	22.22	9.52	0.36	1.88	0.12
OPENMVG [145, 19]	13.94	8.70	3.84	0.37	**0.41**	**0.10**
OURS (W/ ST + AA)	**60.70**	**57.10**	**3.69**	**0.03**	0.81	0.26
OURS (W/ PGO + AA)	**60.70**	**57.10**	3.73	0.17	0.80	0.25

from motion systems that support optimization from $360°$ images.

OpenMVG [145, 19]. We use the recommended setting for $360°$ image input, with incremental SfM using an upright SIFT feature orientation, an upright 3-point Essential matrix solver with A-Contrario RANSAC, following the planar motion model described by [161, 167, 168], with an angular constraint for matching.

OpenSfM [121]. Incremental SfM system that uses the Hessian-Affine interest point detector [61], SIFT feature descriptor [22], and RANSAC [169].

In Table 2.1, we show the results of global pose estimation on the ZInD test set. We outperform OpenMVG by 656% and OpenSfM by 257% in the median percentage of panoramas localized (their 8.7% and 22.2% vs. our 57.1%), with even lower median rotation error (our $0.17°$ vs. their $0.37°$ and $0.36°$). Our median translation error is comparable (our 25 cm vs. their 12 cm and 10 cm). PGO is significantly more accurate than spanning tree when VP estimation is not employed (see Table 2.3). However, when using vanishing point-based dominant axis-alignment, both spanning trees and pose graph optimization on SALVe-verified measurements produce similar global aggregation results. In the left column of Figure 2.7, we show the topological structure of the largest component of the pose graph for a few homes.

2.6 Discussion

Is deep learning necessary for verification, or can heuristics be used? To verify pairwise alignment, matching texture is necessary but hard to feature engineer. Using geometry alone is insufficient (See Figure 2.5(a-b) and Table 2.2), motivating others to explore graph neural networks for the task [122]. We implemented several classifying BEV image pairs via cross-correlation scores, including FFT cross-correlation [170], and they do not work well due in part to difficulty in choosing thresholds. We implemented such a rule-based baseline, and found the results to be near random. Previous works such as LayoutLoc [14] have explored rule-based checking, but found that it only can be successful when given access to *oracle* within-room pano grouping information; estimation of such within-room grouping (i.e. adjacency) is itself one of the fundamental challenges of global pose estimation in an indoor environment.

What type of semantic object is most useful for alignment in this semantic SfM prob-

Table 2.2: Results of ablation experiments on how inputs to SALVe affect global pose estimation accuracy and completeness. Pose graph optimization and vanishing point-based axis alignment ('PGO + AA') are utilized for all entries below.

W/D/O Inputs			Raster Inputs			Localization %		Tour Avg. Rotation Error (deg.)		Tour Avg. Translation Error (meters)	
Doors	Windows	Openings	Floor Texture	Ceiling Texture	Layout	mean	median	mean	median	mean	median
✓	✓	✓	✓	✓		60.70	57.14	3.73	0.17	0.80	0.25
✓			✓	✓		43.30	40.00	2.41	0.07	0.59	0.20
	✓		✓	✓		15.57	13.33	2.20	**0.00**	0.74	**0.11**
		✓	✓	✓		23.87	23.08	**0.66**	0.05	**0.34**	0.18
✓	✓	✓	✓			60.64	58.33	3.75	0.15	0.91	0.25
✓	✓	✓		✓		**60.93**	**64.58**	10.94	0.28	2.12	0.35
✓	✓	✓			✓	19.19	16.67	3.43	0.03	0.53	**0.11**

lem? Doors, but all are essential. Openings are the second-most effective object type to achieve complete localization, and windows are least effective. Among the alignments that the model predicts to be positives with confidence $\geq 97\%$, we find that 63% originate from door-door hypotheses, while 24% originate from opening-opening hypotheses, and 20% originate from window-window hypotheses. While rooms in residential homes are very rarely connected by a window, these window alignments can provide additional redundancy, or ground alignments in very large open spaces when doors are not visible as in Figure 2.3, pair 2. In Table 2.2, we report global pose estimation results when only one type of semantic object is used to create the edges $\mathcal{E}$ of the relative pose graph $\mathcal{G}$.
To what extent is the pose graph shattered into multiple clusters? Typically, the first three connected components contain 61%, 20%, and 7% of all panoramas (See Figure 2.6a). We measure the distribution of connected components (CCs), as global pose estimation relies upon a single CC (we use the largest), and we find that often the second and third largest CCs are also large, indicating the potential for merging, e.g. combining ideas from [122] or [171]. We compute an average probability density function and cumulative density function by averaging per-floor distributions across the test set.

Is the RGB photometric signal from panoramas actually necessary, as opposed to solely using geometric context? Yes, the RGB texture is essential. In Table 2.2, we show that using a layout-only rasterization as input to the CNN, instead of a photometric texture map, leads to severe performance degradation.

Does floor or ceiling texture provide a more useful signal for alignment classification? Floor texture. However, using both signals jointly improves performance. In Table 2.2, we show the results of using as input to the network only the floor texture maps, or only the ceiling texture maps, as opposed to reasoning about both jointly.

Is a Manhattan world assumption helpful? For pose estimation, yes, but for shape estimation, no. Many rooms at critical junctures in the floorplan are non-Manhattan in shape, and 'Manhattanizing' them would be destructive when chaining together. However, room organization in a home is usually tied to three dominant, orthogonal directions. In Table 2.3, we show that using vanishing point estimation to align relative poses up to a $15°$ correction significantly improves both global pose estimation accuracy and slightly improves floorplan reconstruction accuracy. Both vanishing point relative rotation angle correction and

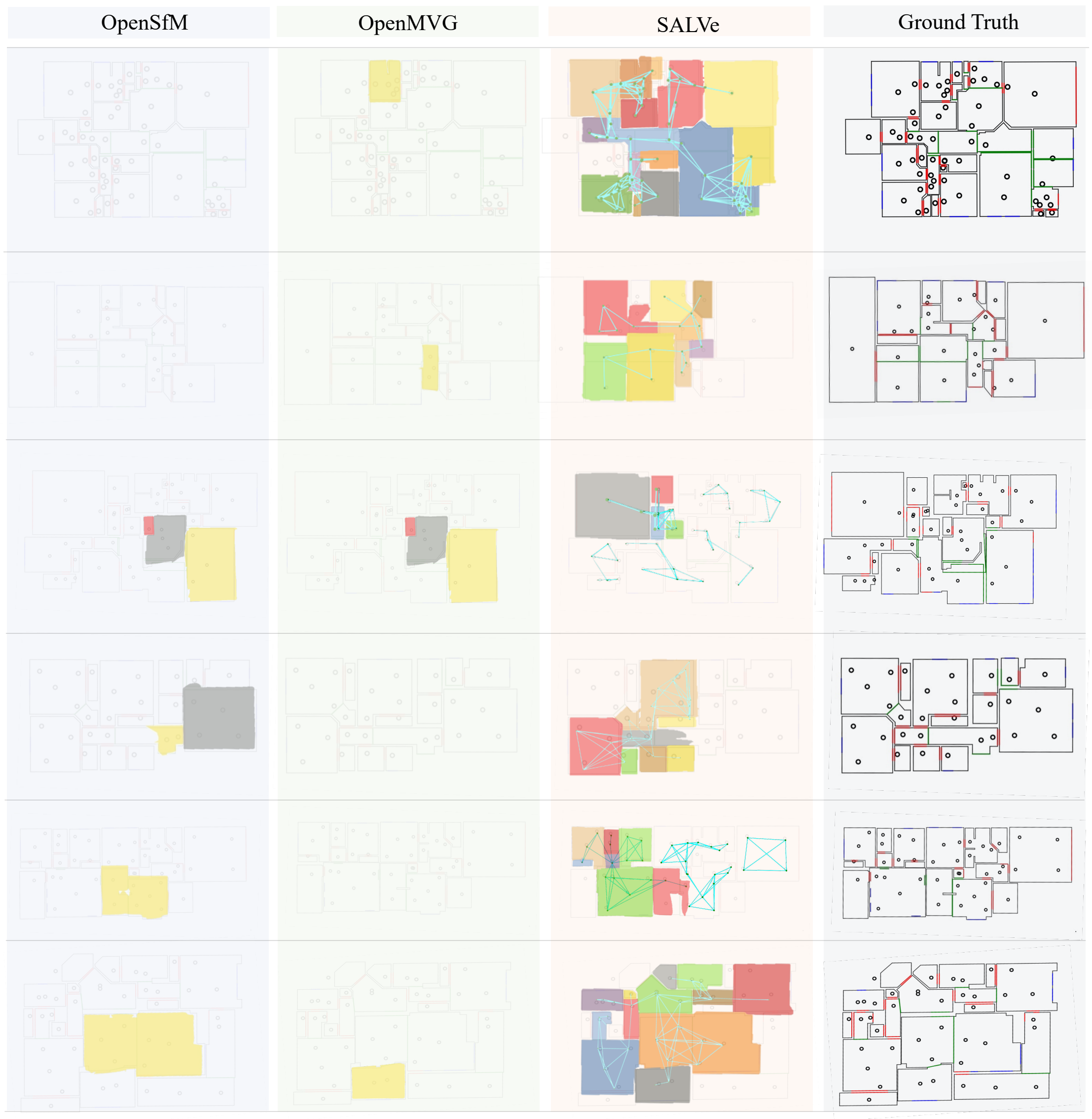


Figure 2.7: Qualitative comparison of floorplan results. ***Column 1:*** OpenSfM. ***Column 2:*** OpenMVG. ***Column 3:*** Ours. ***Column 4:*** Ground truth floorplan. All results are superimposed on the ground truth floorplan. Colored regions indicated the reconstruction result; at times, the baselines localize no panos. Our floorplan recall is significantly better than the state-of-the-art. Each row corresponds to a single floor of a different home. Colored lines represent W/D/O objects – doors, openings and windows. The multiple cyan edges in the overlaid graph correspond to verified W/D/O alignment hypotheses. For an open layout, a successful case often involves edges from panoramas in many different rooms to a single pano. These examples are intended to offer an even-handed selection of reconstructions that indicate both good performance as well as areas for improvements. Rows 1 and 6 illustrate good reconstructions. Row 2 illustrates a more challenging case with only 1-2 panos in most rooms. Rows 3-5 are more challenging as they include bottlenecks in the actual physical layout, which is critical in joining connected components.

Table 2.3: Comparison of results with and without axis-alignment ('AA') of relative poses (via vanishing angles) before global aggregation. The amount of panoramas localized is unaffected, as adjacency is maintained during the correction. For this comparison, 'oracle' layouts are used to isolate the effect of pose error. With vanishing point (VP) information, the difference between PGO and Spanning Tree is not statistically significant (1 cm and $0.04°$ error on average).

METHOD	TOUR AVG. ROTATION ERROR (DEG.)		TOUR AVG. TRANSLATION ERROR (METERS)		FLOORPLAN IoU	
	MEAN	MEDIAN	MEAN	MEDIAN	MEAN	MEDIAN
Spanning Tree	5.41	1.92	0.86	0.33	0.55	0.52
Spanning Tree + AA	**3.69**	**0.03**	0.81	0.26	**0.56**	0.52
PGO	4.93	1.53	0.81	0.29	**0.56**	0.52
PGO + AA	3.73	0.17	**0.80**	**0.25**	**0.56**	**0.53**

Table 2.4: Floorplan reconstruction results against the ground truth manually annotated floorplan. Floorplan 2D IoU is measured in the bird's eye view. The IoU is measured on the largest connected component. 'AA' represents axis-alignment.

METHOD	GLOBAL POSES		LAYOUT		FLOORPLAN IoU	
	ORACLE	ESTIMATED	ORACLE	ESTIMATED	MEAN	MEDIAN
OPENSFM		✓	✓		0.29	0.26
OPENMVG		✓	✓		0.16	0.07
OURS	✓			✓	**0.94**	**0.95**
OURS (PGO + AA)		✓	✓		0.56	0.53
OURS (PGO + AA)		✓		✓	0.49	0.45

pose graph optimization are effective means of decreasing the rotation error. In the Appendix we show how using ground truth layout (near-perfect shape) and W/D/O locations affects performance, as an upper-bound on performance of the first module in our system.

2.6.1 Floorplan Reconstruction Results

Next, we compare performance of the entire floorplan reconstruction system. In Table 2.4, we demonstrate that compared to traditional SfM with oracle room layout and oracle scale, our end-to-end system is able to produce more accurate floorplans with estimated room layouts (our 0.49 mean IoU vs. OpenSfM's 0.29 and OpenMVG's 0.16). The 0.56 mean IoU score using our estimated global poses and oracle layout primarily reflects the completeness of our final floorplan. With oracle pose and estimated room layouts, the 0.94 mean IoU reflects the accuracy of our layout estimation and stitching stages. This baseline has significantly larger IoU in part because the 'oracle' poses are provided for *all* panoramas (see the Appendix for comparison visualizations).

Qualitative Results. Figure 2.7 provides qualitative results for a number of different homes. For floors of some homes, our method produces nearly complete reconstructions, while for others, the results are more sparse. As shown by the third column of Fig. Figure 2.7, the topology of the pose graph directly affects the completeness of the reconstruction; when multiple large connected components appear, the reconstruction is shattered apart. For several homes, OpenMVG and OpenSfM fail to converge, localizing no panoramas.

2.7 Conclusion

We present a new system for automatic 2D floorplan reconstruction from sparse, unordered panoramas. This work represents a breakthrough in the completeness of reconstructed floorplans, with over two times more coverage than previous systems [121, 19], without sacrificing accuracy. We demonstrate how *SALVe*, our novel pairwise learned alignment verifier, capitalizes on the mature field of semantic detection of features (W/D/O) to handle a tractable number of alignment hypotheses and generate high-quality results. A human annotator may use it to accelerate labeling by automatically generating the majority of necessary decisions before making the final choices about glueing connected components. Figure 2.7 only illustrates the largest CC; other CCs are also generated, but not shown (Figure 2.6, a CDF of 89% for the first 3 CCs).
Limitations. Because the number of pairwise alignments is combinatorial in the number of W/D/O, the runtime of the current system is limited, although we have not heavily optimized it. As ZInD [14] contains only unfurnished homes, our system has not yet been evaluated in a furnished home regime, due to dataset availability. Camera localization completeness is still in the 55-60% range. With future improvements to each part of the system, especially omnidirectional depth estimation and layout estimation, we expect floorplan reconstruction performance to continue to improve.

2.8 Appendix

In this Appendix, we provide additional analysis and implementation details. In section 2.9, we provide qualitative comparisons of our floorplan reconstructions, vs. an upper-bound *oracle* baseline that uses ground-truth global pose estimation. In section 2.10, we provide quantitative analysis of SALVe's relative pose classification accuracy with various input modalities. In section 2.11, we provide pseudo-code for our layout stitching algorithm. In section 2.12 and section 2.13, we report detailed quantitative analysis of W/D/O detection accuracy, and W/D/O and layout estimation failure cases. In section 2.14, we describe the coordinate systems used in our work. In section 2.15, section 2.16, section 2.17, we provide additional implementation details about rendering, vanishing-point based axis alignment, and pose graph optimization and spanning tree aggregation. In section 2.18, we describe ablation experiments that compare the use of ground truth W/D/O and ground truth layout, vs. estimated W/D/O and estimated layout. Insection 2.19, we provide additional discussion about our evaluation procedures versus those of concurrent work [122]. In section 2.20, section 2.21, section 2.22, section 2.23, and section 2.24, we provide additional analysis and further examples of positive and negative training examples.

2.9 Qualitative Results: Predicted vs. Oracle Poses

In this section, we provide qualitative comparisons with a baseline that stitches predicted layouts placed at 'oracle' global pose locations (referred to by subsection 2.6.1 of the main paper). For this baseline, the high fidelity of reconstructed shapes (middle column of Figure 2.8 and Figure 2.9) demonstrates the maturity of modern layout estimation networks. This baseline also illustrates the impact of global pose estimation on the entire system.

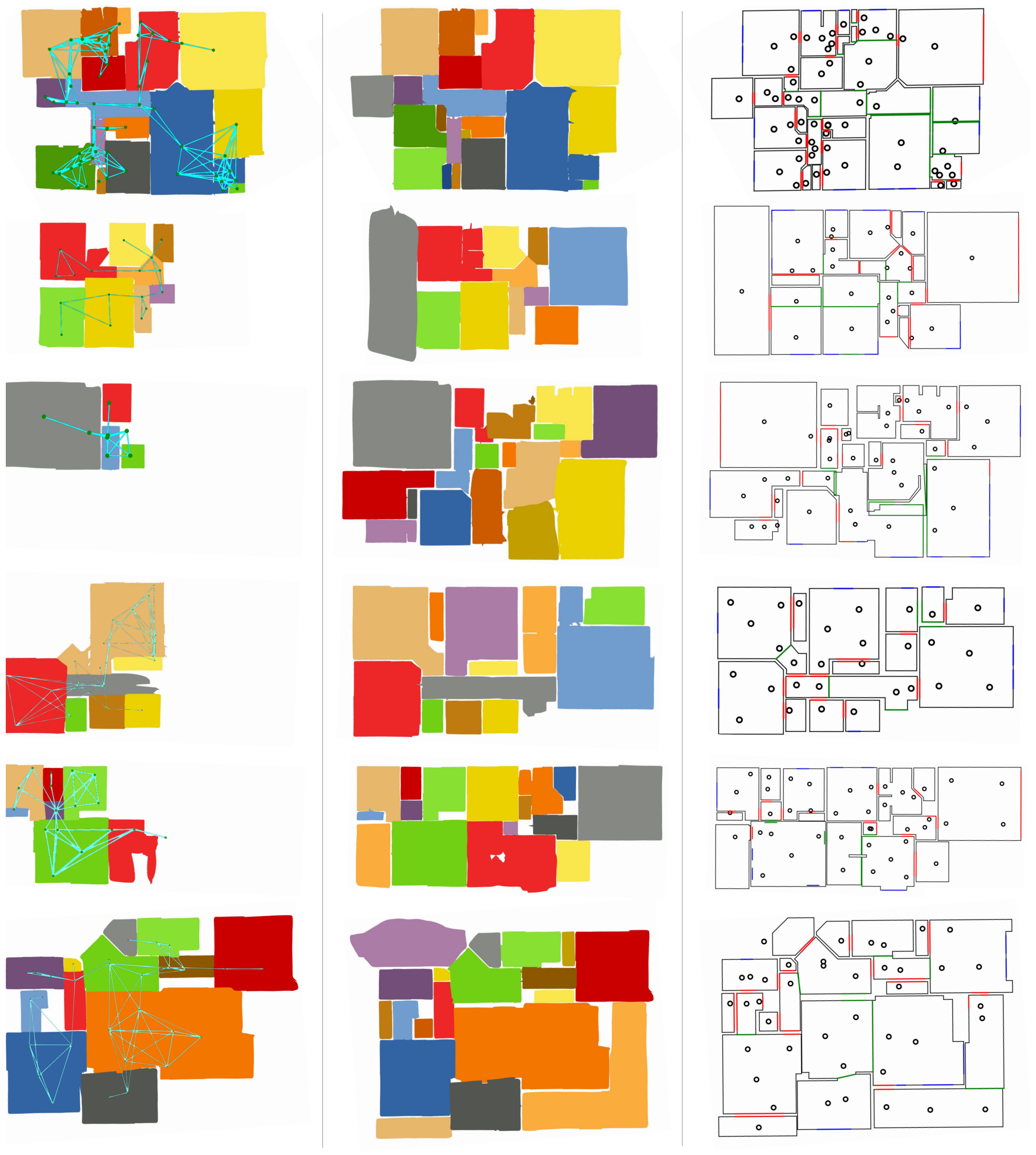


Figure 2.8: Example floorplan results of varying completeness, comparing SALVe's performance vs. an upper bound (perfect global pose estimation). ***Left:*** predicted poses of the largest connected component and predicted room layout. ***Middle:*** oracle (ground truth) poses and predicted room layout. ***Right:*** ground truth floorplan with positions of captured panoramas.

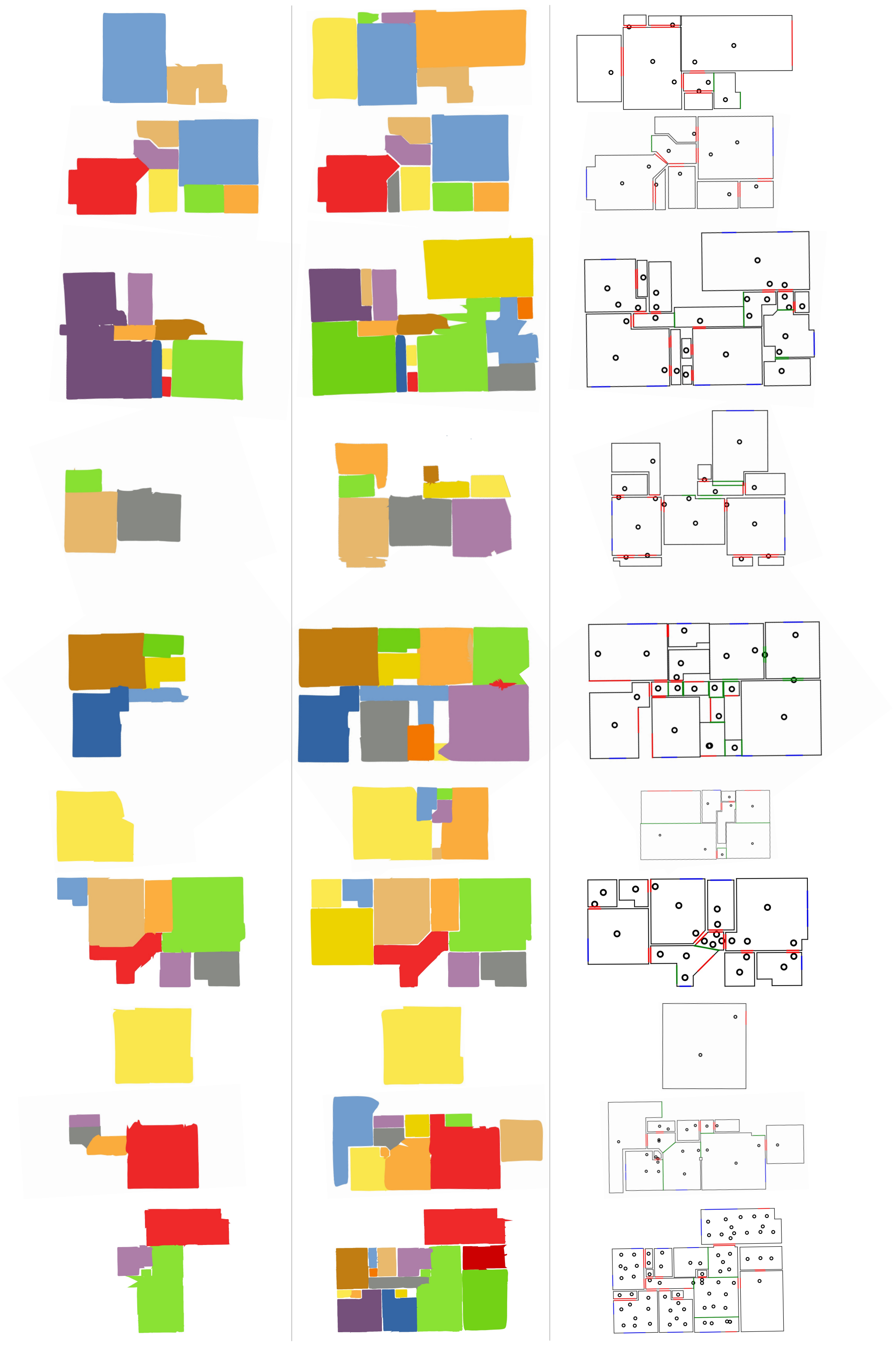


Figure 2.9: Additional comparison between SALVe's performance and an upper bound (perfect global pose estimation). Each row corresponds to a single floor of a different home. ***Left:*** predicted poses of the largest connected component and predicted room layout. ***Middle:*** oracle poses and predicted room layout. ***Right:*** ground truth floorplan with positions of captured panoramas. Colored lines represent W/D/O objects – doors, openings and windows.

Table 2.5: Relative pose classification accuracy on the ZInD test split with different inputs and architectures. Precision, recall, and mean accuracy are reported. Extreme class imbalance means that with more expressive model architectures, gains in mean accuracy are minor, but gains in precision are significant.

Model Architecture	Ceiling Texture Map	Floor Texture Map	Prec.	Rec.	mAcc.
ResNet-50	✓	✓	0.77	0.91	0.96
ResNet-152	✓	✓	**0.85**	**0.91**	**0.95**
ResNet-152	✓		0.70	0.88	0.93
ResNet-152		✓	0.84	**0.91**	**0.95**

Table 2.6: Additional W/D/O detection accuracy results.

	0.5 IoU			0.7 IoU			0.9 IoU		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Door	0.88	**0.92**	0.90	0.87	**0.91**	0.89	0.86	0.81	0.84
Window	**0.94**	0.91	**0.92**	**0.94**	0.89	**0.91**	**0.93**	**0.82**	**0.87**
Opening	0.79	0.65	0.72	0.78	0.59	0.67	0.72	0.43	0.54

2.10 Additional Analysis of Relative Pose Classification Accuracy

Here we provide a more comprehensive quantitative analysis of the influence of input modalities and CNN backbone architecture on SALVe's relative pose classification accuracy (referred to in subsection 2.5.5 of the main paper). We compare ceiling-only texture map input, vs. floor-only texture map input, vs. using both as input.

2.11 Details on Layout Stitching for Floorplan Reconstruction

This section provides additional details about the reconstruction algorithm mentioned in subsection 2.4.5 and Figure 2.4 of the main paper.

Floorplan reconstruction involves three steps: (1) panorama room grouping, (2) highest confidence room contour extraction, and (3) floorplan stitching. Please see Algorithm Algorithm 1 for implementation details. In Figure 2.10, we demonstrate the process of generating final room layout using estimated panorama poses grouped by step (1). Comparing plot (b) to plot (a), we can see that room contour confidence provide useful guidance in selecting the high confidence contour point among different views. In plot (c), each view-dependent room contour largely will agree with each other. In the end, we take the union of different view-dependent room contours to account for the occlusions from each panorama view.

2.12 Details on W/D/O Detection Evaluation

In subsection 2.5.4 of the main paper, we report W/D/O detection results of our HorizonNet [117] model at a 70% IoU threshold. For completeness, we provide here an evaluation of 1d IoU at 50%, 70%, and 90% true positive thresholds (see Table 2.6).

Algorithm 1 Floorplan Reconstruction for a Connected Component in Pose Graph

```
**Inputs:**
$\{I_i\}$: A list of the input panorama images in the connected component.
$\{\mathbf{T}_i\}$: Estimated panorama poses from pose graph, in top-down global 2D coordinates.
$\{(C_i, \sigma_i)\}$: Estimated room contour points and contour point confidence for panorama $I_i$, in top-down global 2D coordinates. (One point per panorama column.)

**Output:**
$S^{opt}_{floorplan}$: Optimized floor plan polygon shape.

**Solution:**
% Step 1: Group panoramas that come from the same room
Initialize panorama connectivity graph $\Gamma$ with one node per pano $I_i$ and no edges
for $(\mathbf{T}_i, \mathbf{T}_j) \in \{\mathbf{T}_i\} \times \{\mathbf{T}_i\}$ do
    $IoU \leftarrow ComputeContourIoU(\mathbf{T}_i, C_i, \mathbf{T}_j, C_j)$
    if $IoU > Threshold$ then
        $\Gamma.AddEdge(i, j)$
    end if
end for
% Each connected component in $\Gamma$ is a room
$\mathcal{G} = \{G^r, r = 1, ..., N_{rooms}\} \leftarrow \Gamma.GetConnectedComponents()$

% Step 2: Extract highest confidence contour for each room
for $G^r \in \mathcal{G}$: do
    Let optimized room shape $S^{opt}_r = \emptyset$
    for $I_i \in G^r$ do
        $\mathcal{P}^i = \{(P^i_j, \sigma^i_j)\}\ \forall\ I_j \in G^r$, where $(P^i_j,\sigma^i_j)$ are the projections of $(C_j, \sigma_j)$ onto pano $i$'s image
        In each image column of pano $i$, choose the most confident contour point from $\mathcal{P}^i$.
        $S^i \leftarrow$ the selected points, projected back into the 2D global coordinates using $\mathbf{T}_i$
    end for
    $S^{opt}_r = \bigcup_{I_i \in G^r} polygon(S^i)$
end for

% Step 3: Floorplan stitching
$S^{opt}_{floorplan} = \bigcup_{l=0}^{N_{rooms}} S^{opt}_r$
```

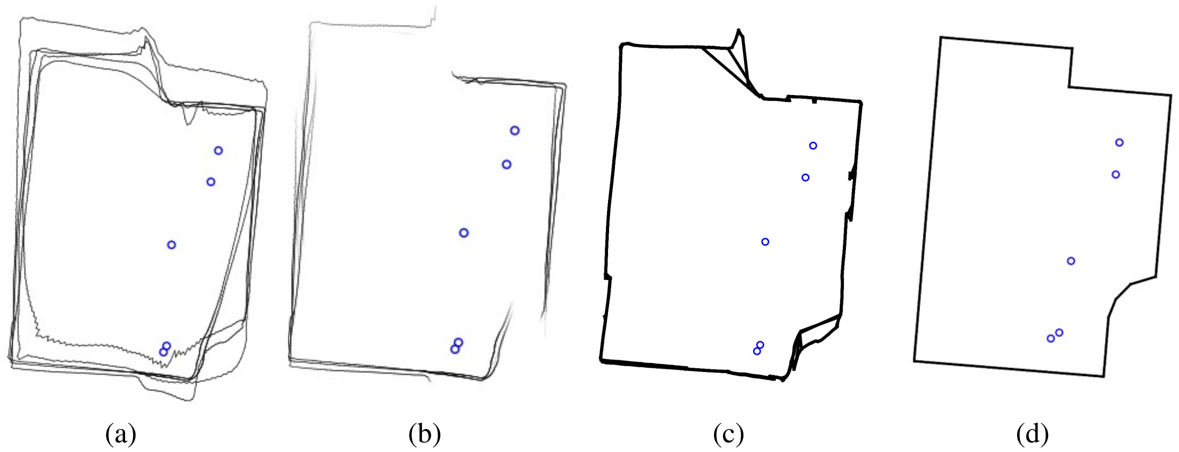


Figure 2.10: Visualization of room shape reconstruction using localized panoramas grouped by room. **(a)** Predicted room layout and predicted panorama locations (blue dots). **(b)** Predicted room layout with contour confidence (transparency) and predicted panorama locations (blue dots). **(c)** Overlay of room contours generated by voting on the highest confidence contour point at each panorama column from each panorama view. The final room layout is the union of these view-dependent contours of highest confidence. **(d)** Ground truth room shape and ground truth panorama positions.

2.13 Layout and W/D/O Failure Cases

Section subsection 2.5.4 of the main paper discusses the accuracy of W/D/O detection. Here we offer, in Figure 2.11, two examples of some the failure modes of the Layout and W/D/O model that provides the input to SALVe.

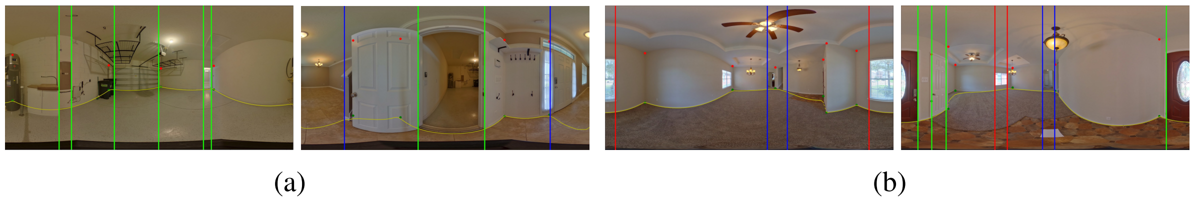


Figure 2.11: Mistakes made by the joint HorizonNet + W/D/O model. Vertical lines indicate start and end columns for each W/D/O object – window, door, and opening. The yellow contour indicates the predicted floor-wall boundary, and dots indicate corner predictions (floor-wall corners in green, and ceiling-wall corners in red). Left and right images are panoramas across which we seek to match W/D/O objects. ***Top:*** A circuit breaker panel is mistakenly identified as a door **(top left)**, but redundancy still allows matching of the true garage door. This allows a relative pose hypothesis to be generated between the foyer and garage panoramas, that have very little visual overlap. ***Bottom:*** A false negative window prediction and inaccurate opening prediction **(bottom left)** makes matching with the **(bottom right)** panorama impossible using W/D/O.

2.14 Coordinate System Conventions

Figure 2.12 shows the coordinate systems used in our work: panoramic spherical coordinate system, room Cartesian coordinate system, world-normalized Cartesian coordinate system (with camera height set to 1.0), and the world-metric coordinate system. These are also the coordinate conventions used by ZInD[3].

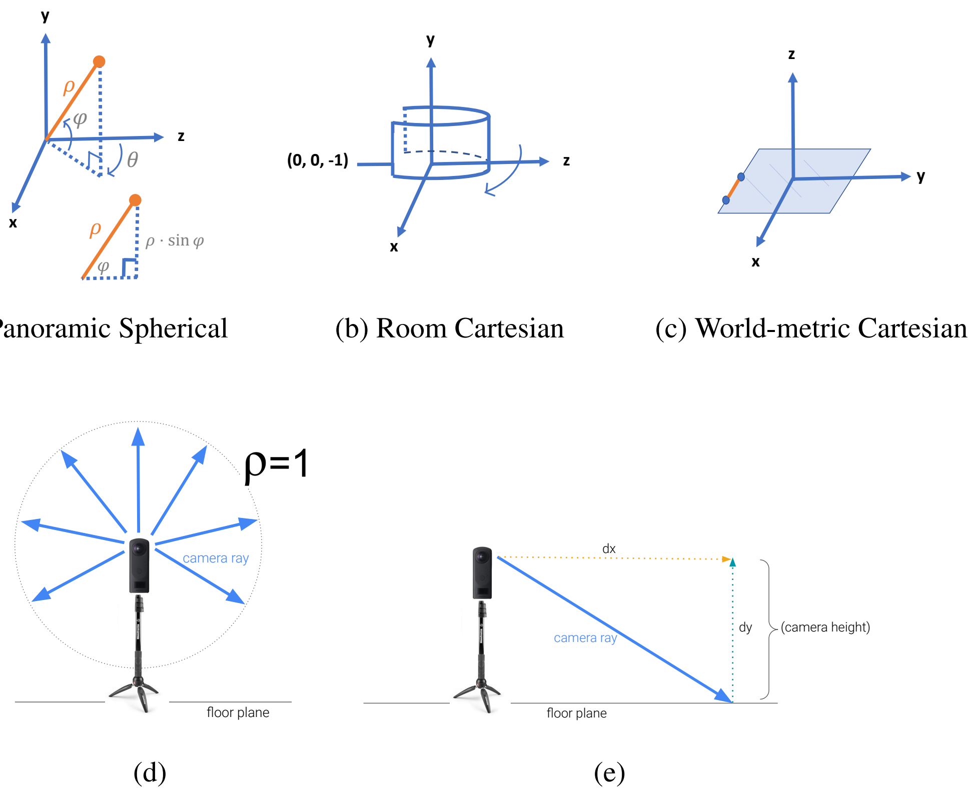


Figure 2.12: Coordinate system conventions.

Scaling to Metric Space. With known camera height, a predicted floor-wall boundary in pixel space with vertices $\{(u, v)_k\}_{k=1}^{K}$ can be mapped to 3D by first converting each vertex to spherical coordinates, and then to Cartesian coordinates, as follows:

$$\begin{aligned}
\theta &= \left(u \cdot \frac{2\pi}{(w-1)}\right) - \pi, \qquad \theta \in [-\pi, \pi] \\
\varphi &= \pi\left(1 - \frac{v}{(h-1)}\right) - \frac{\pi}{2}, \qquad \varphi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right],
\end{aligned} \tag{2.1}$$

where h and w is the height and width of input image in pixels.

We next obtain ray directions (x, y, z) in Cartesian space by assuming that all points

[3]ZInD is publicly available under the following license: https://bridgedataoutput.com/zillowterms.

(θ, ϕ, ρ) lie on the unit sphere (see Figure 2.12d), i.e., $\rho = 1$, and

$$\begin{aligned} x &= \cos(\varphi)\sin(\theta) \\ y &= \sin(\varphi) \\ z &= \cos(\varphi)\cos(\theta). \end{aligned} \tag{2.2}$$

Finally, we rescale the length of each ray such that it intersects the ground plane at $y = 0$, i.e., the magnitude of its y coordinate is equal to the camera height h_c (see Figure 2.12e). These rescaled ray directions are now coordinates in meters.

2.15 Texture Mapping Procedure

In this section, we discuss the interpolation procedure we use when creating bird's eye view (BEV) texture maps, as mentioned in subsection 2.5.2 of the main paper. When generating the orthographic imagery, the raw signal from pixel values at all backprojected depth map locations is sparse and insufficient. We rely upon interpolation to generate a dense canvas from the sparse canvas (see Figure 2.13, top). This interpolation also adds unwanted and undesirable interpolation artifacts (See Figure 2.13, middle subfigure). We design another step to identify the regions where the signal was too sparse to interpolate accurately. We convolve the canvas that is populated with sparse values with a box filter. We zero-out portions of an interpolated image where the signal is unreliable due to no measurements. If a $K \times K$ subgrid of an image has no sparse signals within it, and is initialized to a default value of zero, then convolution of the subgrid with a box filter of all 1's will be zero. In short, if the convolved output is zero in any ij cell, then we know that there was no true support for interpolation in this region, and we should mask out this interpolated value. We multiply the interpolated image with binary unreliability mask to zero out unreliable values. Convolution with a large kernel, e.g., 11×11 pixels in size on a 500p image, can be done on the GPU. We populate the canvas from bottom to top.

2.16 Vanishing Point Axis Alignment

Here we provide details about how we refine the hypothesized relative alignment described in Section 4.2 of the main paper.

To correct minor errors of W/D/O vertex localization, we compute vanishing points and convert them to a vanishing angle θ_{vp} with direction voting from line segments [160]. The vanishing angle θ_{vp} is defined as the horizontal angle between left edge of panorama and the first vanishing point from the left side of the panoramic image. We then refine the panorama horizontal rotation by aligning the pair of vanishing angles, while maintaining the distance between the matching W/D/O. The angular adjustment can be represented by: $\theta_{correction} = (\theta_{vp,1} - \theta_{vp,2}) - {}^{2}\theta_1$, where $\theta_{vp,1}$, $\theta_{vp,2}$ are the vanishing angles of panorama 1 and panorama 2, and ${}^{2}\theta_1$ corresponds to the relative rotation of panorama 1's pose in the room Cartesian coordinate system of panorama 2, i.e. of ${}^{2}\mathbf{T}_1$. We then rotate panorama 1's room vertices (in panorama 2's frame) about the W/D/O midpoint, and recompute $\hat{\mathbf{T}} = (\hat{x}, \hat{y}, \hat{\theta})$ by least-squares fitting between point sets to obtain $\hat{\mathbf{T}}_{corrected} = (\hat{x}', \hat{y}', \hat{\theta}')$ with fixed wall thickness.

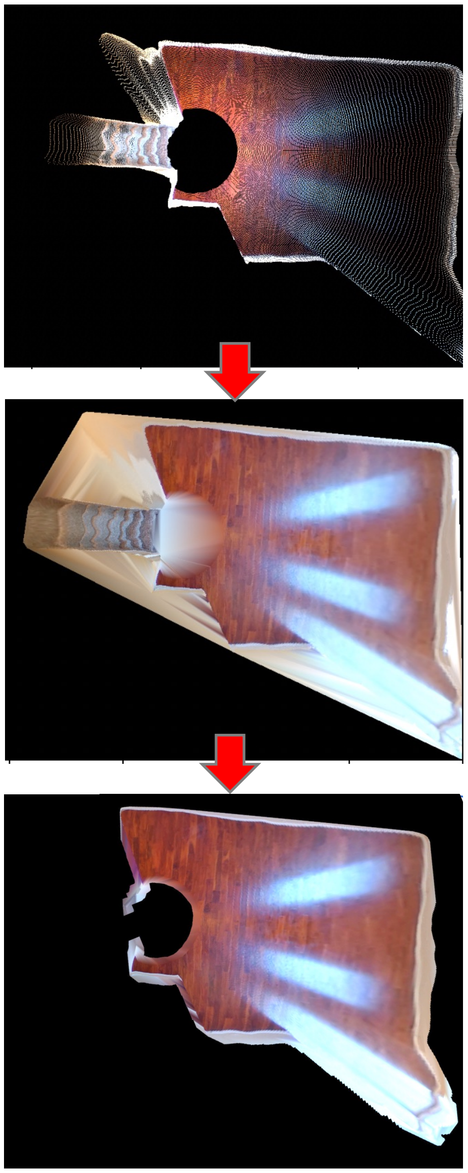


Figure 2.13: Visualization of the sparse to dense interpolation scheme. ***Top*****:** sparse texture map from mono-depth. ***Middle*****:** linearly interpolated texture map. ***Bottom*****:** result after removing interpolation artifacts.

2.17 Details on Global Pose Estimation

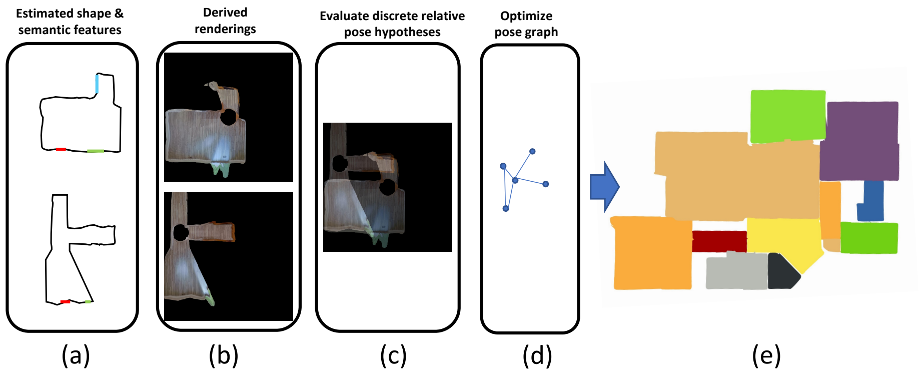

Estimated shape & semantic features
Derived renderings
Evaluate discrete relative pose hypotheses
Optimize pose graph
(a)
(b)
(c)
(d)
(e)

Figure 2.14: Starting with sparse panoramas (1-3 per room), in (a) we infer layout and semantic elements (Windows, Doors, Openings, or W/D/O). From these, in (b) we generate birds eye view (BEV) renderings of floors and ceilings (ceilings not shown here). Next, plausible pairwise relative poses are hypothesized based on matching W/D/O. Each is accepted or rejected (c), by feeding the hypothesis-aligned renderings into our learned SALVe verifier. This example shows two aligned renderings computed by hypothesizing that a window can be used to align both shapes Brighter areas indicate overlap regions. SALVe is trained to evaluate these aggregated overlap regions and output an accept/reject decision about whether the hypothesized relative pose is plausible. From the plausible relative poses, a pose graph is created and optimized (d). This allows room layouts to be positioned in a world coordinate system and fused into a final reconstructed raster floorplan (e).

Here we provide details on the pose estimation and optimization referred to in Section 4.4 of the main paper.

No optimization (unfiltered spanning tree). For N images, the global motion can be parameterized by $N-1$ motions. In the pose graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, when the graph G has a single connected component, the spanning tree is a set of edges such that every vertex in $\mathcal{V}$ is reachable from every other vertex in $\mathcal{V}$. For a graph with N vertices, the minimum spanning tree always has $N-1$ edges. However, global pose estimation with this method is inherently susceptible to error due to contamination by outliers. To describe the method to compute a spanning tree, we assume the images are randomly ordered. Starting from the first image as the root, we incrementally include images in sequence, adding each next image into the current tree at its shortest path from the root.

Pose Graph Optimization. Spanning tree solutions are susceptible to outlier edges in the tree. In order to exploit redundancy and utilize all the available information in the graph, *we use pose graph optimization.*

MAP inference for SLAM problems with Gaussian noise models is equivalent to solving a nonlinear least squares problem [172]. MAP inference comes down to maximizing the product of all factor graph potentials:

$$\mathbf{T}^{\text{MAP}} = \arg\max_{T} \prod_{(i,j)\in\mathcal{E}} \phi_{ij}(\mathbf{T}_i, \mathbf{T}_j), \tag{2.3}$$

where $\phi_{ij}(\mathbf{T}_i, \mathbf{T}_j)$ is a factor graph potential:

$$\phi_{ij}(\mathbf{T}_i, \mathbf{T}_j) \propto \exp\left\{ -\frac{1}{2}\|h_{ij}(\mathbf{T}_i, \mathbf{T}_j) - z_{ij}\|^2_{\Sigma_{ij}} \right\}, \tag{2.4}$$

with $h_{ij}(T_i, T_j) = T_i^{-1} \cdot T_j$ and z_{ij} is the estimated relative pose between images i and j from the alignment step described earlier.

The following objective function is then optimized using GTSAM:

$$\arg\min_{\mathbf{T}} \sum_{(i,j)\in\mathcal{E}} \rho\Big(\|h_{ij}(\mathbf{T}_i, \mathbf{T}_j) - z_{ij}\|^2_{\Sigma_{ij}} \Big), \tag{2.5}$$

making updates $\mathbf{T}_i \oplus \xi := \mathbf{T}_i \circ \exp(\hat{\xi})$, where $\xi \in \mathfrak{se}(2)$. Here, $\rho(\cdot)$ is a Huber noise model.

MAP inference over the pose graph with Gaussian noise models [172] is done by maximizing the product of all factor graph potentials. We initialize the solution from a greedy spanning tree and then optimize using GTSAM [116].

We follow the official GTSAM's PGO implementation example [4].

Once the pose graph is optimized, we use the estimated poses and room layout to create the final floorplan.

2.18 Ablation Experiments Using Oracle W/D/O Detection

In this section, we perform ablation experiments comparing global pose estimation and floorplan reconstruction results with estimated W/D/O locations and estimated layout, vs. a baseline that has access to ground truth W/D/O detections and ground truth layout.

How much worse is performance with predicted W/D/O and predicted layout vs. annotated D/W/O and annotated layout? In Table 2.7, we compute an upper bound for the completeness of our method, by using human-annotated W/D/O and human-annotated layout as input to the system. This measures the ability of the CNN to reason about photometric signal in a less noisy setting (there is still noise from HoHoNet). Note that annotations are not perfect.

The results indicate the already-strong localization precision of our system, with roughly similar camera pose estimation errors (in rotation and translation). We provide no vanishing-point axis-alignment post-processing to these generated relative poses, which leaves the ground-truth (GT) based system susceptible to higher rotation errors. However, the translation error of the model with access to GT W/D/O is still lower on average (22 cm vs. 25 cm).

With GT layout and GT W/D/O, the floorplan IoU is 91% higher – 0.86 median IoU vs. 0.45 IoU with our predicted poses and layout. The percentage of cameras localized is

[4]https://github.com/borglab/gtsam/blob/develop/python/gtsam/examples/Pose2SLAMExample.py

Table 2.7: Ablation experiments on global pose estimation, comparing performance with estimated W/D/O locations and estimated layout, vs. performance with ground truth W/D/O locations and ground truth layout (*oracle*).

Input	Localization %		Tour Avg. Rotation Error (deg.)		Tour Avg. Translation Error (meters)		Floorplan IoU	
	mean	median	mean	median	mean	median	mean	median
Predicted WDO + Predicted Layout	60.70	57.14	**3.73**	**0.17**	**0.80**	0.25	0.49	0.45
GT WDO + GT Layout	**88.58**	**93.44**	5.02	0.21	0.98	**0.22**	**0.78**	**0.86**

Table 2.8: Summary of comparison of our method vs. that of *Extremal SfM* by Shabani *et al.* [122].

	Extremal SfM [122]	Ours
Computational Complexity	Exponential Time $O(n!)$	Polynomial Time $O(n^2k^2)$
# Panoramas / Room	1	≥ 1
# Panoramas / Floor	3.4	23.2
# Floors in Test Set	46	291
Door Configuration	Opposite facing surface normals	Any configuration
Supported Room Type	Small size, Little self-occlusion	Any
Home Type	Apartment	Residential Home
Wall Assumption	Manhattan	None
Evaluated Error Types	Translation	Rotation and translation
Input Signal	BEV mask	BEV (image)
Verifier Type	GNN on tree-structured graph	CNN on pairwise renderings

also much higher (93.44% vs. 57.14%) than the system without access to GT. These results are extremely promising, underscoring the significant potential for further improving the floorplan reconstruction completeness of our system by improving the layout estimation and W/D/O detection network.

2.19 Comparison with Extremal SfM [122]

In this section, we provide additional comparisons with concurrent work by Shabani *et al.* [122] (See Table 2.8 and Table 2.9).

Differences in Assumptions.

- Shabani *et al.*'s one-panorama-per-room assumption limits the number of door hypotheses, as they assume door surface normals must point in opposite directions, whereas we consider twice as many hypotheses, i.e. when surface normals may also point in the same direction. The restricted door hypotheses would be analogous to querying a ZInD oracle for ground truth adjacency, which we do not do.

Differences in Method.

- They do not use openings. Accordingly, the rooms cannot be too large or complex enough to have significant amounts of self-occlusion.

Table 2.9: A more detailed comparison of our input, method, and evaluation vs. those of Shabani *et al.* [122].

COMPARISON TYPE	EXTREMAL SFM (SHABANI ET AL. [122]	OURS
INPUT	• Assumes no room shape overlap in dataset. • Requires exactly one panorama per room. • Requires each input panorama to see most of the room, including W/D. This would be problematic for complex rooms where one panorama sees only a fraction of the room. • Demonstrated on apartments.	• Handles any amount of overlap, but not zero overlap. • Requires one or more panoramas per room. • Has no requirement on panorama capture locations. • Evaluated on ZInD with complex room layouts, including open floor-plans.
METHOD	• Uses W/D alignment, but not openings. • Uses HorizonNet [117] for layout and separately predicted W/D objects at test time. • Uses room topologic information and BEV semantic masks (with no photometric information) directly as input to GNN. • Relies on a tree-type graph topology. They verify on all possible global graph configurations (room snapping combinations) with graph neural network Conv. MPN [157]. • Runs in exponential time with number of rooms.	• Uses W/D/O alignment to generate pairwise initial pose hypotheses. • Uses predicted room layout from HorizonNet [117] with joint W/D/O predictions and wall-floor boundary uncertainty. • Uses BEV photometric signal in panorama pairwise pose verification. • Uses global filtering and optimization similar to traditional Global SfM. Our method is also able to refine coarse panorama poses. • Runs in polynomial time with number of rooms.
EVALUATION	• Evaluated on a small dataset of 46 apartments. The dataset contains 3.4 panoramas per apartment with all Manhattan room layouts. • Generates top-5 possible floorplans. • Localization is considered a success if any of K possible solutions has estimated global poses with mean positional error below δ meters. Rotation error is not considered.	• Evaluated on the test split of ZInD with 291 floorplans of residential homes. ZInD contains 23.2 panoramas per floorplan with Manhattan and non-Manhattan room layouts. • Generates 1 final floorplan. • Evaluated by per-panorama average pose rotation and translation error, and localization completeness.

- Exponential time: They rely upon a tree-type graph topology. They verify on all possible global graph configurations (room snapping combinations) with graph neural network. Conv MPN [157] machinery (follows up on HouseGAN [173] and House-GAN++ [174] machinery). This requires exponential time.

Differences in Evaluation.

- Instead of computing a mean error per pano, they just count the successes. This does not take into account catastrophic failures (see their Figure 9). We have been evaluating as “you get one shot, and for every pano you try to localize, it will go into your mean error”. By comparison, they find the minimum subset out of all panos they localized that are good.

Unfortunately, at the time of our submission their code and dataset used were not publicly available, so a comparison of accuracies on a common dataset is not possible at this time.

2.20 Analysis of Computational Complexity

We compute over the ZInD train/val/test sets putative estimates of these constants n, k. On average for each ZInD tour, we find $n \approx 23.2$. When evaluating $\mathcal{O}(n^2 k^2)$, we find that each floor has on average 10795 (mean) and 8188.0 (median) putative alignments.

After pruning away impossible hypotheses via width ratios, n is unchanged but k is reduced, leaving $\mathcal{O}(n^2k^2) \approx 5804.5$ (mean) and 3441.0 (median). Although there are just as many panoramas to match with, we effectively have fewer instances of each W/D/O type we can feasibly match with (thus reducing k).

This leads to a highly imbalanced classification problem, with negative-to-positive ratios of 18:1 on average, when using predicted layout and predicted W/D/O locations.

Oracle Layout Generator Baseline. For alignments generated from GT W/D/O and GT layout, we halve the computational complexity by discarding those with penetrated freespace (an average negative-to-positive ratio of 7:1). For predicted layout, we cannot prune alignments by freespace penetration heuristics due to arbitrary predicted locations of openings in large rooms.

2.21 Details on Layout-Only Rasterization Baseline

In Table 2.2 of the main paper, we reported results of a model that has no access to photometric information as input, but only to rasterized BEV layout. In Figure 2.15, we show examples of such input.

2.22 Additional Discussion Points

2.22.1 Accuracy vs. Amount of Visual Overlap

Poor accuracy for examples with small support in the dataset (see Figure 2.16) shows the potential for hard negative mining in future work. Negative examples with high IoU (see Figure 2.16d, right) are few in the dataset, and present high error (see Figure 2.16a, right).

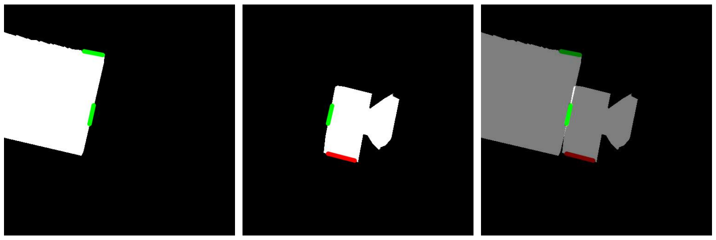
2.22.2 Additional Details on Evaluation Metrics

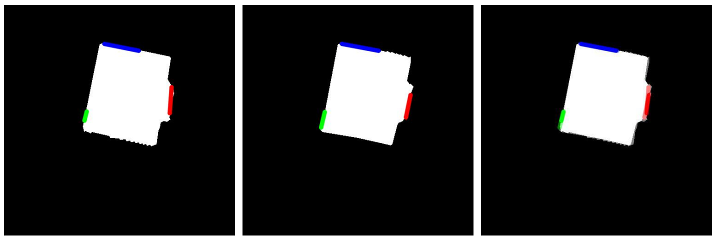
To compute both the ground truth mask and estimated binary mask for IoU computation, we rasterize the scene to a grid resolution of 10×10 centimeters per cell (we found an even finer resolution did not affect results significantly).

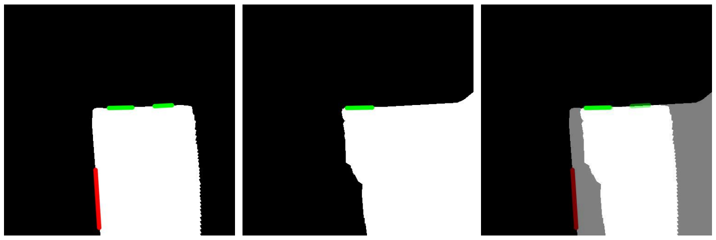
Error Rate in Supervision Generation. Noisy generation of ground truth is susceptible to false negatives. In other words, the generator falsely assumes that there are no ‘positive’ alignments for certain panorama pairs, due to errors just above the maximum tolerated rotation or translation thresholds. At inference time, these lead to false positive predictions, as the model identifies these pairs as positives, conflicting with the ground truth. We find that for spatially adjacent rooms, no putative ‘positive’ hypothesis is generated for less than 9.67% of panorama pairs, due to layout estimation or W/D/O prediction errors.

2.22.3 Model Learning

Previous work has proven that domain-knowledge of indoor space, such as room intersections, loop closure, and multi-view alignment, can be helpful in solving the ‘room merge’ problem [14]. On the other hand, visual overlap of floor and ceiling areas from different texture images provides helpful clues, such as light source reflections, paneling direction


(a) Positive example pair 1.


(b) Positive example pair 2.


(c) Negative example pair 1.

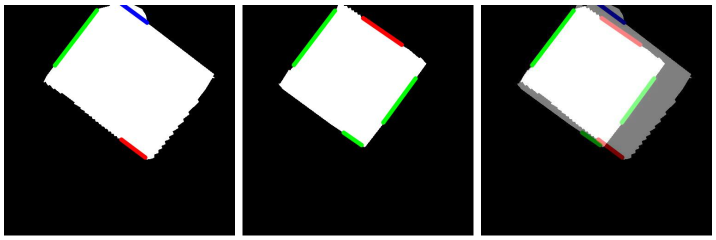
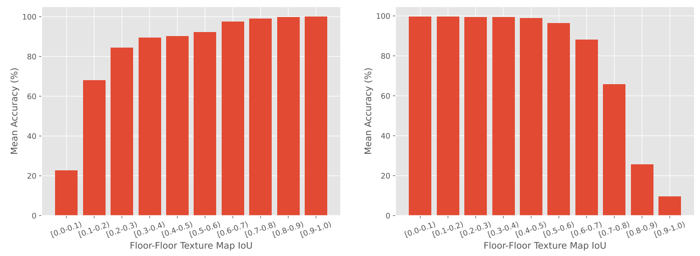
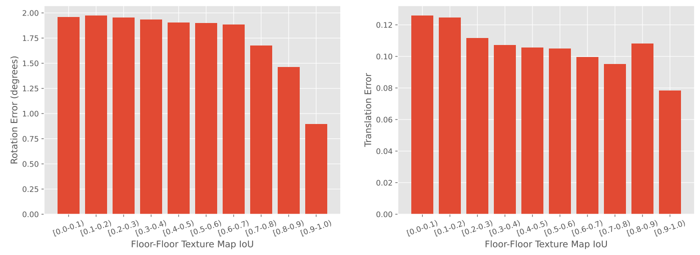

(d) Negative example pair 2.

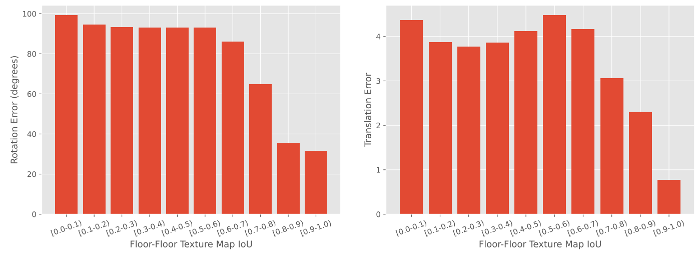
Figure 2.15: Examples of layout-only rasterized input. Each row represents an alignment pair. ***Left:*** rendering for panorama 1. ***Middle:*** rendering for panorama 2. ***Right:*** blended images (for visualization only).



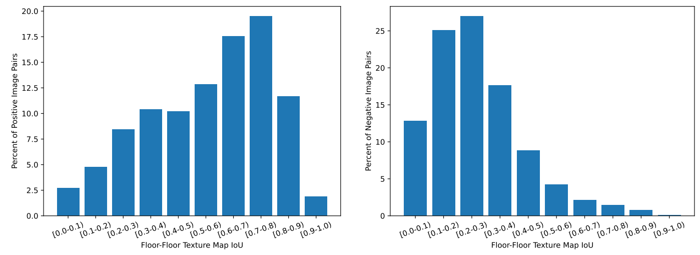
(a) Classification Accuracy vs. Visual Overlap (for GT Positives vs. Negatives)



(b) Rotation and Translation Error vs. Visual Overlap (for GT Positives)



(c) Rotation and Translation Error vs. Visual Overlap (for GT Negatives)



(d) Visual Overlap Distribution in ZinD (for GT Positives vs. Negatives)

Figure 2.16: **(Row 1)** Classification accuracy vs. overlap for the GT **positive** class only **(left)** and **negative** class only **(right)** for ResNet-152 model. **(Row 2)** Relative pose rotation error **(left)** and translation **(right)** vs. amount of visual overlap for GT *positive* examples. **(Row 3)** Relative pose rotation error **(left)** and translation **(right)** vs. amount of visual overlap for GT *negative* examples. **(Row 4)** Distribution of visual overlap (IoU) over rendered buildings for positive pairs **(left)** and negative pairs **(right)** from SE(2) alignments generated from predicted W/D/O's.

of wood flooring, and shared ceiling features, to verify panorama registration [162]. We extract undistorted floor and ceiling orthographic views from each panorama using inferred depth and register each view using an estimated W/D/O alignment hypothesis. While inferred depth signal alone suffers from inaccuracies at very close or far range and near reflective objects such as mirrors, the orthographic views still contain small distortion and therefore provides a strong signal for alignment verification. We train a model to implicitly verify the aligned texture signals (such as light source reflections, paneling direction of wood flooring, and shared ceiling features), as well as model other priors on room adjacency, such as the fact that bathrooms and bedrooms are often adjacent.

While ceiling features on a mosaiced ceiling image have been used for robot localization for at least two decades [162], success of registration using traditional explicit image-based matching is highly dependent on significant appearance similarity. This is typically not the case for our work, due to the large baselines and potentially very different times of capture.

Aligned orthographic views can help identify shared floor texture around room openings, identifying common objects (i.e. refrigerators), or known priors on room adjacency, such as the fact that bathrooms and bedrooms are often adjacent. axis-alignment of walls between two layouts. Because floorplan adjacency is governed by strong priors, such as primary bedrooms are attached to primary bathrooms, such signals can be learned.

Whereas previous works have employed domain-knowledge to manually define features for room-merge costs to employ in ranking [14], we set out to learn such features from data. Previous work has defined costs that aim to minimize room intersections, maximize loop closure, maximize multi-view alignment of semantic elements, and produce the most axis-aligned floorplans [14]. However, some of these costs are only applicable if an annotator can identify which panoramas were captured in different rooms, as layouts within the same room should instead *maximize* room intersection, while those captured in separate rooms should *minimize* room intersection. Each window from panorama $\mathbf{I}_a$ should reproject onto a window in panorama $\mathbf{I}_b$ only if the panoramas were captured in the same room, which is an unknown latent variable. IoU should be high between the two overlaid room-layouts; however, this is not true if they are in separate rooms (cross-room). If, on the other hand, we knew a "same-room" label, then layout-IoU would be useful during localization.

Available Signals for Learning Priors. Many possible complementary signals can be employed in the reconstruction problem. The $360°$ image suffers from significant distortion, while inferred depth suffers from inaccuracies at very close or far range and near reflective objects such as mirrors. Taken together, however, undistorted texture can be extracted in an orthographic manner. As the floor alone can have highly homogeneous texture or varied lighting, the ceiling can also provide helpful clues. Registration of inferred layout alone, with consideration of the image content, can lead to implausible arrangements. No single signal is sufficient.

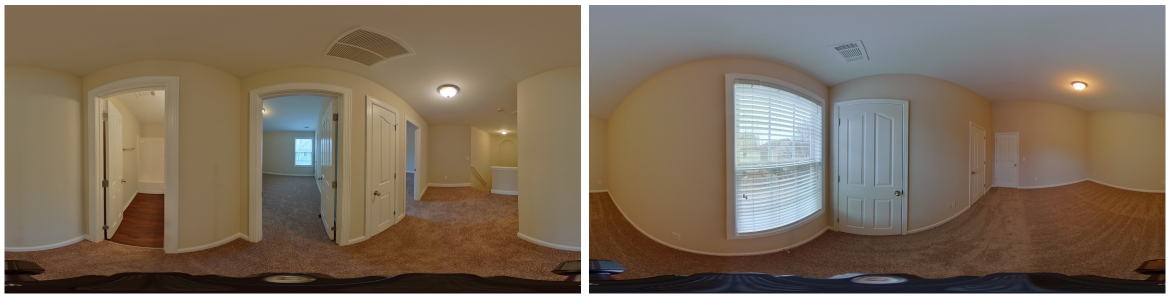
Human annotators use a variety of different cues to solve the merge task, most of them grounded in visual features within the image. For example, they often rely upon identifying shared floor texture around room openings, identifying common objects (i.e. refrigerators), or known priors on room adjacency, such as the fact that bathrooms and bedrooms are often adjacent. axis-alignment of walls between two layouts. Because floorplan adjacency is governed by strong priors, such as primary bedrooms are attached to primary bathrooms, such signals can be learned. Additional such relationships include hallway-to-bedroom adjacency.

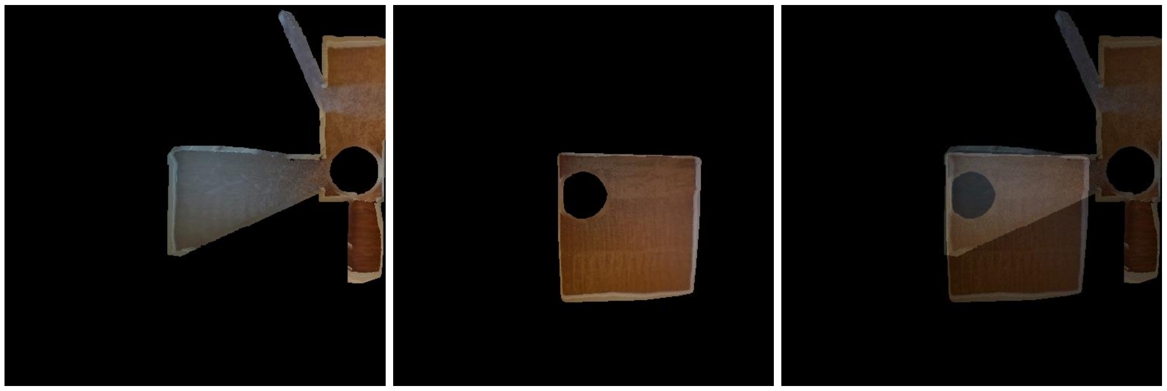
2.23 Additional Examples of Illumination Changes

In Figure 2.17, we provide an example of extreme illumination changes in ZInD [14], which prevent the use of classical image alignment algorithms for BEV image registration.

2.24 Ethical/Privacy/Transparency/Fairness/Social Impact Concerns

Floor plan reconstruction, in general, could potentially lead to privacy issues. However, schematic floorplans are able to abstract away details of the real interior space, thereby revealing the layout and functionality of a home while hiding personal information (PI) and personally identifiable information (PII) information from the images used to reconstruct it. In other words, we can build the floorplan from 360 panos and then immediately use the floorplan as a medium (to convey the space), while suffering from fewer privacy issues compared to releasing all the 360 images used to create it.


(a) Input panorama pair.


(b) Orthographic floor texture maps, with an extreme illumination change.

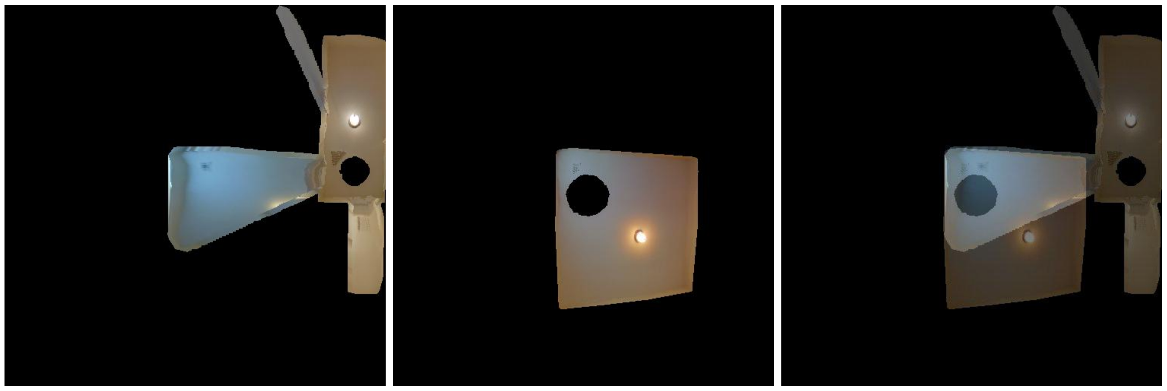

(c) Orthographic ceiling texture maps.

Figure 2.17: Example of an extreme illumination change, as the carpet color appears to shift from brown to grey **(middle)**, and ceiling from warm yellow to light blue **(bottom)**.

CHAPTER 3
CREATING OUTDOOR 3D GEOMETRIC MAPS VIA GLOBAL SFM

In this chapter, we transition from the creation of 2d geometric maps to the construction of 3d geometric maps, via Structure-from-Motion. We first analyze the performance of recent "deep front-ends" for SfM and SLAM by comparing them with classical front-ends on two datasets – HPSequences and YFCC. Afterwards, we put the best-performing methods to the test on several real-world "in-the-wild" datasets by using them in a new global SfM system that we design, called GTSFM.

3.0.1 Deep Front-Ends

In the first portion of this chapter, we introduce comprehensive benchmarks for the analysis of "Deep Front-Ends." We design a novel framework for benchmarking sparse local feature matching and we provide the most comprehensive study to date of such methods. We focus on methods suitable for the "front-end" of 3D computer vision applications and exhaustively test over 800 combinations of detection, matching, and verification methods. We standardize evaluation metrics from the literature and provide a uniform way of evaluating varied approaches to learning keypoint detectors, encoding feature descriptors, and outlier filtering with convolutional neural networks (CNNs). While homography-based datasets are preferred in the literature for detector and descriptor evaluation because of their ground truth pixel-to-pixel mappings, such datasets are not representative of the real 3d world. Accordingly, we experiment not only with a homography dataset (HP-Sequences) but also with a diverse wide-baseline stereo dataset (YFCC-100M) and discover unusual combinations of methods that are extremely effective. We develop new metrics based around the needs of practitioners and find, surprisingly, that many recent joint detector-descriptor methods are not competitive.

3.0.2 GTSFM

In the latter portion of this chapter, we introduce the Georgia Tech Structure from Motion (GTSFM) system, a distributed, global SfM system that incorporates state-of-the-art practices from dozens of recent works in SfM. We demonstrate the several new "Deep Front-Ends" increase the signal-to-noise ratio in Global SfM over classical methods, but most do not permit a signal-to-noise ratio sufficient for challenging large-scale datasets.

3.1 Introduction

Local feature matching systems remain a fundamental building block for camera calibration [175, 176], panorama stitching [177], image retrieval [178], visual localization [179], visual SLAM [180], and wide-baseline stereo applications such as Structure from Motion (SfM) [181, 182, 183]. These systems generally consist of three main stages: (1) identifying salient points ("features") inside each image to track, and (2) finding potential cor-

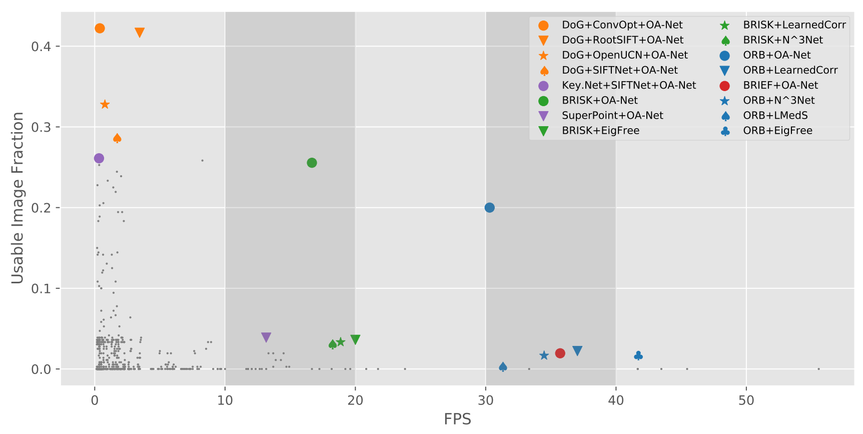


Figure 3.1: Competing objectives for a front-end – speed vs. accuracy – suggest a Pareto front. We illustrate several top-performing methods within each frame-rate range, and mark the rest with light gray dots. Accuracy is measured by the *Usable Image Fraction (UIF)* for the Essential matrix on YFCC-100M.

respondences between these salient points across different images (in order to track them), and (3) identifying and discarding incorrect correspondences.

We refer to these three steps as **D**, **M**, and **V**, i.e. (1) Feature **D**etection, (2) Feature Description and **M**atching, and (3) Putative Correspondence **V**erification (See Figure Figure 3.2). In short, the **D** stage provides detections, the **M** stage provides putative correspondences, and the **V** stage provides verified correspondences. When considering SLAM or SfM "front-ends", one could argue that descriptor matching should be viewed as a separate stage to descriptor extraction. We recognize this viewpoint, but in this work, we will associate the descriptor algorithm with descriptor matching as a single stage, rather than considering them in isolation. Specific matching criteria have almost always been associated with a descriptor; SIFT uses a one-way KNN ratio test [22], RootSIFT is SIFT with a Hellinger kernel [28]. Binary descriptors such as BRISK are to be matched using Hamming distance [54]. Convnet-based descriptors follow a similar protocol; D2-Net [43] uses a reciprocity-based matching test and GLAMPoints [87] uses the ratio test.

In this work, we perform a comprehensive evaluation of subsystems through five major experiments: feature detection on a wide-baseline dataset (w**D**) and homography dataset (h**D**), feature matching on a wide-baseline dataset (w**DM**) and homography datasets (h**DM**), and combined detection, matching, and verification on a wide-baseline dataset (w**DMV**). In addition, we take this opportunity to standardize several metrics, and also discover correlations between intermediate performance measures and final error metrics. Surprisingly, we find that a classical detector, DoG [22], provides the most effective final system performance. Many deep detectors optimized for intermediate metrics such as detector repeatability cannot compete with DoG on the usability of the system for epipolar geometry estimation. We find that although detector and descriptor implementations are abundant, no standard, accepted evaluation framework is readily found. In fact, most works evaluate their method with their own definitions of metrics and private implementations. For example, the proper definition of the repeatability rate and matching score is not agreed

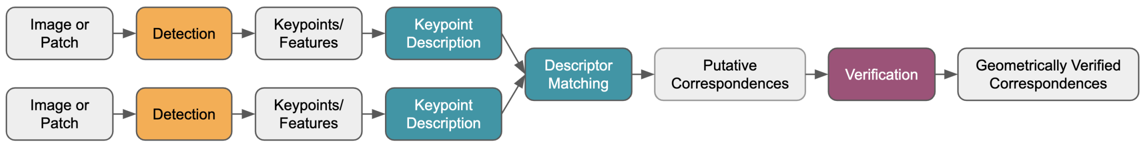


Figure 3.2: We graphically describe our sparse local feature matching framework that accepts either a pair of different images or patches. We denote the stages as DMV: Detection-Description and Matching- Verification.

upon in current works. We believe the lack of a standard evaluation protocol for modern feature-matching implementations is a significant hindrance to comparing and understanding their relative performance. Practically every state-of-the-art feature matching method uses a different set of metrics to gauge its quality (see Appendix).

In recent literature, a large number of new data-driven methods have been introduced for individual stages of a **DMV** system, but the optimal pairing of detector, descriptor, and verifier is poorly understood. Advances in convnet-based detectors and convnet-based descriptors have been measured without regard to the final verification stage, focusing mainly on the repeatability and matching precision of the detector-descriptor pair. This is a limited view, as epipolar geometry provides simple techniques to improve the purity of putative correspondences. Advances in convnet-based verifiers have been focused on verifying specific algorithms, such as SIFT, rather than on benefiting all-purpose system-performance. This is also a limited view, as the verifier may be designed and trained for a non-optimal detector-descriptor pair.

We tackle this problem with a combinatorial search for the best-performing feature matching system, exhaustively evaluating 808 possible combinations. Accordingly, we develop an notion of "final error" – system-based performance measures over a proxy task that we believe benefits a variety of downstream applications – numerous completely outlier-free verified matches. We do not directly measure SfM reconstruction error or visual localization performance.

In concurrent work, Jin *et al.* [104] also present a benchmark to measure the impact of pairing detectors, descriptors, and verifiers according to downstream performance. We have a much more comprehensive study, so we identify a different pipeline as most performant, but in line with their findings.

In our experience, SLAM/SfM practitioners often use the following rule of thumb: *for each image pair, a front-end should provide five times as many correct correspondences as the minimal inlier set, and there should be zero outliers*. The reasons are threefold: (1) least square residuals on outliers will cause significant problems for downstream bundle adjustment, thus purity is essential; (3) a minimal set alone of correspondences may provide insufficient support to RANSAC; (3) there is measurement noise in the image, so a minimal set is still too noisy for estimation in practice. For Essential matrix estimation, these criteria boil down to two measurable quantities that foretell the usability of the system for SfM: (1) after DM, the system should embed at least $5 \times 5 = 25$ correct correspondences in the putatives; (2) after **DMV**, the system should provide at least 25 inliers per image pair, with zero outliers.

We provide three lessons from our analysis, one about **D**, one about **DM**, one about

DMV:

- **D**: Highest performance on Repeatability doesn't directly correlate to highest **DMV** performance. Surprisingly, classical detectors take 8 of the top 9 ranks according to usability of the front-end system.
- **M**: We find no overlap in the top-15 methods by inlier ratio and the top-15 methods according to front-end system usability for SfM. In addition, classical and deep descriptors have surprising parity for system usability.
- **DMV**: If numerous image pairs usable for Essential matrix estimation is desired, *DoG+ConvOpt+OA-Net*, *DoG+RootSIFT+OA-Net* appear to be two of the best current options.

Our contributions are as follows:

- We perform a comprehensive evaluation of subsystems through five major experiments, placing performance on homography and wide-baseline stereo datasets on a uniform footing through our empirical study.
- We introduce two new metrics for downstream performance suited for practitioner's needs – the *Usable Image Fraction Upper Bound* and *Usable Image Fraction*.
- We perform the most exhaustive study to date of the optimal pairing of methods for a feature matching system. As part of an exhaustive combinatorial search, we discover a novel feature matching system – *DoG+ConvOpt+OA-Net* that can provide a guaranteed 25 correspondences with *zero outliers* on 42% of our sampled image pairs from 72 scenes in YFCC-100M.

3.2 Related Work

As we described above, SFM front-ends involve **DMV**: combined detection (**D**), descriptor matching (**M**), and verification (**V**). Due to the extremely extensive literature, we refer the reader to the Appendix for a survey where we describe the following methods in more detail.

Detection There has been a significant amount of progress in learning feature detectors from data but these methods are only evaluated only on homography datasets. Local feature detectors date back 40 years [184]. Originally hand-crafted [185, 186, 187, 59, 58, 22, 32, 89] feature detectors were eventually learned from data [188, 52, 73] and in the recent literature are all convnet-based [90, 95, 27, 42, 53, 98, 77, 189, 72, 87, 93, 33, 31, 93]. Over the years, many have worked to compare the relative performance of different detectors and place them on a uniform footing. In 2000, Schmid [190] presented an evaluation framework for interest point detectors to measure their *repeatability* and in 2004 [61] and 2005 [63] Mikolajczyk *et al.* presented benchmarks for the D stage. More recently, Lenc and Vedaldi [191] presented an empirical study of feature detectors in 2018.

Matching (including Descriptors) Similarly, deep learning has impacted descriptors and matching, but such methods are only evaluated on homography or patch-based datasets, which are not fully general to the 3d world. Local feature descriptors were also originally hand-crafted [184, 192, 21, 193, 22, 32, 28, 62, 86, 38, 54, 75, 194] then learned from data [51, 92, 91, 75, 78], and in the recent literature are also completely convnet-based [50, 47, 96, 96, 79, 40, 30, 64, 84, 95, 68, 42, 67, 49, 39, 43, 57, 72, 85, 44]. Several benchmarks for measuring DM have been proposed: two in 2005 by Mikolajczyk *et al.* [62] [63], one in 2012 by Heinly *et al.* [195] focused on the use of binary descriptors. Several benchmarks have measured the M stage without considering the D stage, such as two patch-based datasets: UBCPatches [91, 37] and HPatches [196]. Indeed, patch-level datasets led to seminal breakthroughs [91] for the field, as they enabled the earliest *learning-based* approaches, which now represent the dominant paradigm. While correctly matching or retrieving patches is certainly helpful, it discards the impact of the the detector, which plays a vital role in a **DMV** system. In addition, most modern methods no longer operate on patch input, but rather on image-input. For example, UCN [40] [197], SuperPoint [42], D2-Net [43], Key.Net [31] and R2D2 [72] accept full-images to their fully-convolutional backbones. Accordingly, we believe the current focus of benchmarks involving the M stage should measure image-level metrics image-level datasets, such as HP-Sequences or YFCC-100M.

Deep Matching [76]

DMV Systems that use combined local feature detection, descriptor matching and geometric verification (**DMV**) for wide-baseline stereo date back to [198, 88] and were used in the well-known PhotoTourism project [199]. Geometric verification using robust estimation techniques [169, 200, 201] has been replaced with deep putative correspondence verifiers [35, 94, 70, 41, 102, 202, 69, 97, 36]. The jury is still out on deep architectures for the verification stage. Schonberger *et al.* [203] performed an end-to-end **DMV** evaluation specifically for SfM and measured the reconstruction quality by reprojection error. Their finding was surprising – advances in raw matching performance from learned descriptors did not necessarily lead to superior reconstruction results.

3.3 Detection (D) Benchmark

In this section, we discuss the "**D**" stage of our *Deep Front-Ends* benchmark, corresponding to feature detection. We define the task, evaluation metrics, and present an analysis of experimental results.
Problem Description Given two images of a scene, the **D**-stage task is to identify numerous small 2d features in both images that are equivariant to viewpoint [63, 53, 98, 42, 204], invariant to photometric transformations [21, 90], and are highly spatially distributed over the image [205, 206, 26, 87].
Datasets Because wide-baseline stereo represents many applications in the 3d world and homography datasets are the *de facto* choice in the literature, we select datasets that cover both regimes. For wide-baseline stereo, we select the test split of YFCC-100M [207, 208];

since it provides relative poses between each image pair as would be obtained by Structure from Motion or visual SLAM, it can be employed for the D, DM, and DMV benchmarks. While several other suitable datasets exist, such as Wilson and Snavely's 1dSfM (e.g. Roman Forum) [209], we note that YFCC-100M has seen adoption in several works [94, 197, 69, 97]. YFCC-100M has significant diversity of scenes throughout the world (72 scenes) and large size: the visibility graph of several scenes exceeds 10,000 edges with at least 100 covisible keypoints per image pair. However, due to its large size, we are forced to reduce the size of the evaluation split; we randomly sampling 5 image pairs from the edges of the visibility graph of each scene (with at least 100 covisible keypoints). For each of YFCC-100M's 72 scenes, we take these five image pairs from the scene's test split, providing 360 image pairs in total.
Methods and Evaluation We evaluate nine method detector-only methods and also scrape the detections from 10 joint detector-descriptors methods, leading to an evaluation of 19 methods (see Column 1 of Table 3.3). In order to measure the D-stage, we use keypoint distance-based repeatability between each image pair, i.e. the ratio of corresponding keypoints and covisible keypoints [190, 42, 87], with a 3 pixel true positive threshold. Because (YFCC-100M) does not provide per-keypoint covisibility information, we use two slightly different repeatability metrics for our wide-baseline stereo dataset (YFCC-100M) and homography dataset (HP-Sequences); while for HP-Sequences we set the ratio's denominator to the number of covisible-keypoints, for YFCC-100M we use the number of all keypoints. We refer the reader to the Appendix for additional evaluation details. As many of noted [53, 191], repeatability tends to increase with additional feature detections; accordingly, we sort keypoints by confidence and compute repeatability with cardinality thresholds of 150, 300, 600, 1200, 2400 keypoints, in accordance with [77, 72]. Due to the very large size of YFCC-100M, we randomly sample five image pairs from each of the 72 scenes for evaluation. In order to accelerate computation, we resize YFCC-100M images from an original average resolution of 444×625 px to 267×375 px, preserving aspect ratio. We resize HP-Sequences images from an original average resolution of 768×1078 px to 307×431 px, also preserving aspect ratio.

Analysis

We present a graphical representation of results on both datasets in Figure 3.3; due to space constraints we provide the two full tables in the Appendix. We also provide comparisons of methods according to feedforward runtime in the Appendix. We note that the top-achieving method differs among homography datasets and wide-baseline stereo datasets: SuperPoint and FAST for illumination-invariance on homography, Harris and FAST for viewpoint-invariance on homography; D2-Net performs poorly on homography and well on wide-base line; SuperPoint follows the opposite trend. Notably, the FAST detector appears to perform well in both regimes. R2D2 and LF-Net have a rapid increase in repeatability as the number of keypoints increase.

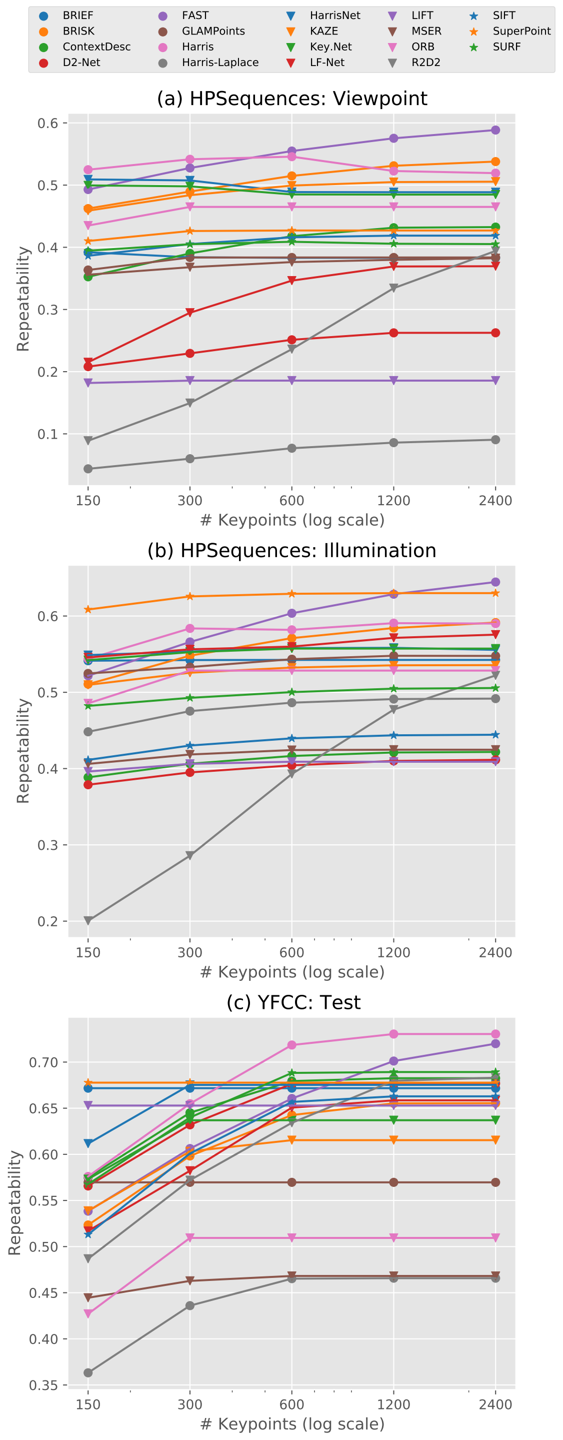


Figure 3.3: **D**-stage results depicting repeatability as a function of keypoint budget. Top: hD, on the two splits of a homography dataset (HP-Sequences), one with illumination-variant image sequences, and the other with viewpoint-variant image sequences. Below: w**D**, on the test split of a wide-baseline stereo dataset (YFCC-100M).

Detectors	Descriptors	Matchers	Verifiers
Harris [186], MSER [58]	SIFT Desc. [21, 22]	One Way w/o ratio test w/o bijection	RANSAC [169], LMedS [200]
Harris-Laplace [59, 61], FAST [73]	PCA-SIFT Desc. [51], SURF Desc. [32]	One Way w/o ratio test w/ bijection	Deep Fundamental Matrix [70]
DoG [22],	RootSIFT Desc. [28], ConvOpt [78]	One way w/ ratio test w/o bijection	LearnedCorr [94], MLESAC [201]
DDet/CovDet [53], Key.Net [31]	DeepDesc [79], BRIEF [38]	One way w/ ratio test w/ bijection	Eig-Free [41], N^3 Net [69]
GLAMPoints [87],	TFeat [30], UCN [40, 197]	Two way w/o ratio test	Order-Aware Net [97]
HarrisNet (ours)	Spread-Out HardNet [99]	Two way w/ ratio test	NG-RANSAC [36]
	SIFTNet (ours), BRISK Desc. [54]	SuperGlue [76]	*combinations of Deep + classical*
	ORB Desc. [75]		
Joint Detector-Descriptors			
BRISK [54], ContextDesc [57] D2-Net [43], KAZE [25], LF-Net [68], ORB [75], R2D2 [72], RootSIFT [28] SIFT [22], IMIPS [39], PCA-SIFT Desc. [51], LIFT-TF [95], SuperPoint [42], SURF [32]			

Table 3.1: "Front End" Search Space. Open-source methods of each group are listed in chronological order. All classical front-end implementations (not deep nets) are from OpenCV or VLFeat, all dating to 2014 or earlier. Methods that lacked publicly available models and source code at the time of experiments included: Quad-Networks [77], Self-Improving Visual Odometry [210], Epipolar Adaptation [93], KeyPointNet + IO-Net [83], Reinforced SuperPoint [33] (five detectors), and the SuperGlue matcher (Source code has since been made publicly available for SuperGlue).
[76].

3.4 Detection and Matching (DM) Benchmark

In this section, we present the combinatorial **DM** stage of our *Deep Front-Ends* benchmark, corresponding to descriptor description and matching (**M**) using detections from the (**D**) stage. We define the problem and discuss the choice of suitable evaluation metrics and evaluation datasets. For the **DM** benchmark, we use the same splits of YFCC-100M and HP-Sequences as employed in the **D** benchmark.

Problem Description and Evaluation The **DM** task is defined as, given an image pair, detect features, form a fixed dimensional descriptor vector to describe each feature, and then match their descriptors, providing putative correspondences. We evaluate four metrics (described below: Inlier Ratio, PMR, MMA-AUC, UIF-LB) on both homography (HP-Sequences) and wide-baseline (YFCC-100M) datasets, but can compute another two metrics (MCovFrac and M-Score) only on the homography dataset due to the lack of per-point co-visibility information on YFCC-100M [207], which these two additional metrics require.

Inlier Ratio Measures the precision of matching and the descriptor's discriminative ability as $(^{\#\text{Correct Matches}}/_{\#\text{Putative Matches}})$. We define correct matches (true positives) as putative correspondences with 3 px deviation from a pixel-pixel mapping (homography) [62] or 3 px from a pixel-to-line mapping (symmetric distance from the epipolar line) [40]. The Inlier Ratio has significant performance consequences for robust estimation modules that use feature matches, such as RANSAC, where execution times increase exponentially as the inlier ratio decreases [195].

Putative Match Ratio (PMR) PMR is the quantity of features useful for generating matches: $(^{\#\text{Putative Matches}}/_{\#\text{Covisible Features Proposed}})$ [195].

Matching Score (M-Score) [63, 195, 43, 31, 95] The matching-score is the product of PMR and the Inlier Ratio [195]. In other words, it is the ratio of ground-truth correspondences that can be recovered by the two detection-description stages, over the number of features proposed by the detector in the *shared viewpoint region*, per LIFT [95].

Mean Matching Accuracy Area-Under-the-Curve (MMA-AUC) Since the 2.5 or 3 pixel true positive (TP) threshold in Inlier Ratio and M-Score is arbitrary, MMA instead evaluates the Inlier Ratio (matching-precision) at different thresholds from 1 to 10 px [43, 72], averaged over many image pairs. We define MMA-AUC as the area under this curve, providing a single scalar with information about how inlier ratios fluctuate with the choice of TP threshold.

Matching Coverage Fraction (MCovFrac) Measures the coverage of the covisible portion of an image by correctly matched key points. A coverage mask is generated from true positive keypoints, each one adding a disk of fixed radius (25px) [26, 87]. Because YFCC does not provide co-visibility information, we report MCovFrac only on HP-Sequences.

Usable Image Fraction Upper Bound (UIF-UB) Given n_c correct correspondences embedded in n_p putative correspondences for a single image pair, and given the size of a minimal inlier set n_{mi}, the feasible usability of this i'th image is $\zeta_i = \mathbb{1}\{n_c \geq 5 \cdot n_{mi}\}$. In other words, an accepted practice is to require 5x correct correspondences as the number of minimal correspondences. Thus, an upper bound on the verifier's performance (given a perfect verifier) is formed by averaging ζ_i over $n_{\mathcal{D}}$ images in the dataset $\mathcal{D}$: $^1/_{n_{\mathcal{D}}} \sum_i \zeta_i$. For a homography, this is the number of image pairs that have at least $4 \times 5 = 20$ correct correspondences retrieved by the **DM** stages; for an Essential matrix [176], $5 \times 5 = 25$, and for a Fundamental matrix, $8 \times 5 = 40$.

Combinatorial Search Problem for Methods Our combinatorial search space for **DM** consists of 9 detectors, 10 descriptors, 11 detector-descriptors (see the first two columns of Table 3.3). This provides $9 \times 10 = 90$ pairwise combinations by combining a detector-only and descriptor-only method, and another 11 from joint detector-descriptors. In practice, a novel pair is formed by using a detector-only pair to detect m keypoints, which are then passed to the descriptor-only method for description. For many models, the detector and descriptor are not advertised as separable, e.g. SIFT [22] and SURF [32], but can be separated in practice, e.g. by using Harris keypoints with the SIFT descriptor.

Of course, there is varied input for different methods at intermediate stages. For example, we note that several descriptor-only methods accept as input entire images, e.g. UCN [40] and SIFTNet (see Appendix). Therefore, in such scenarios, when a detector passes m keypoints to a descriptor-only method, instead of feeding in local patches, an entire image is passed to the descriptor-method, and then embeddings are selected at all m keypoints. When the descriptor requires a scale or orientation which the detector cannot provide, we use a default value (see Appendix).

We aim to preserve the fidelity of the original authors' implementation (including prescribed matching strategy) as much as possible; additional details about the matching schemes employed are provided in the Appendix.

Results

We present the **M**-stage results on a homography dataset (HP-Sequences) in Figure 3.4 and **M**-stage results on a wide-baseline stereo dataset (YFCC-100M) in Figure 3.5. We provide a comparison of relative feedforward runtimes for different methods in the Appendix.

Across both datasets, we find a number of common trends. First we discover that the inlier ratio (matching-precision) is directly correlated to MMA AUC, intuitive because the inlier ratio is a single point on the MMA curve. Second, surprisingly, we find no correlation between UIF-UB and the inlier ratio. However, when examining usability, we discover that even if the inlier ratio is low, a high PMR can drive up the UIF-UB metric (see Figure 3.5). When PMR is high, a larger amount of feature detections are converted to matches, providing a balancing force even if they are incorrect.

In the homography setting of the **M**-stage, we make a number of discoveries. First, viewpoint-variant images are more challenging than illumination-variant images for *all* detector-descriptor pairs. Since Inlier Ratio is more interpretative, we report Inlier Ratio in Figure 3.4. We find that the usability of the top methods are nearly identical: A classical detector like *FAST/Harris* combined with *OpenUCN/SpreadOut HardNet* achieve greater than 97% usability. Joint methods like *R2D2* and *ConvOpt* make an appearance in top-10 ranks by usability in the homograpy dataset, unlike their absence in wide base line methods. No other metric demonstrates a high correlation with UIF-UB: low PMR is bad but high PMR does not guarantee a good usability. Methods like *SuperPoint* and *MSER+SURF* have high inlier ratio but lower usability. Matching Coverage Fraction surprisingly turns out to be the most correlated with usability.

In the wide-baseline setting of the **M**-stage, we make a number of discoveries. Notably, SIFT provides the highest inlier ratio of any method, outperforming the 2nd-ranked method *HarrisNet+TFeat* by 11% (73% vs. 62%). When comparing usability of the method for Essential Matrix or Fundamental Matrix estimation, we discover that the ability to estimate one matrix is nearly identical with the other. Accordingly, we show only E-Matrix usability in Figure 3.5. *FAST+OpenUCN* provides the most usability, outperforming ContextDesc by 2% (100% vs. 98%). No single detector "wins" for usability (UIF-UB), and classical and deep methods perform about as well; FAST, Harris, ContextDesc, and SURF perform similarly on UIF-UB.

3.5 Detection, Matching, and Verification (DMV) Benchmark

In this section, we discuss the **DMV** system-wide benchmark, corresponding to all three combined front-end stages. We define the task, evaluation metrics, and present experimental results. This final benchmark measures the quality of the entire feature-matching system. An end-to-end evaluation of front-ends is important because feature detectors *in and of themselves* have little utility. Feature descriptors do have have utility, for example as bags of visual word (BoW) elements for applications such as image database retrieval and cross-time localization [179]. Most 3D computer vision applications cannot afford high descriptor outlier rates and thus require all three of the **DMV** phases.

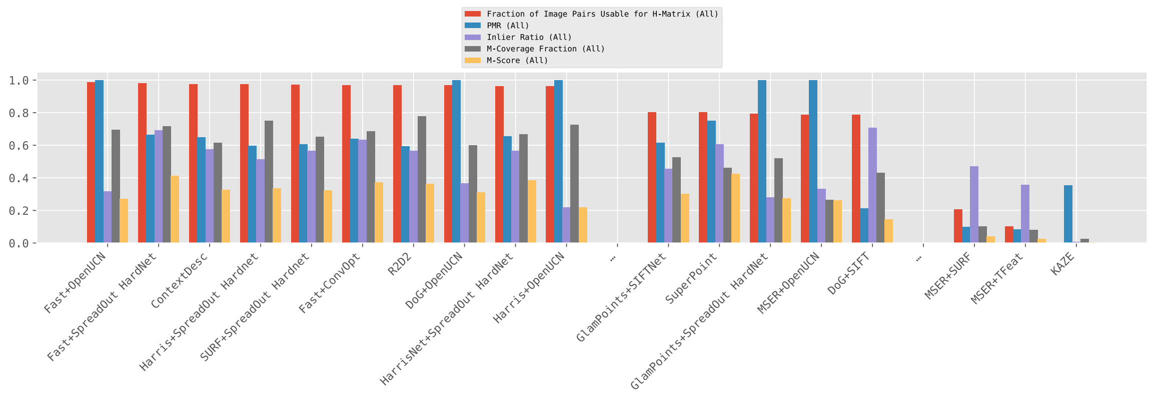
Problem Description The **DMV** task is defined as, given an image pair, detect features, match their descriptors, verify the putative correspondences by providing binary labels, and present the final, verified correspondences. In other words, the verification stage shatters the putative correspondences into two sets – inliers and outliers – by performing binary classification. If the user cannot provide such binary labels, but can only estimate an E or F matrix, this is also acceptable, as we will immediately classify each putative correspondence according to a fixed epipolar constraint violation threshold ρ. Indeed, after **DMV**, one should be able to assume the verified correspondences are practically noise-free and directly employ them in the Normalized 8-Point [175] or 5-Point [176] algorithms, without any further filtering. After **DMV**, the “front-end” is considered completed, and the “back-end” [211] begins its work.

Method Combinations We present a large combinatorial search problem for **DMV**: identifying the three most effective detectors, descriptors, and verification algorithms to use together for a specific end-to-end task. Our combinatorial search space for **DMV** consists of 808 possible (detector, descriptor, verifier) triplet combinations: $(9 \times 10) + 11 = 101$ from the **DM** stage, and and 8 verifiers (see Table Table 3.3, giving $(90 + 11) \times 8 = 808$, which represents a non-trivial amount of evaluation computation. Several verification (**V**) methods require specific side information [70, 102] from specific detectors and cannot be paired with arbitrary detection-descriptor pairs. Accordingly, we exclude them from our benchmark. We discuss evaluation and implementation details in the Appendix.

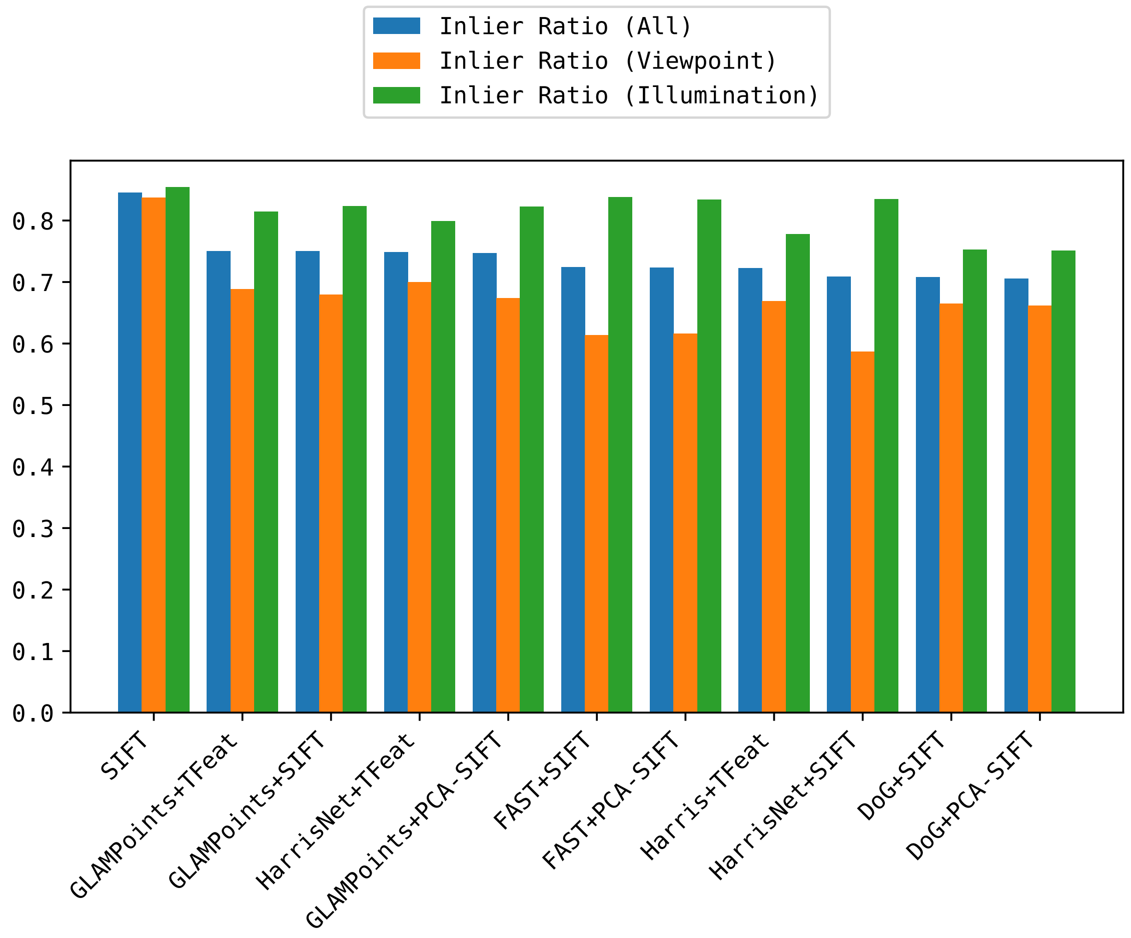
Datasets In the **DMV** benchmark, we exclusively use the YFCC-100M dataset. Homography based datasets are not suitable for such a verification benchmark for a number of reasons, mainly because the scenes are not representative of the real 3d world and while at least two deep homography estimation models [212, 213] have been proposed, the pretrained models are not publicly available. In addition, several deep verifiers [94, 69] require knowledge of camera intrinsics for normalization of image coordinates, which HP-Sequences cannot provide.

Evaluation We define a new metric, the *Usable Image Fraction* (UIF). Significantly more stringent than its upper-bound, UIF-UB, the UIF states that for each image pair, a method must generate at least 5x the number of minimal correspondences, with 100% purity. The UIF is the fraction of image pairs in the dataset that are “usable” for E- or F-matrix estimation. Others have defined metrics to measure F-matrix estimation using virtual points [70]; however, we find that these metrics are less interpretable for practitioners than the UIF.

Analysis

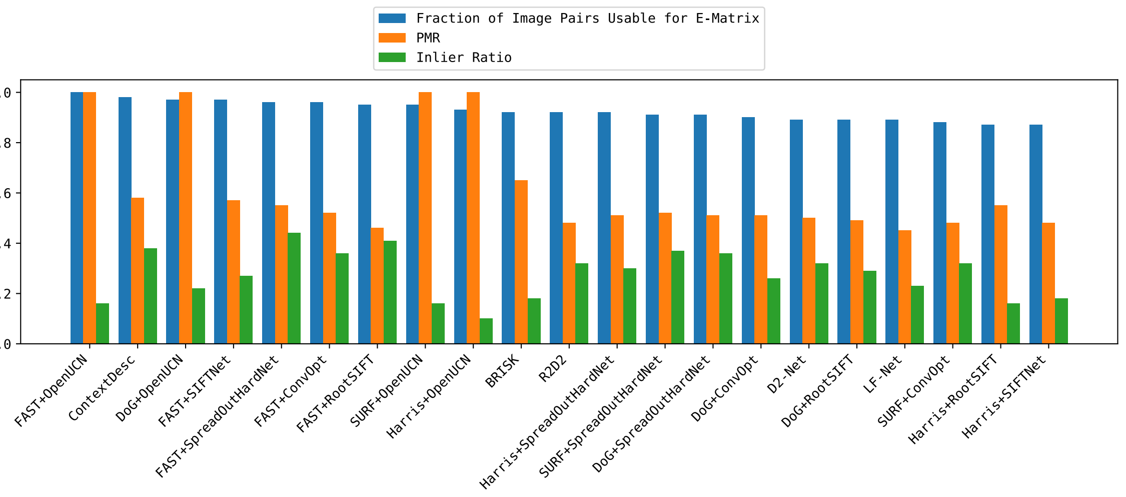


(a) Top 10, sorted by UIF-UB (for $\mathcal{H}$), and other representative entries



(b) Top 10, Sorted by Inlier Ratio

Figure 3.4: Homography **DM** (h**DM**) results on HP-Sequences. **(top)** From left-to-right, we show the top-10 performers on usability for homography estimation (UIF-UB), and other representative entries. **(bottom)** Top-10 performers according to Inlier Ratio. We note that viewpoint-variant sequences are more challenging than illumination-variant sequences. We provide the full table in the Appendix.



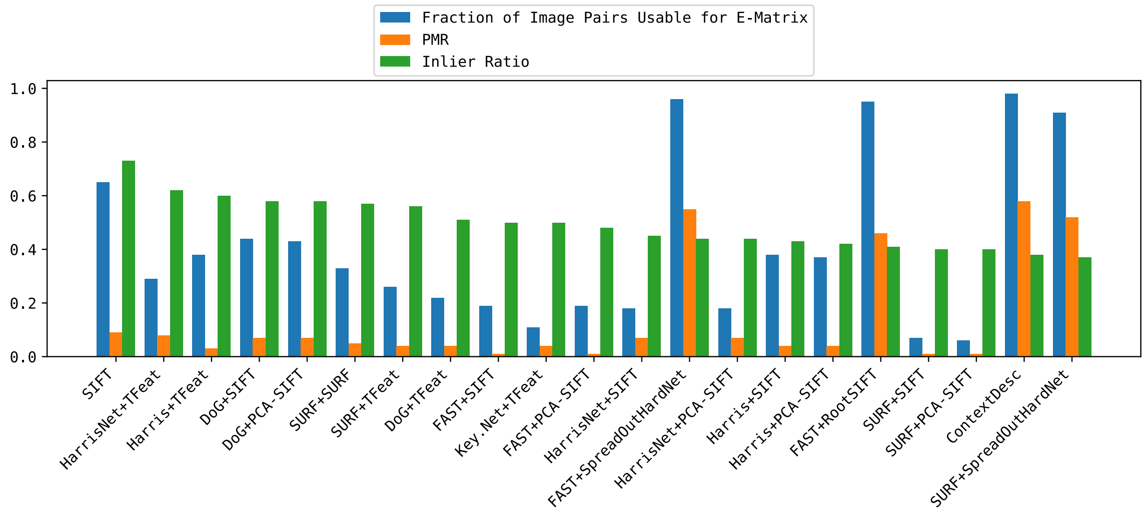


Figure 3.5: Wide-baseline stereo descriptor matching (wDM) results (YFCC-100M). From left to right, the 20 top-ranking methods (of 102 methods) are presented, **(top)** sorted according to their E-Matrix estimation usability, or **(bottom)** according to inlier ratio. It appears that high PMR may help boost E-Matrix estimation usability. We provide the full table in the Appendix.

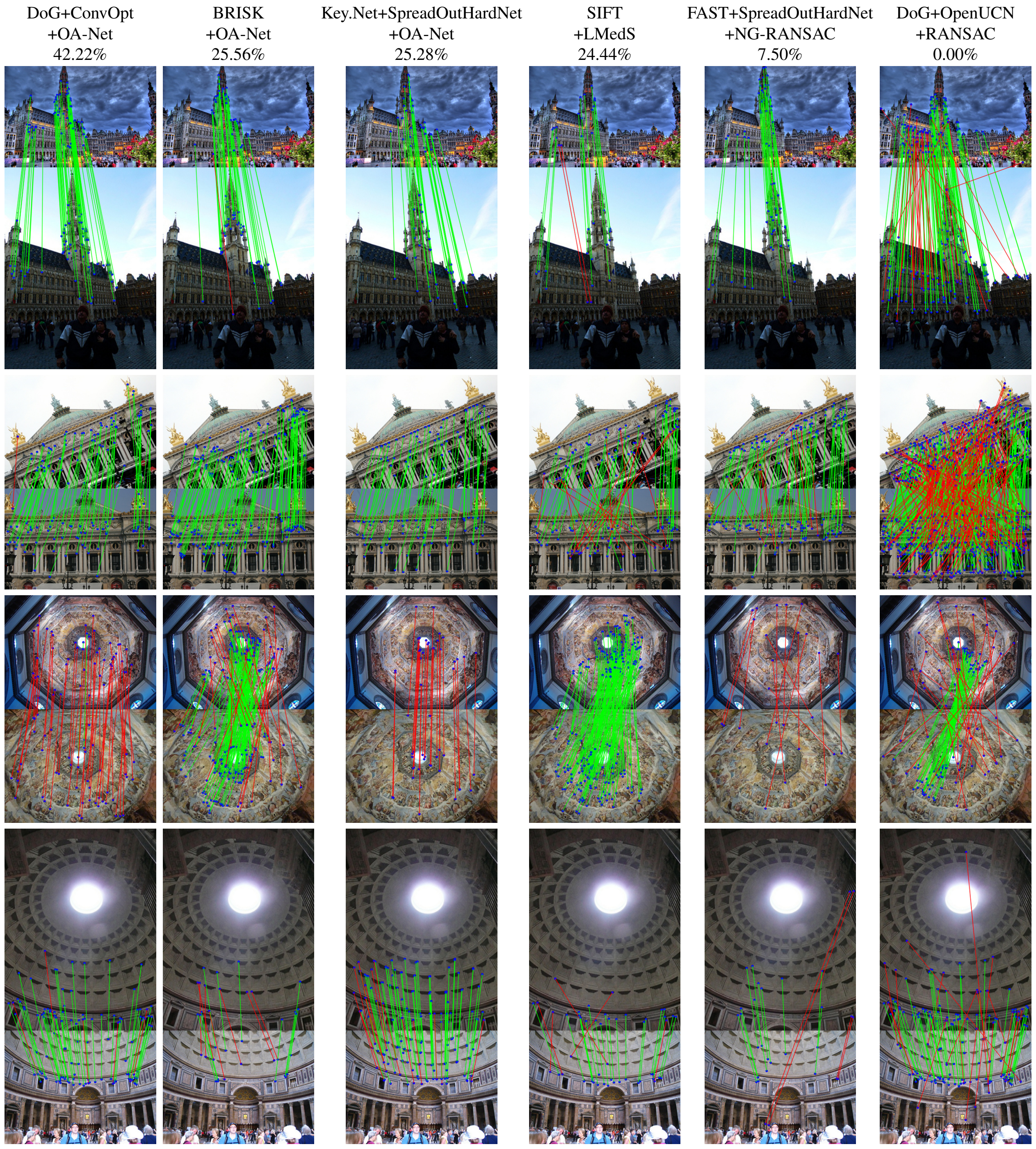


Figure 3.6: Performance of six **DMV** systems on YFCC-100M. Rows 1-2: *Grand Place Brussels*, Rows 3-4: *Paris Opera*, Rows 5-6: *Florence Cathedral Dome Interior*, Rows 7-8: *Pantheon Interior*.

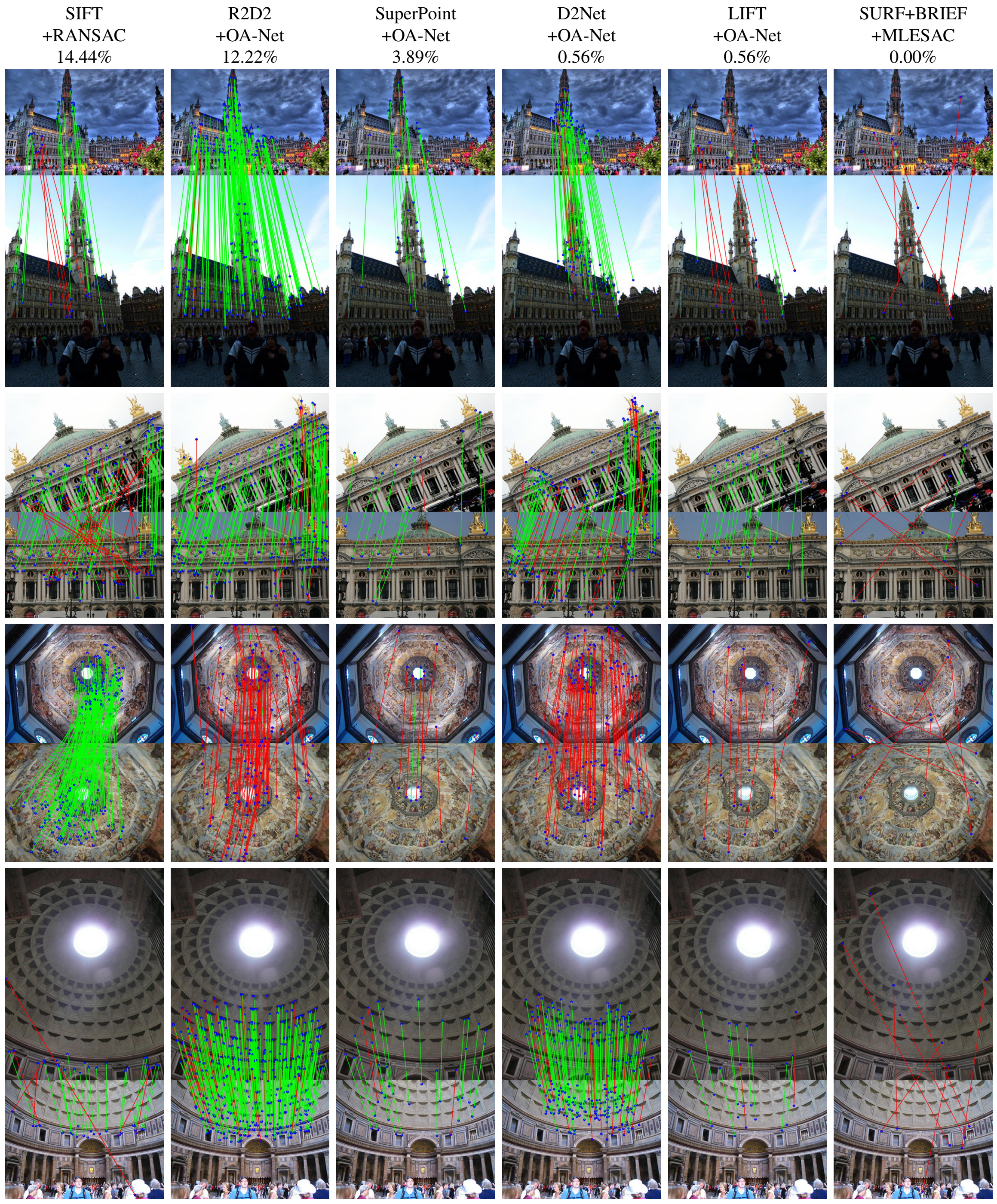


Figure 3.7: Performance of six **DMV** systems on YFCC-100M. Rows 1-2: *Grand Place Brussels*, Rows 3-4: *Paris Opera*, Rows 5-6: *Florence Cathedral Dome Interior*, Rows 7-8: *Pantheon Interior*.

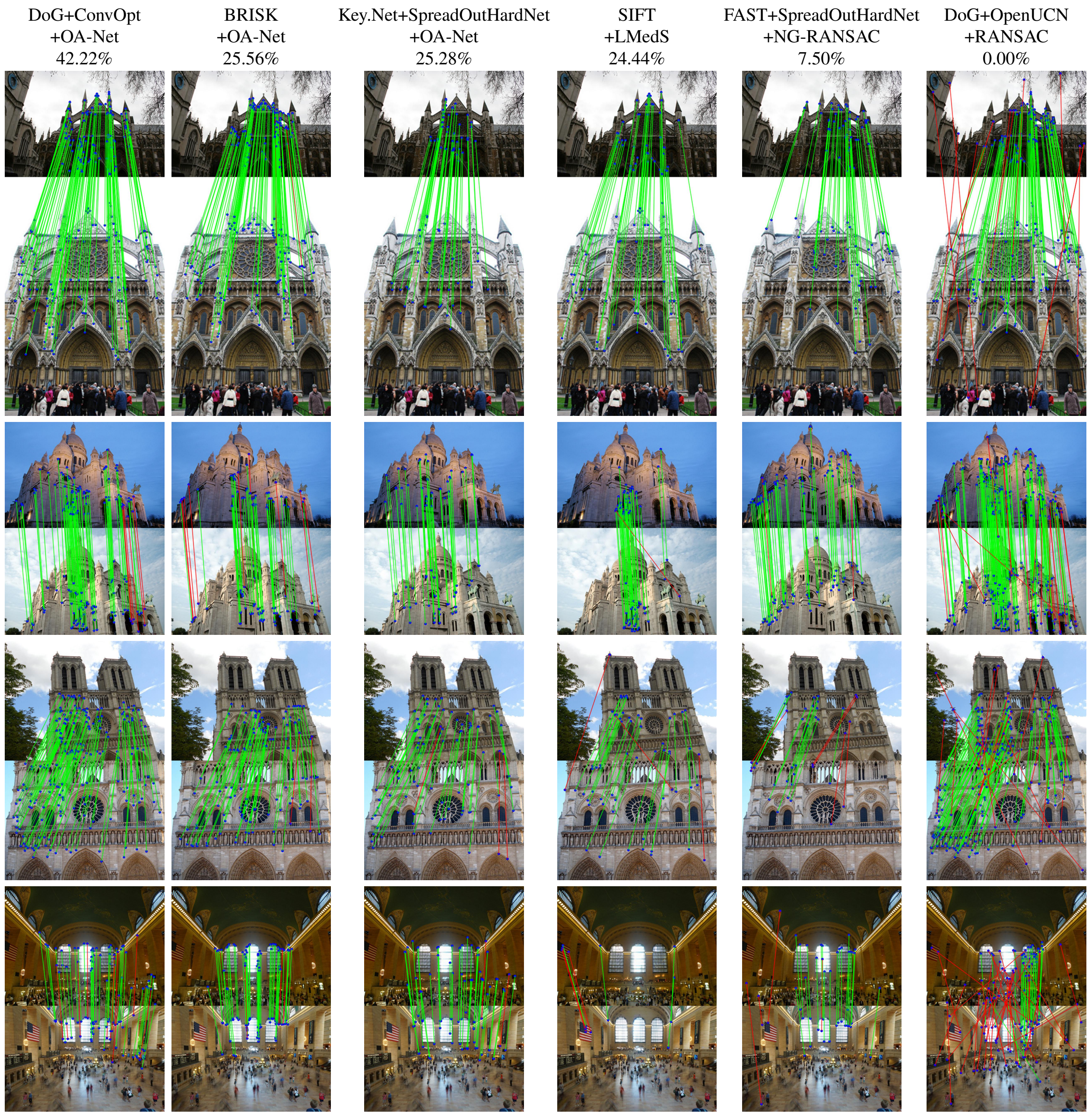


Figure 3.8: Performance of six **DMV** systems on YFCC-100M. Rows 1-2: *Westminster Abbey*, Rows 3-4: *Sacre Coeur*, Rows 5-6: *Notre Dame Front Facade*, Rows 7-8: *Grand Central Terminal*.

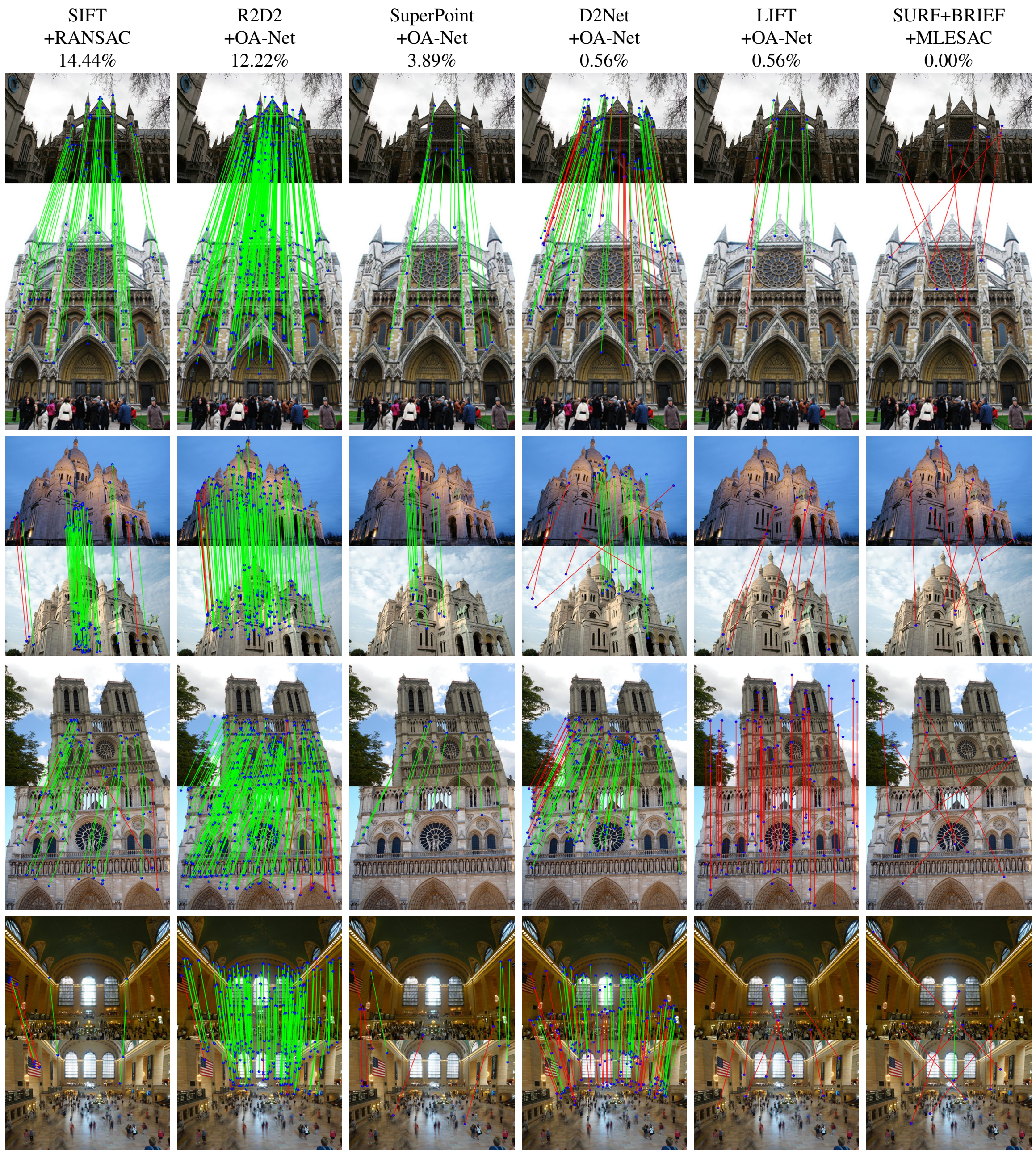


Figure 3.9: Performance of six **DMV** systems on YFCC-100M. Rows 1-2: *Westminster Abbey*, Rows 3-4: *Sacre Coeur*, Rows 5-6: *Notre Dame Front Facade*, Rows 7-8: *Grand Central Terminal*.

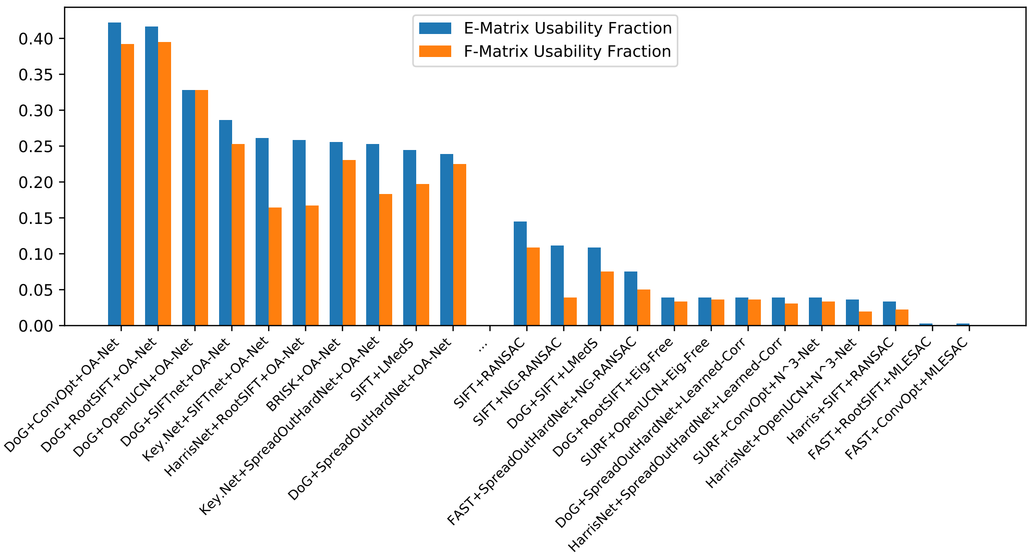


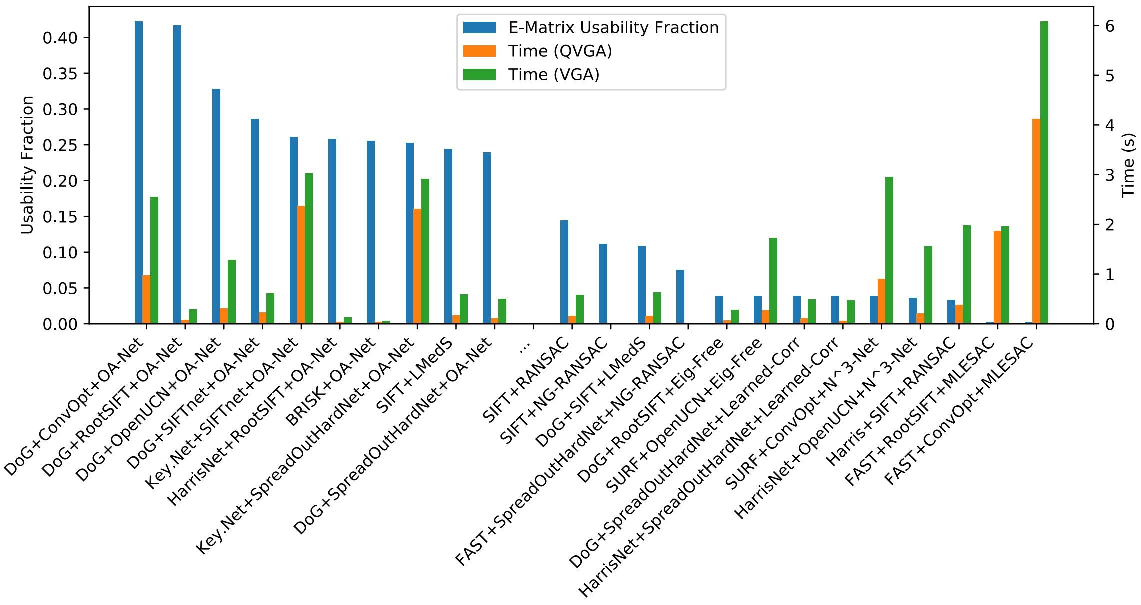
Figure 3.10: Wide-baseline stereo **DMV** (w**DMV**) results. (From left to right): the top-10 entries out of 808, sorted by E-matrix usability fraction (UIM-LB). Rightmost 11 entries: a representative sampling of other methods in the remaining $808 - 10 = 798$ methods .

We provide a graphical representation of quantitative results in Figure 3.10. Because of the huge amount of entries in each table, we provide a link[1] to CSV files with the complete results. Because performance comes at a cost, we additionally report the computational cost (in feedforward runtime) per method in the Appendix. We also provide qualitative examples of verified correspondences from our top method *(DoG + ConvOpt+OA-Net)* and a traditional system *(SIFT+RANSAC)* in Figure 3.6. Surprisingly, none of the recently proposed joint detector-descriptors [42, 72, 43, 39, 68] appear in the top-10 entries, when ranked by usability for the F-matrix (UIF). While intuition might suggest that RANSAC should provide a system with near 100% verification precision, we see in practice that this is not the case. If there are insufficient inliers even to form a minimal set, one can expect a degenerate E or F matrix estimate. Like Jin *et al.* [214], we note that simultaneously increasing the strictness of the inlier threshold and number of iterations for RANSAC provides a significant performance boost; decreasing the inlier threshold from 3 px (OpenCV default) to 0.5 px and increasing the correctness confidence from 0.99 (OpenCV default) to 0.999 increases the UIF of SIFT+RANSAC from 14% to 25%.

3.6 Additional Runtime Experiments

In this section, we perform a comprehensive evaluation of runtime requirements for each **DMV** combination. We test each combination in isolation, averaging 50 passes over each of 4 image pairs (200 forward passes per combination). We use a 4-core CPU (Intel(R) Xeon(R) CPU @ 2.20GHz, 15 GB RAM) with a NVIDIA Tesla T4 GPU (14 GB RAM).

[1]https://github.com/deep-front-ends/deep-front-ends



(a)

Figure 3.11: **wDMV** (YFCC-100M) combined runtime analysis.

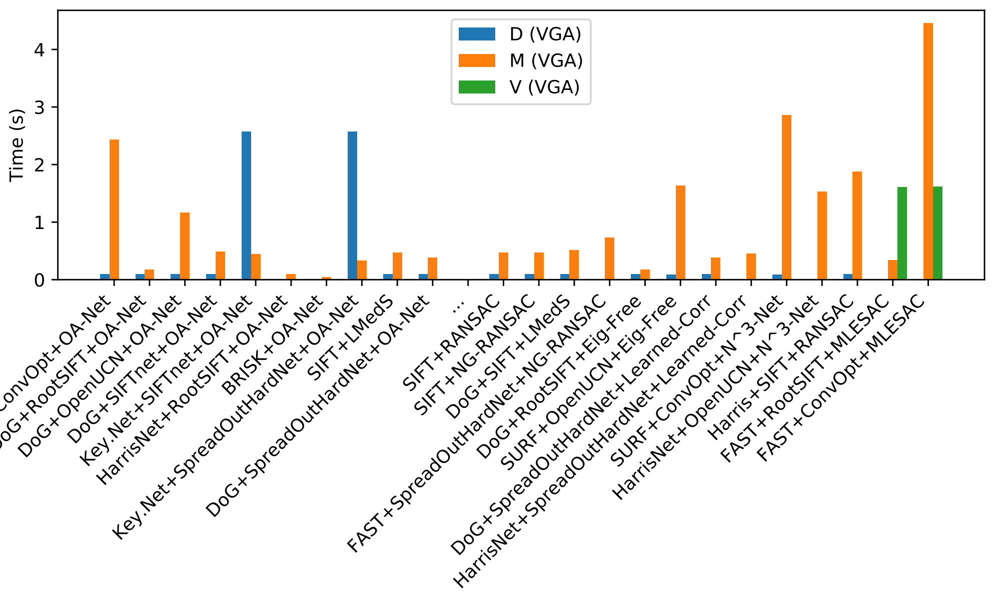
All methods that can utilize a GPU are executed on a GPU, else run on the CPU. We run this timing analysis on two image resolutions: VGA (480×640) and QVGA (240×320).

In Figure 3.11, we show a comparison of combined **DMV** runtime for the top-10 methods, and a representative sampling of other methods. Although *Dog+ConvOpt+OA-Net* has slightly higher Usable Image Fraction (UIF) than *Dog+RootSIFT+OA-Net* (42.22% vs. 41.67%), its drastically higher runtime (2.551 sec. vs. 290 millisec.) demonstrates that for practitioners with real-time requirements, instead *Dog+RootSIFT+OA-Net* is a superior choice.

In Figure 3.12, we show a comparison of where performance costs are paid per stage in a DMV pipeline. Key.Net [31] detection is exceptionally slow, dramatically slowing down the *Key.Net +SIFTNet+OA-Net* combination. MLESAC [201] is the slowest verifier. We see that RootSIFT's performance costs are significantly cheaper for descriptor formation and matching than ConvOpt [78] or Open-UCN [40, 197], leading to significant runtime gains across the **DMV** system.

3.7 Discussion: Deep Front Ends

In the above sections of this chapter, we have given a snapshot of the state of the art in the front-end for visual SLAM and SfM. We discover that deep detectors optimized for repeatability cannot beat a combination that includes a classical detector such as DoG when it comes to usability for the Essential or Fundamental matrix. Our results suggest that optimizing for repeatability may not be the optimal approach to building a front-end for SLAM or SfM. We find a surprising parity between deep and classical descriptors for usability, and find OA-Net [97] provides excellent verification performance for practitioners. We make



(a) at VGA resolution

Figure 3.12: **wDMV** (YFCC-100M) per-stage runtime analysis. Runtime comparison of the top-10 performing methods, and a sampling of other methods from the 808 tested. At VGA resolution (shorter image size is 480 px)

our code (including a full suite of unit tests for each evaluation metric) publicly available at https://github.com/borglab/gtsfm.

3.8 GTSFM: Incorporating the Deep Front End

Building accurate maps of the world is essential for spatial artificial intelligence (AI), with applications from autonomous robots to AR/VR. Structure from Motion and multi-view stereo (MVS) have proven to be effective methods for creating maps with vision-only inputs. For certain types of scenes with simple to medium complexity, high-fidelity world models can be easily extracted with tools such as COLMAP [216]. This pseudo-ground truth enabled new breakthroughs in machine learning, enabling methods such as NERF [217], NerFormer [218], accurate monocular depth predictions for humans [219], and much more.

However, state-of-the-art SfM techniques are incremental and accordingly are slow on large datasets. Incremental SfM commences by finding a good first image pair, then triangulating 3D points from two-views, then adding one additional image pair at a time, registering it to the 3d points, then performing bundle adjustment, removing outliers, and continuing until all possible image pairs have been registered. This is certainly not the only possible approach; *global* SfM methods have also been explored for some time [220, 103, 221, 222, 146, 145, 223], but are known to suffer from poor accuracy [224].

Why is global SfM not sufficiently accurate? Without noise from ‘front-end’ measurements, Global SfM would be exact. A single false positive can seriously degrade perfor-

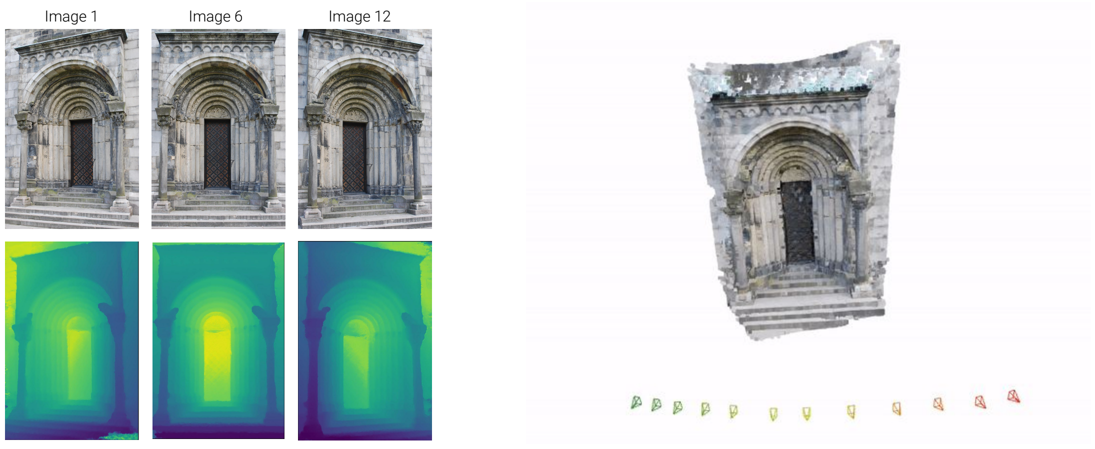


Figure 3.13: Qualitative results of GTSFM on the *Lund Door* dataset, consisting of 12 images. **(Left)** Depth maps, generated by PatchmatchNet [215]. **(Right)** Multi-view stereo output (aggregation of backprojected depth maps).

mance. A key problem is that reasoning about outliers is challenging. Techniques from sequential methods, such as filtering out measurements inconsistent with the current model at each step, are not directly applicable in a global setting. It is harder to reason a priori about which measurements are unreliable.

One way to think about the SfM or SLAM problem is to divide SfM into a front-end, and a back-end (optimization). The hardest part of SfM is correspondence, and when to trust correspondences, and of all the places where deep learning can be injected into the geometric modeling of SfM, feature matching is the most apparent part.

In this work, we aim to overcome these accuracy shortcomings by injecting deep learning into the SfM front-end. Deep front-ends still produce some noisy measurements, so we investigate new methods to make the system robust to this noise.

3.9 Related Work

A traditional SfM system computes keypoints, descriptors, matches, and verifies correspondences. But very little, if any of 20 years of research on using machine learning for the front-end, has been translated into practice or incorporated into libraries today, from OpenSfM [19] to OpenMVG [19] to Theia [20] to COLMAP [216]. A generation of research (two decades) in machine learning for the 'front-end', but no state-of-the-art SfM system today uses deep learning. SfM systems were created before deep learning began to show promise in this domain, thus all components are hand-crafted. Furthermore, end-to-end methods for SfM aren't accurate enough. Accordingly, we utilize local features and well-modeled geometry in the back-end.

3.9.1 Incremental SfM

Incremental SfM traditionally uses point correspondences to iteratively establish camera poses and global structure. Pollefeys *et al.* [225] introduced some of the modern frame-

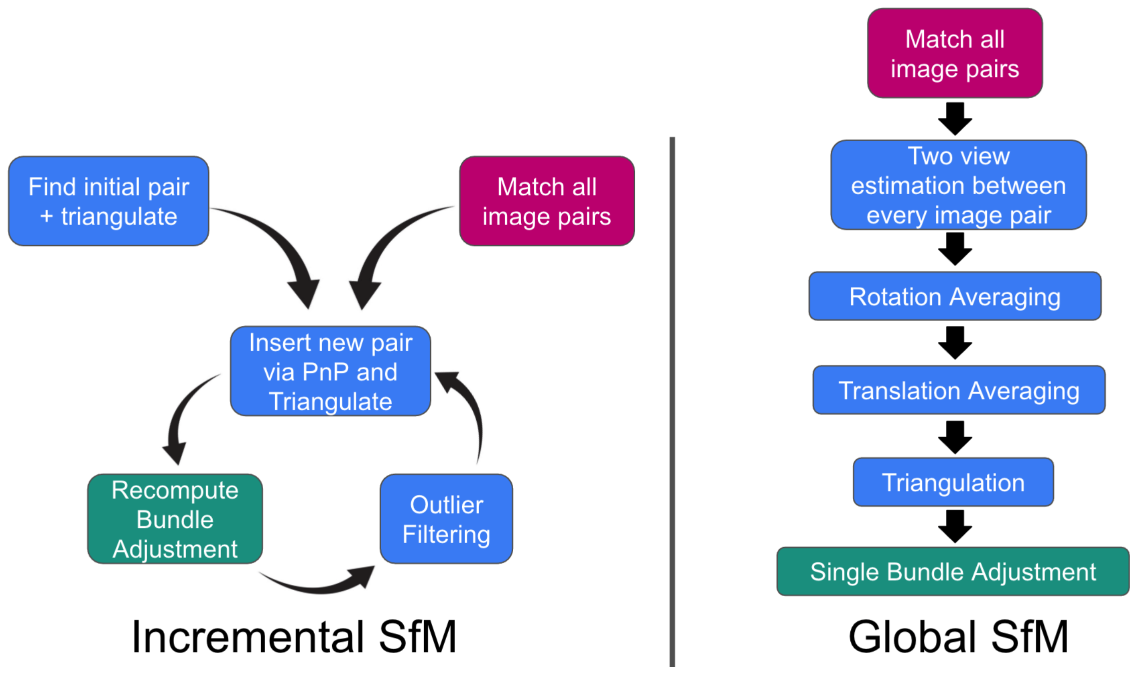


Figure 3.14: System diagrams of Incremental vs. Global SfM.

work for incremental SfM, which was expanded to massive datasets in Bundler [226], VisualSfM [227], and COLMAP [216]. Benchmarks such as Tanks and Temple [224] indicate that COLMAP represents the state-of-the-art over both incremental and global SfM, but COLMAP can be slow in practice. COLMAP has been extended in many ways, such as the use of feature volumes to refine track measurements in Pixel-Perfect SfM [228].

3.9.2 Global SfM

In Global SfM, also known as *non-sequential* SfM or *batch* SfM, one matches all possible image pairs, obtains a large number of two-view pose constraints, synchronizes all of these binary rotation measurements with some form of least squares, then estimates the camera positions, triangulate 3D points, and use a single bundle adjustment to refine points and poses. Both incremental and global SfM are subject to a feature matching stage with $O(n^2)$ complexity for n images. Global SfM is not new – Govindu introduced formulations for it two decades ago [220, 103, 221].

An advantage of Global SfM is its ability to exploit redundancy. For a graph with nodes as camera poses, and edges as two-view pose measurements, we can exploit all of the links in a graph, to average out noise and distribute error evenly across the entire graph. For a dataset of N images, there can be up to $\frac{N(N-1)}{2}$ pairs for which the relative motions can be estimated, potentially providing a highly redundant set of observations can be efficiently averaged [103]. However, the community has yet to find techniques to use this redundancy to an advantage in accuracy.

Many global SfM systems rely upon rotation averaging for accurate bundle adjustment initialization [229]. OpenMVG [19] uses Martinec and Pajdla's least-squares rotation averaging [230] technique. New rotation averaging methods have recently been proposed, such as Shonan Rotation Averaging [119] and Rotation Coordinate Descent (RCD) [231].

DISCO [222] initializes a solution using a discrete Markov random field (MRF). Distributed SfM has been explored in HyperSfM [232, 233] and DAGSfM [234].

Other approaches to large-scale SfM rely upon external measurements, such as GPS, IMU, and wheel encoders. For example, Google's city-scale SfM project, Street View SfM, uses 9.2 billion panoramic images [235] by combining local models. They avoid $O(n^2)$ matching by using the natural linear path of the vehicle trajectory to establish potential connectivity between images; only images within a fixed window of each other along the path are considered for joint participation in image tracks. A window size of 1500 cameras (a 100-panorama window of a 15-camera rosette) is used, and sub-centimeter accurate relative poses come purely from IMU measurements, instead of from RANSAC.

3.9.3 Outlier Rejection for SfM

Outlier rejection is critical to successful SfM. Not only is it very difficult to triangulate points from inexact camera positions, but Bundle Adjustment with Gaussian noise models cannot deal with outliers. While incremental systems can reject outliers at each registration stage via reprojection error, global SfM does not enjoy this privilege, and its performance is heavily reliant upon low outlier rates. Global SfM systems instead utilize a number of elaborate outlier rejection techniques to eliminate noisy measurements to prevent them from playing a role in joint optimization.

Relative Pose Consistency The most common outlier rejection approaches rely upon cycle consistency [236] of relative measurements within triplets. For example, [147] accumulating these deviations over a large set of loops one can obtain the statistics needed to infer the the set of false positives. Likewise, Enqvist, Kahl, and Olsson [146] and Moulon *et al.* in OpenMVG [145, 19] by composing rotations in a cycle and measuring the deviation from identity. 1dSfM [209] rejects outlier translation directions based on consistent ordering on 1d projections. Theia [20] also uses filtering based on global-to-relative agreement heuristics. Instead of using hand-crafted heuristics, Phillips [237] uses graph neural networks (GNN) to introduce learning-based cycle consistency on the keypoint match graph, instead of relative pose graph.

RANSAC Another technique for outlier rejection is to generate random spanning trees from relative poses in a RANSAC-like scheme [229] for estimating global camera poses, such that

$$ {}^{w}\mathbf{R}_i = {}^{w}\mathbf{R}_j \left({}^{j}\mathbf{R}_i \right) \tag{3.1} $$

roughly holds for as many relative rotations as possible.

Point Correspondence Consistency Other approaches reason about the consistency of point correspondences. Sweeney *et al.* [223] optimizes a cost function that penalizes epipolar transfer error [238] in each of 3 views for image triplets.

3.9.4 Multi-View Stereo

MVSNet [239] was a pioneering work, which showed how to use differentiable homography warping to create a 3D cost volume. PatchmatchNet [215] removed the 3D cost volume by introducing an iterative multiscale Patchmatch [240] in an end-to-end trainable architecture. Some MVS methods are scene-specific, e.g. NeRF and many of its variants. NeRF [217] allows generated depth maps for rendered novel views as the expected termination of each camera ray in the encoded volume. MVSNerf [241] combine NeRF and learning-based MVS, and evaluate depth reconstruction error, instead of only view synthesis quality.

3.10 Approach

GTSFM is a new global SfM library that exploits the following modules:

3.10.1 Front-End

We use a deep detector, SuperPoint [42], with a deep matcher, SuperGlue [76], and a RANSAC verifier. We optionally follow with two-view bundle adjustment as recommended by [242].

3.10.2 Outlier Rejection

We experiment with a number of different outlier rejection types.
Rotation Cycle Consistency In a noise-free setting, we have a consistency criterion over loops L of image pairs e_k:

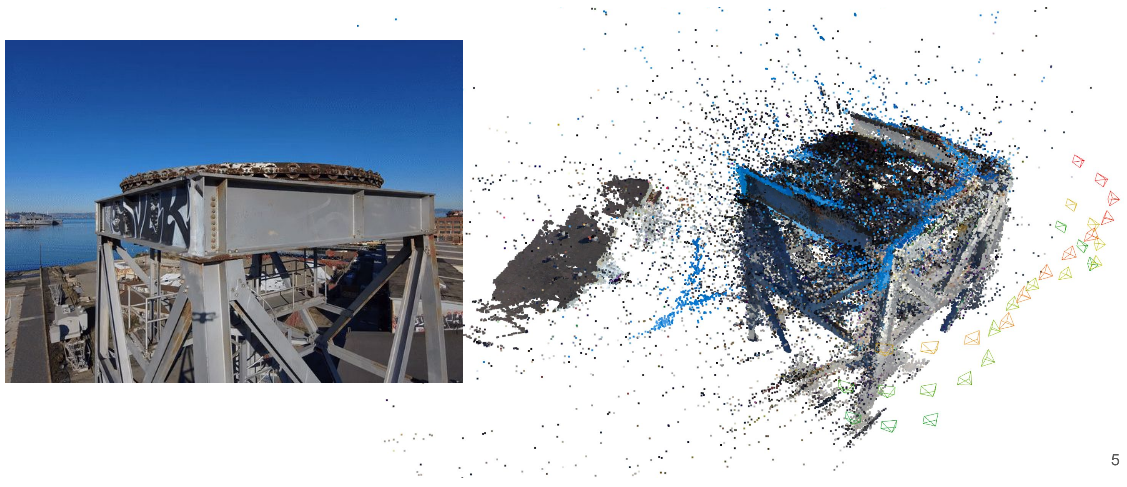
$$\mathbf{R}_L = \mathbf{R}_{e_{|L|}} \times \cdots \times \mathbf{R}_{e_1} = I, \quad e_k \in L \tag{3.2}$$

However, in a setting with noisy measurements $\mathbf{R}_{e_k}$, we can find all loops $L \in \mathcal{L}_k$ that an image pair e_k participates in, i.e. $\mathcal{L}_k = \{L \mid e_k \in L\}$ and accept those with a median cycle error below a certain threshold ϵ_{cycle}. More formally, we assign a label $\{y_k\} \in \{0, 1\}$ for each image pair edge e_k according to:

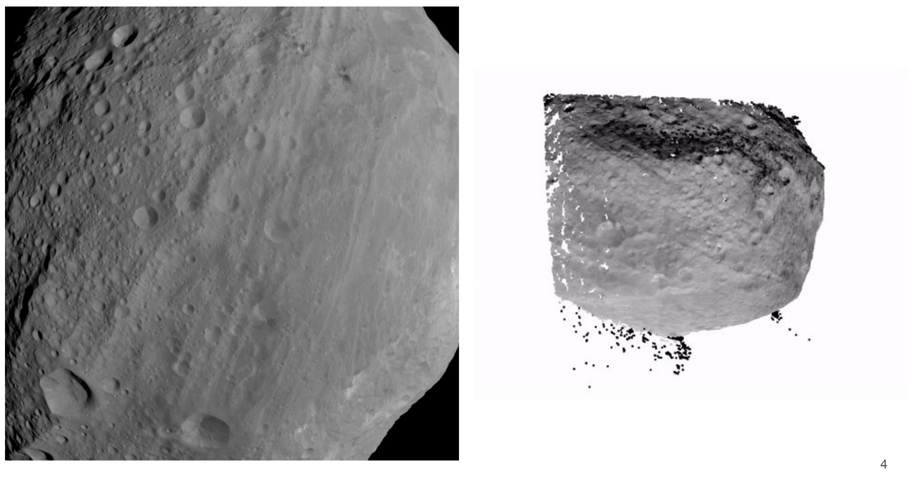
$$y_k = \underset{L \in \mathcal{L}_k}{\text{median}} \Big\{ \| \log(\mathbf{R}_L) \|_2 \mid e_k \in L \Big\} < \epsilon_{cycle} \tag{3.3}$$

where $\log$ is the logarithmic map, extracting the magnitude of the axis-angle representation. We reject all image pairs that fall into $\mathcal{E}_{reject} = \{e_k \mid y_k = 0\}$. In practice, we find that the `median` operator provides a better tradeoff of precision and recall than the `min` operator, which optimizes recall but not precision (as it allows false positives).
Epipolar Point Transfer Given known epipolar geometry between a triplet of images, we can (in certain cases) know exactly where a corresponding keypoint should be found within a 3rd image. This allows us to "transfer points" to a 3rd image, using intersection of epipolar lines (via Fundamental matrices). Hartley and Zisserman [238] (on p. 380) show how to do this. Suppose we know the three fundamental matrices F_{21}, F_{31} and F_{32} relating the three views, and let points $\mathbf{x}$ and $\mathbf{x}'$ in the first two views be a matched pair. We wish to find the corresponding point $\mathbf{x}''$ in the third image.

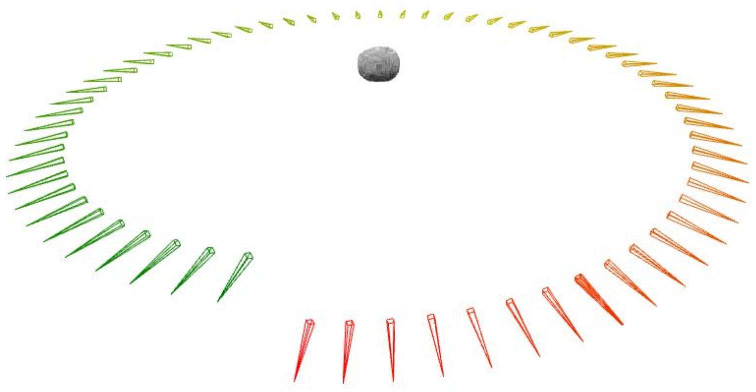
The required point $\mathbf{x}''$ matches point $\mathbf{x}$ in the first image, and consequently must lie on the epipolar line corresponding to $\mathbf{x}$. Since we know F_{31}, this epipolar line may be computed, and is equal to $F_{31}\mathbf{x}$. By a similar argument, $\mathbf{x}''$ must lie on the epipolar line



(a)



(b)



(c)

Figure 3.15: Qualitative results of GTSFM on the *Skydio-Crane-Mast-32* dataset **(top row)**, and NASA asteroid data, as captured by a telescope during the RC-3 of the Dawn spacecraft as it entered an Rotation Characterization 3 (RC3) orbit around the asteroid *Vesta* **(middle and bottom rows)**.

$F_{32}\mathbf{x}'$. Taking the intersection of the epipolar lines gives:

$$\mathbf{x}'' = (F_{31}\mathbf{x}) \times (F_{32}\mathbf{x}') \tag{3.4}$$

Similar details can be found in Section 3.1 of [223]. However, this method suffers from degeneracy near the trifocal plane, and we find it to be less effective.
Trifocal Tensor Point–point–point correspondence. Given point $\mathbf{x}$ in view 1, $\mathbf{x}'$ in view 2, and $\mathbf{x}''$ in view 3, ...

$$[\mathbf{x}']_\times \Big(\sum_i x^i \mathbf{T}_i \Big) [\mathbf{x}'']_\times = \mathbf{0}_{3\times 3} \tag{3.5}$$

We follow the implementation of Julia and Monasse [242], but instead of extracting camera poses from $\mathbf{T}$, we count the number of inliers.

3.10.3 Rotation Averaging

We solve for the rotations of global camera poses via multiple rotation averaging [243]. We use only the relative rotation measurements corresponding to image pair edges in the largest connected component of the relative pose graph. The problem can be defined as:

$$\underset{\mathbf{R_1},\dots\mathbf{R_n} \in SO(3)}{\text{argmin}} \sum_{(i,j)\in\mathcal{N}} d({}^i\overline{\mathbf{R}}_j, {}^i\mathbf{R}_w {}^w\mathbf{R}_j) \tag{3.6}$$

(modifying Hartley's notation).

We use Shonan [119]. In Shonan Averaging [119], the maximum likelihood problem is posed as:

$$\max_{\mathbf{R}\in SO(d)^n} \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \text{tr}({}^w\mathbf{R}_i {}^i\overline{\mathbf{R}}_j {}^j\mathbf{R}_w) \tag{3.7}$$

In the paper's notation,

$$\max_{\mathbf{R}\in SO(d)^n} \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \text{tr}(\mathbf{R}_i \overline{\mathbf{R}}_{ij} \mathbf{R}_j^T) \tag{3.8}$$

Shonan uses a convex relaxation based off of

$$f^*_{MLE} = \min_{\mathbf{R}\in SO(d)^n} \text{tr}\left(\overline{\mathbf{L}}\mathbf{R}^T\mathbf{R}\right) \tag{3.9}$$

where $\mathbf{R} = (\mathbf{R}_1, \dots, \mathbf{R}_n)$ is the $d \times dn$ matrix of rotations $\mathbf{R}_i \in SO(d)$, and $\overline{\mathbf{L}}$ is the connection Laplacian, a symmetric $(d \times d)$-block-structured matrix constructed from the measurements ${}^i\overline{\mathbf{R}}_j$.

3.10.4 Translation Averaging

Given camera rotations in a global frame, and pairwise translation directions, we recover the position of each camera (translation in a global frame. We use 1dSfM [209], which

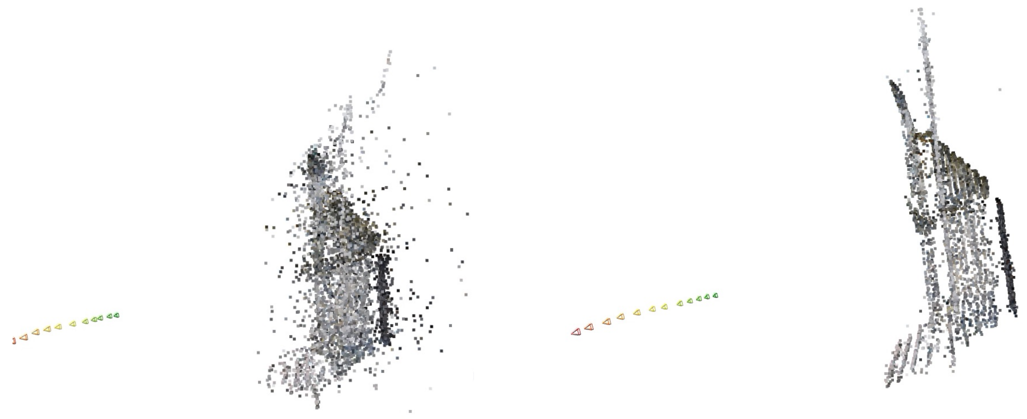


Figure 3.16: Qualitative results on *Lund-Door-12* dataset, before **(left)** and after **(right)** bundle adjustment and filtering of 3d points by reprojection errors.

optimizes a chordal error:

$$err_{ch}(\mathcal{T}) = \sum_{(i,j)\in\mathcal{E}} d_{ch}\Big(\hat{\mathbf{t}}_{ij}, \frac{\mathbf{t}_j - \mathbf{t}_i}{\|\mathbf{t}_j - \mathbf{t}_i\|}\Big)^2 \tag{3.10}$$

where d_{ch} is defined as:

$$d_{ch}(\mathbf{u},\mathbf{v}) = \|\mathbf{u}-\mathbf{v}\|_2 \tag{3.11}$$

3.10.5 Data Association + Triangulation

We triangulate points using DLT followed by non-linear refinement.

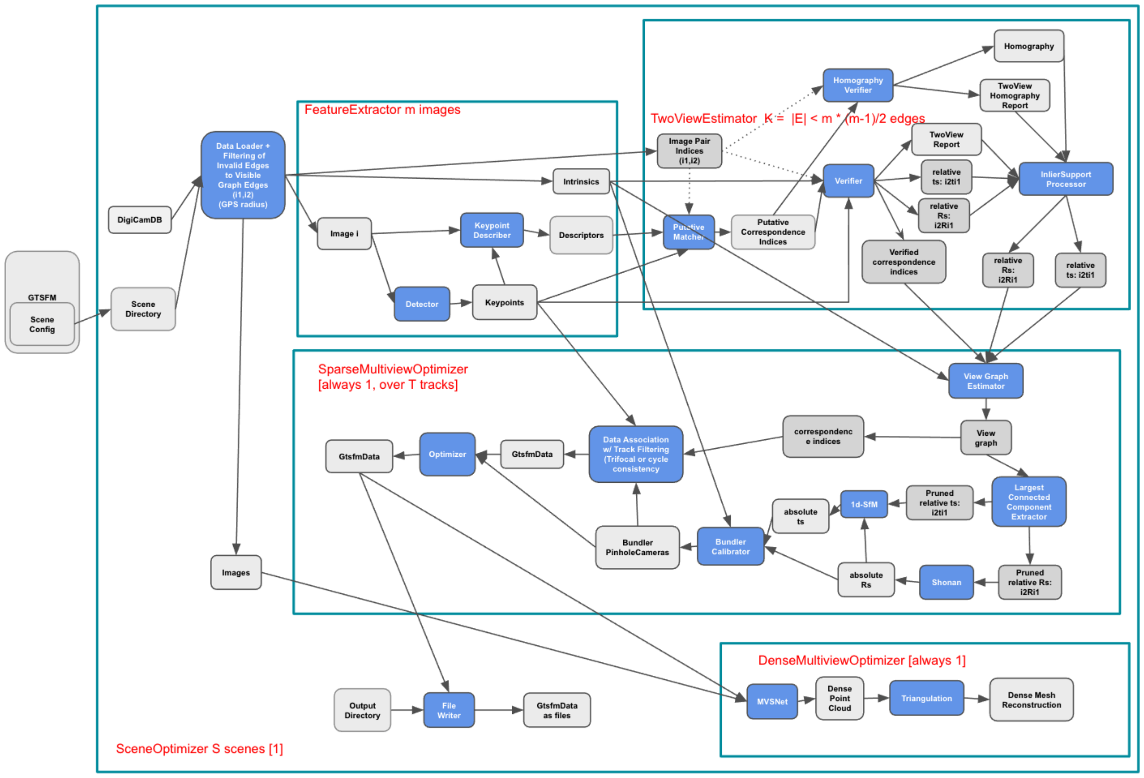
3.10.6 Bundle Adjustment

We refine the initial camera poses and triangulated point cloud using bundle adjustment (see Figure 3.16). We use the Bundler camera model for calibration, with a single focal length f, two radial distortion coefficients k_1, k_2 (quadratic and quartic), and a image center (u_0, v_0) (principal point) in pixels. We initialize the 6-dof SE(3) camera poses from the global camera rotations estimated from Shonan Rotation Averaging and the global camera positions estimated from 1dSfM, and refine points and camera parameters using the Levenberg-Marquardt algorithm [244, 245].

Bundle adjustment is an optimization problem that refines an initial estimate. It can be formally defined as follows: Given sparse points $\{\mathbf{P}_j \in \mathbb{R}^3\}$, intrinsic parameters $\mathbf{C}_i$ of the cameras, and camera poses $\{({}^w\mathbf{R}_i, {}^w\mathbf{t}_i) \in SE(3)\}_{i=1}^N$ for each image, represented as rotation matrices and translation vectors [1]:

$$E_{BA}(\cdot) = \sum_j \sum_{(i,u)\in\mathcal{T}(j)} \|\Pi({}^i\mathbf{R}_w\mathbf{P}_j + {}^i\mathbf{t}_w; \mathbf{C}_i) - \mathbf{p}_u\|_\gamma \tag{3.12}$$

where $\mathcal{T}(j)$ is the set of images i and keypoints u in track j, $\Pi(\cdot)$ projects to the image plane, and $\|\cdot\|_\gamma$ is a robust norm.



GTSFM
Scene Config
DigiCamDB
Scene Directory
Data Loader + Filtering of Invalid Edges to Visible Graph Edges (i1,i2) (GPS radius)
FeatureExtractor m images
Intrinsics
Image i
Keypoint Describer
Descriptors
Detector
Keypoints
TwoViewEstimator K = |E| < m * (m-1)/2 edges
Image Pair Indices (i1,i2)
Putative Matcher
Putative Correspondence Indices
Homography Verifier
Homography
TwoView Homography Report
TwoView Report
Verifier
relative ts: i2ti1
relative Rs: i2Ri1
InlierSupport Processor
Verified correspondence indices
relative Rs: i2Ri1
relative ts: i2ti1
SparseMultiviewOptimizer [always 1, over T tracks]
View Graph Estimator
View graph
correspondence indices
Data Association w/ Track Filtering (Trifocal or cycle consistency
GtsfmData
Optimizer
GtsfmData
Largest Connected Component Extractor
Pruned relative ts: i2ti1
1d-SfM
absolute ts
Bundler PinholeCameras
Bundler Calibrator
absolute Rs
Shonan
Pruned relative Rs: i2Ri1
Images
DenseMultiviewOptimizer [always 1]
MVSNet
Dense Point Cloud
Triangulation
Dense Mesh Reconstruction
Output Directory
File Writer
GtsfmData as files
SceneOptimizer S scenes [1]

Figure 3.17: GTSFM system architecture.

3.10.7 Multi-View Stereo

We experiment with PatchmatchNet [215] and Instant NGP [246]. PatchmatchNet estimates depth maps for each reference frame *independently* using source images are used as evidence. For cameras with known intrinsics, e.g.

$$\begin{bmatrix} u \cdot d \\ v \cdot d \\ d \end{bmatrix} = K_{ref} * p_c \tag{3.13}$$

Depth maps are then backprojected into a 3d point cloud via:

$$p_c = K_{ref}^{-1} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \cdot d \tag{3.14}$$

and then brought into the world frame via:

$$p_w = {}^wT_c * p_c = {}^wT_c * \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.15}$$

Instant-NGP [246], on the other hand, estimates 3d structure *jointly* using a neural field.

3.11 Experimental Results

We now present results on 4 datasets – one of easy difficulty, two of moderate difficulty, and one of extreme difficulty.

3.11.1 Datasets

Lund-Door-12 The first, the simplest, is Carl Olsson's Lund Door 12 image dataset[2], featuring very small baselines and almost no occlusion. Exhaustive matching over the 12 images yields 66 image pairs.
Skydio-Crane-Mast-501 A dataset of extreme difficulty, consisting of 501 images captured in Crane Cove Park, San Francisco[3], featuring object symmetry, repetitive features, thin structures, and extreme depth ranges (from centimeters up to several miles). It is captured by a drone that makes multiple horizontal circular passes around a crane tower, and then makes multiple sweeps over the top of the tower. Sequential image matching over the 501 images with a 20 frame lookahead yields 9790 image pairs.
Skydio-Crane-Mast-32 A medium difficulty dataset, representing a 32-image subset of the aforementioned Skydio Crane Mast dataset, but with views of only a single face of the four crane mast faces, limiting the effect of false positives due to object symmetry. Exhaustive image matching over the 32 images yields 496 image pairs.
Notre-Dame-20 A medium difficulty dataset, representing a 20-image subset of the 715-image Notre Dame PhotoTourism dataset[4] [199]. Exhaustive image matching of the 20 images presents 171 image pairs.

3.11.2 Evaluation

We quantitatively measure the effect of different front-ends on GTSFM's performance, using as ground-truth the output of COLMAP [216] on each dataset, a high-quality incremental SfM system. Each front-end that we analyze includes Essential matrix estimation with RANSAC, using Nister's 5-Point algorithm [247]. We do not employ any two-view bundle adjustment postprocessing on the estimated relative pose, to avoid influencing the performance of the front-end. In order to remove the effect of spurious image pair measurements that would be rejected in a real SfM system, We report front-end relative pose errors only after discarding any image pairs with an inlier ratio below 10% and any image pairs with less than 15 absolute inliers (per COLMAP's convention [216]).

For several methods, in order to disambiguate the errors from the front-end vs. back-end optimization, we illustrate what access to a front-end 'oracle' provides. The oracle has access to ground truth pose errors, and thus our 'oracle' baselines discard erroneous front-end measurements if the relative rotation or relative translation angular error with respect to ground truth exceeds a threshold (5° in our experiments).

We evaluate 9 metrics per each front-end combination:

[2]Available at https://www.maths.lth.se/matematiklth/personal/calle/dataset/dataset.html.

[3]Available on Sketchfab at [download link] with high-resolution images provided by Skydio [download link].

[4]Available online at http://phototour.cs.washington.edu/datasets/.

Relative Rotation Angular Error: Defined as $\theta_{\text{rel.rot.error}} = \| \log \left({}^{i_2}\hat{\mathbf{R}}_{i_1}^{\top} \circ {}^{i_2}\mathbf{R}_{i_1} \right) \|_2$, measured in degrees, where (i_1, i_2) represent an image pair. We report both median and mean values, over all image pairs.
Relative Translation Angular Error: Defined as $\theta_{\text{rel.trans.error}} = \cos^{-1}\left(\frac{{}^{i_2}\hat{\mathbf{t}}_{i_1} \cdot {}^{i_2}\mathbf{t}_{i_1}}{\|{}^{i_2}\hat{\mathbf{t}}_{i_1}\| \|{}^{i_2}\mathbf{t}_{i_1}\|} \right)$, measured in degrees, where (i_1, i_2) represent an image pair. We report both median and mean values, over all image pairs.
Relative Pose Error Deg.: Defined as $\max(\theta_{\text{rel.rot.error}}, \theta_{\text{rel.trans.error}})$ the maximum of relative rotation angular error and relative translation angular error above, per the convention of [104]. We report both median and mean values, over all image pairs.
# Cameras Localized: The number of cameras for which a global pose is estimated, corresponding to the cameras located in the largest connected component of the relative pose graph after outlier rejection.
# Tracks (Unfiltered): Defined as the total number of keypoint tracks j over the entire dataset.
3d Track Length (Unfiltered): Defined as the number of views in each keypoint track j. We report both median and mean values, over all tracks.
Reprojection Error (Unfiltered): Defined as $\|\Pi({}^{i}\mathbf{R}_w \mathbf{P}_j + {}^{i}\mathbf{t}_w; \mathbf{C}_i) - \mathbf{p}_u\|_2$, for 3d point $\mathbf{P}_j$ from track j, view i, keypoint detection $\mathbf{p}_u$, and optimized camera pose $({}^{w}\mathbf{R}_i, {}^{w}\mathbf{t}_i)$, in the notation of Equation 3.12. Reprojection error is measured in pixels. We report both median and mean values, over all image pairs.
Global Rotation Angular Error: Defined as $\theta_{\text{global.rot.error}} = \| \log \left({}^{w}\hat{\mathbf{R}}_{i}^{\top} \circ {}^{w}\mathbf{R}_{i} \right) \|_2$, and measured in degrees. We report both median and mean values, over all camera poses.
Global Translation Angular Error: Defined as $\theta_{\text{global.trans.error}} = \cos^{-1}\left(\frac{{}^{w}\hat{\mathbf{t}}_{i} \cdot {}^{w}\mathbf{t}_{i}}{\|{}^{w}\hat{\mathbf{t}}_{i}\| \|{}^{w}\mathbf{t}_{i}\|} \right)$, and measured in degrees. We report both median and mean values, over all camera poses.

Table 3.2: Quantitative results on the Lund Door-12 dataset. As the dataset is quite simple, all methods (except SIFT + Mutal Nearest Neighbor Matcher + OANet + RANSAC) localize all 12 images in the largest connected component with very low error. Global rotation and global translation angular errors are below 2 degrees for all methods, in both mean and median. "Mutual NN" indicates a mutual nearest-neighbor matcher.

Method			Front-End Results			Back-End Results					
Detector + Descriptor	Matcher	Verifier	Relative Rotation Angular Error (Med. / Mean)	Relative Translation Angular Error (Med. / Mean)	Relative Pose Error (deg.) (Med / Mean)	# Cameras localized	# Tracks	3d Track Length (Unfiltered) (Med. / Mean)	Reproj. Error (Unfiltered) px (Med. / Mean)	Global Rotation Angular Error (deg.) (Med. / Mean)	Global Translation Angular Error (deg.) (Med. / Mean)
ORB	Mutual NN	RANSAC	1.00 / 1.29	1.75 / 4.99	1.95 / 5.04	12 / 12	777	2 / 3.21	0.37 / 0.48	0.12 / 0.14	0.68 / 1.11
Brisk	Mutual NN	RANSAC	0.45 / 0.50	0.90 / 1.26	0.90 / 1.30	12 / 12	8493	3 / 3.67	0.22 / 0.31	0.02 / 0.01	0.05 / 0.08
KAZE	Mutual NN	RANSAC	0.30 / 0.43	0.75 / 1.03	0.82 / 1.06	12 / 12	5081	4 / 5.58	0.22 / 0.39	0.02 / 0.03	0.07 / 0.08
DoG + ConvOpt	Mutual NN	RANSAC	0.29 / 0.34	0.61 / 0.98	0.61 / 0.99	12 / 12	7148	4 / 4.73	0.16 / 0.27	0.01 / 0.01	0.03 / 0.04
DoG + ConvOpt	Mutual NN	OANet + RANSAC	1.21 / 2.08	1.97 / 7.06	2.17 / 7.23	12 / 12	3133	3 / 3.96	0.11 / 0.17	0.06 / 0.06	0.24 / 0.31
SIFT	Mutual NN	OANet + RANSAC	1.22 / 2.46	2.73 / 7.70	3.05 / 7.97	3 / 12	3037	3 / 3.91	0.07 / 0.10	0.48 / 0.33	0.89 / 0.80
SIFT	Mutual NN	RANSAC	0.29 / 0.34	0.54 / 0.90	0.58 / 0.92	12 / 12	10065	4 / 4.82	0.16 / 0.26	0.02 / 0.02	0.04 / 0.05
SuperPoint	SuperGlue	RANSAC	0.44 / 0.52	0.81 / 1.22	0.84 / 1.29	12 / 12	3125	4 / 4.16	0.46 / 0.60	0.04 / 0.04	0.13 / 0.16

3.11.3 Quantitative Results

In Table 3.2, Table 3.3, Table 3.4, and Table 3.5, we present quantitative results on the four aforementioned datasets. In general, catastrophic front-end average errors (i.e. the presence of significant outliers) indicates that a high quality solution will not be recoverable from

Table 3.3: Quantitative results on the Notre-Dame-20 dataset, a medium-difficulty dataset. Many methods localize with low error, but SuperGlue yields the highest recall (most number of images localized in the largest connected component) while maintaining low global pose angular errors. The oracle is not needed for SuperGlue, but for SIFT, the oracle reduces global translation angular errors from 20° to 2° (a massive 10x reduction in this error metric). Metrics essential for qualitatively good scene reconstructions are shown in red. "Mutual NN" indicates a mutual nearest-neighbor matcher.

Method			Front-End Results			Back-End Results					
Detector + Descriptor	Matcher	Verifier	Relative Rotation Angular Error (Med. / Mean)	Relative Translation Angular Error (Med. / Mean)	Relative Pose Error (deg.) (Med / Mean)	# Cameras Localized	# Tracks	3D Track Length (Unfiltered) (Med. / Mean)	Reproj. Error (Unfiltered) px (Med. / Mean)	Global Rotation Angular Error (deg.) (Med. / Mean)	Global Translation Angular Error (deg.) (Med. / Mean)
ORB	Mutual NN	RANSAC	0.23 / 0.30	76.29 / 65.02	76.29 / 65.02	3 / 20	285	2 / 2.08	0.15 / 0.23	0.14 / 0.16	74.48 / 70.22
Brisk	Mutual NN	RANSAC	2.32 / 3.42	5.71 / 16.85	5.71 / 17.13	7 / 20	930	2 / 2.61	0.28 / 0.55	1.30 / 1.80	1.85 / 15.11
KAZE	Mutual NN	RANSAC	2.72 / 3.36	7.11 / 19.66	7.11 / 19.75	12 / 20	2220	2 / 2.95	0.31 / 0.59	0.68 / 1.03	3.80 / 11.14
DoG + ConvOpt	Mutual NN	RANSAC	1.41 / 3.14	4.43 / 11.23	6.10 / 11.54	11 / 20	1915	2 / 2.91	0.28 / 0.51	0.57 / 0.66	2.07 / 7.62
DoG + ConvOpt	Mutual NN	OANet + RANSAC	2.64 / 3.63	14.27 / 30.13	14.27 / 30.18	5 / 20	539	2 / 2.23	0.16 / 0.24	2.03 / 3.17	15.74 / 22.51
SIFT	Mutual NN	OANet + RANSAC	3.62 / 8.51	9.84 / 26.90	10.00 / 28.60	10 / 20	1101	2 / 2.48	0.19 / 0.30	0.62 / 0.94	2.82 / 7.36
SIFT	Mutual NN	RANSAC	2.41 / 5.55	4.92 / 13.46	5.46 / 14.83	17 / 20	3724	2 / 3.11	0.30 / 0.78	0.57 / 0.96	1.64 / 20.01
SIFT + Oracle	Mutual NN	RANSAC	1.33 / 1.40	1.78 / 2.03	2.29 / 2.27	13 / 20	2509	3 / 3.25	0.29 / 0.66	0.25 / 0.33	0.97 / 1.71
SuperPoint	SuperGlue	RANSAC	2.12 / 3.11	3.50 / 7.39	4.17 / 7.69	19 / 20	2330	2 / 2.44	0.47 / 0.72	0.58 / 0.84	1.45 / 3.12
Superpoint + Oracle	SuperGlue	RANSAC	1.69 / 1.85	2.29 / 2.25	2.45 / 2.61	19 / 20	1517	2 / 2.81	0.65 / 0.96	0.47 / 0.58	1.11 / 2.50

Table 3.4: Quantitative results on the Skydio-Crane-Mast-32 dataset, a medium-difficulty dataset. Superpoint+SuperGlue+Ransac-E alone recovers global translation angular errors of $< 14°$ in mean, SIFT+Mutual-Nearest-Neighbor-Matcher+Ransac-E has double the error at $26°$. This is evident in the reconstruction: SuperGlue's poses are accurate, while SIFT's are catastrophically poor. Metrics essential for qualitatively good scene reconstructions are shown in red. "Mutual NN" indicates a mutual nearest-neighbor matcher.

Method			Front-End Results			Back-End Results					
Detector + Descriptor	Matcher	Verifier	Relative Rotation Angular Error (Med. / Mean)	Relative Translation Angular Error (Med. / Mean)	Relative Pose Error (deg.) (Med / Mean)	# Cameras Localized	# Tracks	3D Track Length (Unfiltered) (Med. / Mean)	Reproj. Error (Unfiltered) px (Med. / Mean)	Global Rotation Angular Error (deg.) (Med. / Mean)	Global Translation Angular Error (deg.) (Med. / Mean)
ORB	Mutual NN	RANSAC	4.14 / 9.97	24.80 / 55.31	24.80 / 55.99	3 / 32	33	2 / 2.15	0.41 / 0.54	1.55 / 1.78	120.76 / 100.82
Brisk	Mutual NN	RANSAC	1.78 / 3.80	4.86 / 33.17	5.68 / 33.68	26 / 32	2995	2 / 2.41	0.83 / 1.77	1.41 / 1.58	41.89 / 47.72
KAZE	Mutual NN	RANSAC	3.38 / 8.17	7.88 / 46.33	8.77 / 47.68	28 / 32	5440	2 / 2.91	0.69 / 2.07	0.92 / 2.46	30.09 / 36.19
DoG + ConvOpt	Mutual NN	RANSAC	1.28 / 2.92	2.10 / 24.91	2.49 / 25.28	32 / 32	3514	2 / 2.74	0.70 / 1.41	1.01 / 1.11	27.54 / 52.41
DoG + ConvOpt	Mutual NN	OANet + RANSAC	3.16 / 10.05	42.16 / 53.85	42.16 / 54.12	3 / 32	68	2 / 2.13	0.15 / 0.22	0.29 / 0.27	12.70 / 10.70
SIFT	Mutual NN	RANSAC	1.85 / 4.55	2.91 / 33.80	3.63 / 34.34	32 / 32	5498	2 / 2.82	1.01 / 26.13	1.50 / 1.87	9.93 / 26.16
SIFT + Oracle	Mutual NN	RANSAC	1.15 / 1.38	1.20 / 1.45	1.62 / 1.85	25 / 32	4666	2 / 2.86	0.67 / 1.48	0.76 / 0.75	0.66 / 6.17
SuperPoint	SuperGlue	RANSAC	2.24 / 13.93	4.14 / 32.74	4.93 / 36.50	31 / 32	7678	2 / 2.65	0.87 / 1.31	0.52 / 0.82	0.66 / 13.35
Superpoint + Oracle	SuperGlue	RANSAC	1.13 / 1.36	1.35 / 1.67	1.70 / 2.04	32 / 32	9088	2 / 2.79	0.84 / 1.23	0.21 / 0.21	0.35 / 0.46

Table 3.5: Quantitative results on the Skydio-Crane-Mast-501 dataset, an extreme-difficulty dataset. All non-oracle methods with high recall (i.e. the ability to estimate at least 90 of the 501 cameras) fail catastrophically, with average global translation angular errors over 50°. Metrics essential for qualitatively good scene reconstructions are shown in red. "Mutual NN" indicates a mutual nearest-neighbor matcher.

Method			Front-End Results			Back-End Results					
Detector + Descriptor	Matcher	Verifier	Relative Rotation Angular Error (Med. / Mean)	Relative Translation Angular Error (Med. / Mean)	Relative Pose Error (deg.) (Med / Mean)	# Cameras Localized	# Tracks	3D Track Length (Unfiltered) (Med. / Mean)	Reproj. Error (Unfiltered) px (Med. / Mean)	Global Rotation Angular Error (deg.) (Med. / Mean)	Global Translation Angular Error (deg.) (Med. / Mean)
DoG + ConvOpt	Mutual NN	RANSAC	1.82 / 3.62	10.09 / 40.88	10.47 / 41.07	31 / 501	3937	2 / 3.11	0.73 / 1.39	1.48 / 2.08	2.39 / 9.74
SIFT	Mutual NN	OANet + RANSAC	2.84 / 7.33	59.06 / 65.45	59.28 / 66.43	19 / 501	1132	2 / 2.49	0.58 / 1.07	2.74 / 3.20	35.60 / 43.25
SIFT	Mutual NN	RANSAC	2.8 / 5.41	13.30 / 45.98	13.79 / 46.34	137 / 501	21626	2 / 3.07	3.29 / 7.66	9.98 / 9.61	50.75 / 54.78
SIFT + Oracle	Mutual NN	RANSAC	0.92 / 1.26	1.55 / 1.84	1.88 / 2.14	59 / 501	10881	2 / 2.60	1.10 / 4.34	4.73 / 4.99	27.46 / 36.02
SuperPoint	SuperGlue	RANSAC	4.83 / 39.69	20.85 / 48.70	27.02 / 65.13	91 / 501	26191	2 / 2.52	1.57 / 3.88	70.21 / 83.99	57.14 / 61.37
Superpoint + Oracle	SuperGlue	RANSAC	0.87 / 1.22	1.27 / 1.61	1.62 / 1.93	193 / 501	68913	2 / 2.50	0.74 / 1.09	5.68 / 5.61	7.43 / 15.44

back-end optimization. Among the many back-end optimization metrics we compute and analyze, we find that only two consistently correlate closely with the visual quality of the reconstruction – (1) the average global translation angular error (indicating the correct relative placement of cameras, as a measure of precision) and (2) the number of cameras localized by Global SfM (recall). In practice, we find that global rotation angular error can be low after bundle adjustment, even when the result is qualitative poor. However, the same is not true for translation angular error, which is generally indicate of significant errors in global pose estimation. We find that high *relative pose error* in the front-end is almost always correlated with high *global translation angular error* after translation averaging and bundle adjustment in Global SfM.

New deep front-ends that exploit learned matching [76] demonstrate superiority on the medium-difficulty datasets, *Notre-Dame-20* and *Skydio-Crane-Mast-32* (see Table 3.3 and Table 3.4), but do not provide a high enough signal-to-noise ratio to solve the most challenging of datasets, *Skydio-Crane-Mast-501*, where all front-ends perform poorly in a global SfM framework (see Table 3.5). For *Skydio-Crane-Mast-501*, this is likely due to the fact that most deep front-ends and deep matchers are trained to maximize recall on datasets where covisibility is always guaranteed, which is not the case "in-the-wild," leading to false positives. Perhaps unsurprisingly, on an easy benchmark, *Lund-Door-12*, both classical and deep-front-ends perform comparably and easily solve the task (see Table 3.2).

When comparing front-ends "in-the-wild", we find that a newly released method, *SuperPoint* + *SuperGlue* + *RANSAC-E* [76] outperforms in recall and pose precision our best performing D-M-V combination by UIF from an earlier snapshot in time (*DoG* + *ConvOpt* + *Mutual Nearest Neighbor Matcher* + *OANet* + *RANSAC-E*), which recovers only 3 of 32 and 5 of 20 camera poses on *Skydio-Crane-Mast-32* and *Notre-Dame-20*, respectively. Comparing against a classical SIFT-based front-end combination, we also see SuperGlue's advantages: on *Skydio-Crane-Mast-32*, the SuperGlue-based front-end has substantially lower global translation angular error ($0.66°$ / $13.35°$ in median / mean) than a SIFT-based front-end ($9.93°$ and $26.16°$), with comparable recall (31/32 vs. 32/32 of cameras localized). It is also apparent on *Notre-Dame-20*, where the SuperGlue-based front-end achieves much lower global translation angular error *on average* ($1.45°$ / $3.12°$ in median / mean) vs. a SIFT-based front-end ($1.64°$ and $20.01°$), with comparable recall (19/20 vs. 17/20 of cameras localized).

3.12 Appendix

3.13 Survey of Front-Ends

Deep front-ends for SLAM and SfM can be divided into two main categories: those based on sparse local feature matching, and those based on dense, correspondence-free methods. Methods of the latter category generally estimate either dense depth maps [248, 249, 250, 251, 252, 253] or relative poses (R, t) [254, 255, 256], or both [257, 258, 259, 260, 261]. Although dense, deep front-ends are promising, our focus is the former stream of work.

3.13.1 Survey of Local Feature Detectors

Handcrafted feature detection dates back at least 40 years to Moravec's [184] corner detector and the literature is extensive. Tuytelaars and Mikolajczyk [89] provide an early survey. Although the earliest local features were represented as 2D points [186], oriented circles from DoG [22], ellipses (Harris-Affine), and oriented ellipses (e.g. Harris-Affine with orientation assignment) have been proposed [262]. While, deep methods, rather than classical methods, are a primary focus of this work, we provide a brief summary of classical endeavors towards invariance and equivariance.

Rotational invariance of detectors was an early requirement. Moravec [184] achieved it by comparing a patch with small shifted versions of itself and assessing stability via summed square differences (i.e. the auto-correlation function/surface). Harris [186] achieves rotational invariance by approximating both the auto-correlation surface and eigenanalysis of the auto-correlation matrix using combinations of cheap mixed derivatives I_xI_y, I_x^2, I_y^2 with respect to shifts (x, y). Beaudet's Hessian detector [185] is also rotationally invariant, by using the determinant of the Hessian of image intensity, i.e. $I_{xx}I_{yy} - I_{xy}^2$. Fast approximations of these second derivatives needed for the Hessian were introduced in the Difference-of-Gaussians (DoG) [22] and SURF [32], which used box filters with heavily discretized 2nd order Gaussian derivatives and integral images.

Many of these handcrafted detectors sought keypoints based on simple heuristics, such as strong two-dimensional signal change indicates a corner. Other handcrafted heuristics were also introduced, such as the "SUSAN" principle, stating that if one were to place a circle around a keypoint, the center pixel should be sufficiently different from the intensities along the perimeter of the circle [187].

A transition from hand-crafted to machine learning-based feature detection has been a longstanding goal of the community. Neural networks [188], SVMs [52], decision trees (FAST) [73], or boosting were employed.

Detectors introduced in the recent literature [42, 77, 72, 87, 93, 33, 31] are all based on convolutional neural networks (convnets). We briefly review their supervision paradigms in chronological order below. While the earliest supervision paradigm for convnets was to imitate hand-crafted detectors, per TILDE [90], LIFT [95], and others [27], this approach is no longer seen as state-of-the-art: by imitating handcrafted detectors, one can never learn to outperform them. Homography supervision emerged as the next supervision paradigm; While datasets with homography two-view relations exist, such as OxfordMatching [62] and HP-Sequences [196], these datasets are very limited in size and are better suited for testing. Although homographies naturally arising from pairs of images are desirable for their realistic nature, plentiful training data with significant diversity can be obtained by generating synthetic homography pairs. Specifically, these may be created from any image source by sampling random homographies and applying the warp [42, 72, 31]. A third paradigm was to use synthetically rendered shapes such as triangles and checkerboards where corners are easy to define [42]; unfortunately, models trained on this data are not useful for real-world application unless subsequently fine-tuned. Fourth, epipolar constraints were introduced as supervision. While homography is useful, the 3D world is governed by epipolar constraints. In this paradigm, one uses datasets with Fundamental-matrix two-view relations to create strong negatives and possible positives [93]. Finally, very recent work foregoes domain-specific supervision above by using downstream performance as supervision. While the previous paradigms directly optimize detection evaluation metrics, a superior approach may be to instead optimize downstream matching performance via end-to-end learning. Keypoint selection can be considered an action that generates a reward in a reinforcement-learning paradigm [33, 87]; ground-truth labels, i.e. usefulness for accurate matching, can be determined on the fly.

A trend in architecture design of such networks is to use fully-convolutional networks over entire image input for simultaneous detection and description [42, 72], sharing parameters for both tasks. Earlier approaches required passing all patches of an image through a convnet [77, 90]. Key.Net [31] showed how a FCN could not only differentiably extract multiple maxima over a score map, but also use dramatically fewer parameters than others by operating on image intensity derivatives input.

Equivariant[5] detectors [53, 98, 42, 189, 31] have been a dominant area of research by enforcing consistently high and low responses of a keypoint detector's score map under a geometric transformation of input image [77], or by determining ground-truth locations with a pre-trained detector under equivariance [42].

3.13.2 Survey of Local Feature Descriptors

In the literature, many have designed handcrafted descriptors specifically for photometric, scale, and rotation invariance. The earliest hand-crafted feature descriptors date back 40 years to Moravec [184], a modified version of normalized cross-correlation of patches. Schmid and Mohr [192] were the first to use differential invariants (the "local jet") [263] as keypoint descriptors. Lowe's SIFT descriptor [21, 22] achieved invariance to constant brightness changes by forming a 128-d vector from histograms of gradients in the local keypoint neighborhood. Mahendran and Vedaldi [194] showed how SIFT could be computed as a feedforward pass through a fully-convolutional network. RootSIFT [28] showed how raising ℓ_1-normalized SIFT vectors to the $^1/_2$'th power yields superior performance for descriptor comparison; descriptor values are non-negative magnitudes of histograms by definition. SURF [32] showed how a smaller 64-d descriptor could be formed by using sums of derivatives and their absolute values, without orientation-based histograms. Mikolajczyk and Schmid [62] extended SIFT's descriptor to form histograms over a log-polar grid instead of a Cartesian one, which they name GLOH. A log-polar representation may generate a better local representation by oversampling the immediate neighborhood of the point [193, 44]. DAISY [86] extended the log-polar formulation of GLOH from a single ring around the keypoint to multiple ring centers in a regular pattern around a keypoint. BRIEF [38] aimed to speed up the matching by using binary descriptors instead of real-valued ones, showing how simple intensity comparisons at randomly selected pixel-pairs could approximate local gradients. BRISK [54] replaced BRIEF's random pixel-pair sampling with a deterministic sampling pattern, enforcing uniform density at any given radius around a keypoint. Oriented FAST and Rotated BRIEF (ORB) [75] showed how the pixel-pair sampling locations should be rotated before comparisons are computed.

Incorporating machine learning into feature descriptor methods has been a focus for more than a decade. PCA-SIFT [51] learned a projection matrix from a rotated, scaled, and flattened gradient image around a keypoint to a low-dimensional descriptor such that variance is maximized in the new subspace. Winder and Brown [92, 91] used SfM to create patch training data from collections of tourist photographs; derivative-free Powell optimization [264] could then be used to optimize the parameters of a SIFT- or GLOH-like descriptor algorithm. ORB [75] showed how to use greedy optimization to find less

[5]Equivariance: changes in the input lead to corresponding changes in output.

correlated pixel-pair comparisons in BRIEF. ConvOpt [78] showed how selection of spatial pooling regions and a projection matrix for dimensionality reduction could be learned sequentially as two convex optimization problems.

Recent local feature descriptors algorithms in the literature are all convnet-based and learned from data using metric learning. Spatial proximity high-dimensional Euclidean space should indicate geometric similarity. For a given query descriptor, distances to all other descriptors should be ranked according to geometric similarity. While most early methods optimized an upper bound on the true ranking (known as the Essential loss [265]) by the ranking of just two or three descriptors at a time, many recent works optimize over the ranking of the entire list at once [49, 266, 72].

Patch input-based convnet architectures initially were predominant. Jahrer *et al.* [50] use the two-tower (Siamese) architecture [267, 268, 269] to learn a metric space for local feature descriptors. Seven years later, MatchNet [47] and DeepCompare [96] also adopted a Siamese architecture, but rather than learning explicit descriptors, treated the matching of local patch pairs as learning the binary classification decision boundary between matching and non-matching pairs, producing a similarity score between two patches. DeepCompare [96] showed how to introduce multi-resolution patch input to the Siamese architecture. DeepDesc [79] showed how a Siamese architecture could be trained with a contrastive loss. Triplet Feature Network (TFeat) [30] explored the use of a triplet loss, and HardNet [64] and L2-Net [84] formulate novel losses for patch metric learning. LIFT [95] trained differentiable orientation and keypoint selection modules to precede the DeepDesc descriptor, but learned them separately. LF-Net [68] was the first jointly learned detection and description method. Log-Polar [44] incorporates classical log-polar patch formulations [193, 44, 86] into a deep network.

Most recent descriptor methods no longer accept patch input, but rather process an entire image as input, reusing activations for overlapping regions and increasing the receptive field. UCN [40] showed how to generate dense descriptor embeddings fully-convolutionally and apply the contrastive loss to an entire image simultaneously. SuperPoint [42] showed how the contrastive loss could be computed with cosine-similarity instead of Euclidean-distance-based similarity. IMIPS [39] ... D2-Net [43] pushed joint-detection and description to the extreme with a single loss function that combines the triplet loss and a repeatability loss acting on different axes of a 3d tensor of embeddings; metric space margin violations are weighted by their detection confidence scores, such that low detection weight is assigned to keypoints involved in incorrect correspondences. R2D2 [72] showed how dilated convolution could be used instead of strided convolution to create fully-dense, non-subsampled pixel embeddings.

3.13.3 Survey of Outlier Rejection Algorithms

Classical approaches to putative 2d correspondence verification involve robust estimation of epipolar geometry using *RANdom SAmple Consensus* (RANSAC) [169]. Like RANSAC, MLESAC [201] samples putative solutions, but maximizes the likelihood of the solution rather than just the number of inliers. LMedS [200] provides robustness to outliers by minimizing the median of squared residuals instead of their sum.

Several have now shown that robust estimation can now be performed by using deep learning. D-SAC [35] extended RANSAC to a differentiable variant by employing policy gradients [270] in order to backpropagate through hard decisions (choosing a maximum over hypotheses). Recent approaches to correspondence outlier filtering instead rely upon estimating inlier probabilities per each putative correspondence using deep nets. Given that putative correspondences represent an unordered set, most approaches to modern deep-learning correspondence verification use the permutation-invariant PointNet architecture [271].

The earliest such architectures [94, 70] used deep nets to estimate inlier probabilities per correspondence, establish a weighted homogeneous least squares problem [272, 175], and then employed the differentiability of eigendecomposition or SVD to differentiably estimate the essential or fundamental matrix, respectively. While [94] required curriculum learning (adding a cross-entropy loss early training), Dang *et al.* [41] showed how one need not produce an Essential matrix at training time, but rather only the inlier probabilities; a loss function of the ground truth eigenvector (essential matrix) and predicted weights applied to the data matrix can perform a suitable optimization.

Several have extended these early architectures to use nearest-neighbor context and effective normalization [202]. Neighbors-Mining Network (NM-Net) [102] utilizes local affine structure from the Hessian-Affine detector [61] as side information to identify suitable neighbors. However, NM-Net discards the epipolar-geometric aspect of the problem entirely, treating outlier rejection only as binary classification. Neural Nearest Neighbor Networks (N^3 Net) [69] adds an additional differentiable k-nearest neighbor layer. Order-Aware Net [97] adds additional local context for each point.

NG-RANSAC [36] also learns a PointNet architecture, but trains it using policy gradients such that it parameterizes an inlier probability distribution over all putative correspondences. Rather than using these inlier probabilities in a weighted least squares problem, they exploit the distribution for biased sampling of high-probability hypotheses under a limited budget. Unlike all others, SuperGlue [76] poses the problem as a optimal partial assignment problem between two sets of local features, wherein a graph neural network predicts the cost function of the differentiable assignment optimization.

3.14 Benchmark Evaluation Details

In this section, we provide more details about the evaluation of the **D**, **DM**, and **DMV** stages.

3.14.1 Feature Detection Evaluation

The repeatability rate is the percentage of the total observed keypoints that are detected in both images, and was first formally defined by [190]. In order to measure the detector's repeatability, i.e. ability to repeatedly fire on the same 2d structures in two view, we use keypoint distance-based repeatability, i.e. the number of corresponding keypoints divided by the number of covisible keypoints. While several works [61, 63, 77, 191] have instead used an ellipse-overlap based repeatability metric, we prefer keypoint-distance-based repeatability [42, 273, 87] for a number of reasons. While ellipse-overlap accounts for the scale of a feature, most deep networks are fully convolutional and no longer estimate patch-

or ellipse-based regions. In addition, with ellipses, if we set the scale to be larger, we artificially increase the ellipse IoU. Finally, the calculation of the closed form overlap area of two ellipses is computationally demanding [274].

Lenc and Vedaldi [53, 191] note that with an increased number of features, it becomes easier to match features by accident. Accordingly, repeatability may be made arbitrarily large simply by detecting enough features, making repeatability biased for settings that produce more features, as R2D2 demonstrates empirically [72]. Thus, just as they do, we compute repeatability and matching score as the feature detection threshold is increased. We compute repeatability with keypoint cardinality cutoffs of 150,300, 600, 1200, 2400 keypoints, in accordance with Quad-Net [77] and R2D2 [72]. If the method cannot produce 3000 keypoints (e.g. IMIPS can only support 128), we use the maximum amount the method can produce at higher thresholds (analogous to a budget) for every single threshold ≥ 128. We sort the keypoints by score, and like R2D2 [72], find that repeatability increases with more keypoints.

3.14.2 Detection, Description, and Matching (DM) Benchmark Evaluation

The **M** stage produces *putative correspondences*: correspondences commonly accepted or hypothesized to be correct, but without final verification. The output of a user's algorithm should be verified (potentially geometrically) correspondences which can be presented for evaluation.

Suitable Matching Schemes

For the **M** Stage, we seek to preserve the fidelity of the original authors' implementation, such that the descriptor and matching algorithm they prescribe is used in our benchmark. In practice, we see three matching regimes employed: *one-way nearest neighbor matching with ratio test* [22, 62, 25], *mutual nearest neighbor matching* [43, 275, 276], *greedy matching*. However, in order to *fairly* evaluate the intermediate stage and in keeping with Heinly *et al.*, for those methods that use 1-way NN matching, we convert the matching to a greedy NN matching scheme, such that a 1:1 constraint for putative matches is met; Accordingly, a cluster of keypoints in a 3 pixel radius in the source image I_a cannot all count as true positive matches with a single keypoint in the target image I_b.

While earlier benchmarks have not addressed the impact of matching algorithms on the fairness of evaluation metrics, we believe this aspect should not be overlooked (see Figure 3.18. Like Heinly *et al.* [195], we desire a 1:1 constraint for putative matches: a cluster of keypoints in a 3 pixel radius in the source image I_a should not all count as true positive matches with a single keypoint in the target image I_b. There are two ways to achieve this: mutual nearest-neighbor constraints (which not all methods use), or a greedy matching strategy. For all methods that use mutual nearest-neighbor (NN) matching, we proceed with their implementation; for others, such as SIFT that allow multiple matches per keypoint, we convert their matching to a greedy NN matching scheme. We briefly review how each matching algorithm operates on computed distance matrix $\in \mathbb{R}^{m_a \times m_b}$ between keypoints in I_a and keypoints in I_b:

One-way nearest neighbor matching [22, 62, 25]: Does not provide 1:1 putative matches. Proceed along row dimension, and select the argmin of each row. A single column may be selected for multiple rows. An optional ratio test [22] is performed afterwards.

Mutual nearest neighbor matching [43, 275, 276]: Provides 1:1 putative matches (when points are unique). Traverse each row i. The argmin j of row i is selected, and then the argmin $\hat{i}$ is selected over the entire column j. If $i = \hat{i}$, then the row and column are mutual nearest neighbors, and a match is made.

Greedy matching: Provides 1:1 putative matches. Iteratively choose the lowest (i, j) entry as a match, and remove row i and column j from consideration, and proceed, until no entries remain.

Matching Evaluation Metrics

For **homography-based datasets** such as HP-Sequences [196], Matching-Score (M-Score) and MMA have become the de facto gold standard for evaluating detector-descriptor pairs [43, 31, 95]. However, M-Score suffers from a large number of problems, as we will illustrate. M-Score was originally defined by [63] as the ratio between the number of correct matches and the smaller number of detected features in the pair of images. Heinly *et al.* [195] modified the denominator to include only the covisible number of features and provided a decomposition of M-Score into putative match ratio (PMR) and inlier ratio. PMR is the quantity of features useful for generating matches:

$$PMR = \frac{\text{\#Putative Matches}}{\text{\#Covisible Features Proposed}} \tag{3.16}$$

and Inlier Ratio measures the precision of matching and the descriptor's discriminative ability as

$$InlierRatio = \frac{\text{\#Correct Matches}}{\text{\#Putative Matches}} \tag{3.17}$$

Inlier Ratio is computed using the standard precision formula $^{TP}/_{(TP+FP)}$, where a true positive match has small deviation from pixel-pixel mapping (homography) [62] or pixel-to-line mapping (epipolar geometry) [40].

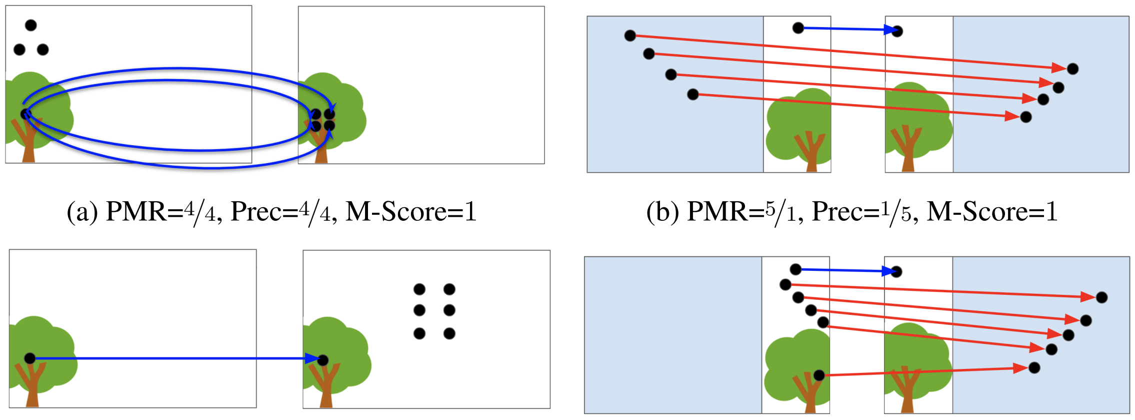
The product of the two quantities above constitutes the matching score:

$$\text{M-Score} = \frac{\text{\#Correct Matches}}{\text{\#Covisible Features Proposed}} = PMR \cdot InlierRatio \tag{3.18}$$

In short, this is the ratio of ground-truth correspondences that can be recovered by the two detection-description stages, over the number of features proposed by the detector in the *shared viewpoint region*, per LIFT [95].

Matching-Score (M-Score) The shared viewpoint constraint is intuitive: a tiny amount of the image is mutually visible (co-visible), the huge number of features in both images (but few shared) shouldn't hurt performance. When considering two views of the same scene, with an identical viewpoint (covisibility of all points in the scene is 100%), and when equal number of keypoints are detected in both images, this metric works as expected. In reality, such a scenario arises rarely.

Unfortunately, this metric clearly does not address the case when a detector proposes



(a) PMR=${}^{4}/_{4}$, Prec=${}^{4}/_{4}$, M-Score=1

(b) PMR=${}^{5}/_{1}$, Prec=${}^{1}/_{5}$, M-Score=1

(c) PMR=${}^{1}/_{7}$ or ${}^{1}/_{1}$, Prec=1, M-Score=${}^{1}/_{7}$ or ${}^{1}/_{1}$

(d) PMR=${}^{6}/_{1}$ or ${}^{6}/_{6}$, Prec=${}^{1}/_{6}$, M-Score=1 or ${}^{1}/_{6}$

Figure 3.18: Different scenarios where the *Matching-Score* achieves 100%, but clearly the feature matching is of poor quality. (a) w/o 1:1 matching constraint, with same viewpoint, same # covisible keypoints. (b) w/ 1:1 matching constraint, different viewpoint, same # covisible keypoints. (c) w/ 1:1 matching constraint, same viewpoint, differing # keypoints. (d) w/ 1:1 matching constraint, different viewpoint, differing # keypoints. In (a-b), we have an equal number of ground truth correspondences and keypoints points in the shared viewpoint region, yielding a Matching-Score of 100%. (b) illustrates that M-Score is not an ideal metric. In (c-d) the Matching-Score is simply undefined, if a minimum of # covisible keypoints is chosen, the M-Score would also achieve 100%.

different number of features in a shared region (See Figure 3.18). Even if a standard number of keypoint detection are thresholded in every image, we have no guarantees about the amount in any covisible region. When m_a keypoints are detected in the shared viewpoint region in image I_a, and m_b keypoints in the shared region of image I_b, it becomes unclear which should serve as the denominator. This arises from an ambiguity if PMR is computed from $I_a \rightarrow I_b$, or $I_b \rightarrow I_a$, which is not defined. Ideally, the metric should be symmetric to order. Three such symmetric functions are possible for the denominator – minimum, average,or maximum, measuring best-case, average-case, and worst-case scenarios. We compute the latter two metrics.

Matching Coverage Fraction (MCovFrac) As discussed in the main paper, MCovFrac Measures the coverage of an image by correctly matched key points. A coverage mask is generated from true positive keypoints, each one adding a disk of fixed radius (25px) [26, 87]. However, we slightly modify the original definition – instead of measuring MCovFrac as a fraction of the image covered by the coverage mask, we evaluate only over the *co-visible region*. Although this amounts to multiplication of the prior metric only by a constant, it modifies the score such that a dense distribution of correct feature matches provides a matching coverage fraction of exactly 1. Thus, the range is resolved to $[0, 1]$, as desired.

DM Pairings Several subtle challenges arise when testing detector/descriptor pairings. First, when combining scale variant detectors (e.g. GLAMPoints, SuperPoint) with scale-invariant descriptors (SURF, SIFT), it is not obvious which scale to provide, as these deep detectors estimate no such scale. Heinly *et al.* [195] noted this issue in their own benchmark: combining a scale invariant descriptor with a detector that was not scale invariant, and combining detectors and descriptors that are both scale invariant. In both such cases, [195] discarded the scale information, and computed the descriptor at the native image resolution. We use a default scale of 2. Second, it is not always possible to preserve the fidelity of the author's original method; any method could be designed for a particular input image size. In order to provide fair timing comparisons, we disregard the intended image size and evaluate all methods at a common resolution.

3.14.3 Additional DMV Evaluation Details

Several constraints exists when selecting a suitable dataset to measure the entire system's performance. As discussed in the main paper, we use YFCC-100M [208, 207]. Homography-based datasets are not suitable for the verification benchmark for a number of reasons. First, their scenes generally do not have frames from wide baselines; while planar scenes can be taken from wide baselines, camera rotations about a fixed camera center or static webcams (a predominant fixture of HP-Sequences) would have a very narrow baseline. Second, rather than involving epipolar geometry estimation, the model fitting step would involve homography estimation, a task which the majority of deep verification methods do not perform and do not provide pre-trained models for. Indeed, deep homography prediction models such as MagicWarp [212, 213] are the rare exception for verification and their pre-trained models are not publicly available. Third, these homographies are not emblematic of the real 3D world. Thus, while HP-Sequences [196] is our preferred dataset for the Detection-Description tasks, we exclude it from our verification task. In addition, HP-Sequences does not provide camera intrinsic matrices, which prevents us from normalizing the image coordinates which several deep methods require [94, 69].

We exclude several verifiers from our benchmark, as they cannot be used in an "all-purpose" fashion. For example, pre-trained models publicly available for [70] require the use of Difference of Gaussians (DoG) keypoint geometry as side information. NM-Net [102] requires Hessian-Affine keypoint geometry as side information. Accordingly, we cannot pair them with arbitrary detection-descriptor pairs, and we exclude them from our benchmark. However, EigFree [41], OA-Net [97], LearnedCorr [94], N^3 Net [69] , and NG-RANSAC [36] can be run in a general-purpose fashion, even if trained with input from a specific detector-descriptor pairing. RANSAC's [169] fully general nature can be considered a strength in our benchmark framework.

3.15 Implementation Details

In this section, we provide information regarding the origin of source code for each method's implementation.

Implementations of the following classical detectors and/or descriptors were mainly obtained from OpenCV 4.2, compiled from source:

- BRIEF [38] (OpenCV)
- BRISK [54] (OpenCV)

- ConvOpt [78] (OpenCV)
- DoG [22] (OpenCV)
- FAST [73] (OpenCV)
- Harris-Laplace (OpenCV)
- KAZE [25] (OpenCV)
- MSER (OpenCV)
- ORB [75] (OpenCV)
- PCA-SIFT [51]
  https://github.com/ahojnnes/local-feature-evaluation
- SIFT [21, 22] (OpenCV)
- SURF [32] (OpenCV)
- SIFT [21, 22] (VLFeat)

Implementations of the following deep convnet-based detectors and descriptors were obtained from the original authors:

1. ContextDesc [57] https://github.com/lzx551402/contextdesc
2. D2-Net [43] https://github.com/mihaidusmanu/d2-net
3. CovDet/DDet [53] https://github.com/lenck/ddet
4. DeepDesc [79] https://github.com/etrulls/deepdesc-release
5. GLAMPoints [87] https://gitlab.com/retinai_sandro/glampoints
6. IMIPS [39] https://github.com/uzh-rpg/imips_open
7. Key.Net [31] https://github.com/axelBarroso/Key.Net
8. LIFT (Tensorflow) [95] https://github.com/cvlab-epfl/tf-lift
9. LF-Net [68] https://github.com/vcg-uvic/lf-net-release
10. OpenUCN [40, 197] https://github.com/chrischoy/open-ucn
11. R2D2 [72] https://github.com/naver/r2d2
12. Spreadout HardNet [99] https://github.com/ColumbiaDVMM/hardnet
13. SuperPoint [42]
    https://github.com/MagicLeapResearch/SuperPointPretrainedNetwork

14. TFeat [30] https://github.com/vbalnt/tfeat

15. TransformCovariant [98]
https://github.com/ColumbiaDVMM/Transform_Covariant_Detector

The implementations of HarrisNet and SIFTNet are described in subsection 3.15.1 and subsection 3.15.2 of this appendix.

The implementations of the following putative correspondence verification algorithms were obtained from the following sources:

- Eig-Free [41] https://github.com/Dangzheng/Eig-Free-release
- Learned-Corr [94]
https://github.com/vcg-uvic/learned-correspondence-release
- LMEDS [200] (OpenCV)
- MLESAC [201]
https://github.com/vcg-uvic/learned-correspondence-release/blob/master/tests.py
- NG-RANSAC [36] https://github.com/vislearn/ngransac
- N^3 Net [69] https://github.com/visinf/n3net/
- Order-Aware Net [97] https://github.com/zjhthu/OANet
- RANSAC [169] (OpenCV)

3.15.1 HarrisNet Implementation Details

While modern convnet-based feature detectors may seem a far step away from the Harris Corner detector, Harris [186] can be implemented as a convnet, which we entitle *HarrisNet*. Harris provides information about the SSD curvature with determinant, trace, and mixed derivatives (elementwise multiplication of channels), which a deep network could learn to approximate with convolutions, non-linearities, and pooling over many layers; However, such heavy computation could be considered wasteful when simple, closed-form expressions exist and can be modeled with shallow architectures.

We implement HarrisNet, a shallow 5-layer architecture, with batch size N input:

1. *Image Gradient Layer*: Compute image intensity gradient $[I_x, I_y]^T$ using 3×3 Sobel filters. With this convolution, we expand the grayscale image to a 2-channel feature map, $\mathbb{R}^{N \times 1 \times H \times W} \rightarrow \mathbb{R}^{N \times 2 \times H \times W}$.

2. *Channel Product Layer*, returns the three products I_x^2, I_y^2 and $I_x I_y$ between the two channels I_x, I_y of the previous layer. Lifts the feature map to three channels $\mathbb{R}^{N \times 2 \times H \times W} \rightarrow \mathbb{R}^{N \times 3 \times H \times W}$.

3. *Second Moment Matrix Layer* Convolution with a Gaussian kernel returns channels containing the three values required for the Second Moment Matrix at each pixel:

$$\begin{aligned} S_{xx} &= G_k(\sigma) * I_x^2, \\ S_{yy} &= G_k(\sigma) * I_y^2, \\ S_{xy} &= G_k(\sigma) * I_x I_y \end{aligned} \tag{3.19}$$

   where $*$ represents convolution, and $G_k(\sigma) \in \mathbb{R}^{k \times k}$ represents a Gaussian kernel. The output dimension is unchanged from the input dimension $\mathbb{R}^{N \times 3 \times H \times W}$

4. *Corner Response Layer* computes the corner response map R over the entire image,

$$R = \text{det}(M) - \alpha(\text{tr}(M))^2 \quad M = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{bmatrix} \tag{3.20}$$

   converting $\mathbb{R}^{N \times 3 \times H \times W} \rightarrow \mathbb{R}^{N \times 1 \times H \times W}$

5. *NMS Layer* - performs non-maximum suppression to keep only the strongest corners in local regions. $\mathbb{R}^{N \times 1 \times H \times W}$. We utilize max pooling with a 7×7 kernel. This will fill every entry in the subgrids with the maximum nearby value. By binarizing the image according to locations that are equal to their maximum, and and multiplying the binary image with the cornerness response values, we can achieve NMS.

After NMS, the top K 2-d keypoint locations $\{\mathbf{x}_k\}_{k=1}^{K}$ are returned, ranked by corner response.

3.15.2 SIFTNet Implementation Details

Convnet patch-embeddings are now considered more powerful descriptors than SIFT, but a rotation-variant, scale-variant (without trilinear histogram interpolation) SIFT can be implemented as a single feedforward pass through a 5-layer fully convolutional convnet. We extend several layers of Densely-Computed SIFT (DSIFT) [194] to a network which we entitle *SIFTNet*. These layers are implemented as follows, producing feature map tensors with indicated sizes:

1. *Image Gradient Layer*: Compute image intensity gradient $[I_x, I_y]^T$ using 3×3 Sobel filters. With this convolution, we expand the grayscale image to a 2-channel feature map, $\mathbb{R}^{N \times 1 \times H \times W} \rightarrow \mathbb{R}^{N \times 2 \times H \times W}$.

2. *Orientation Projection Layer*: Geometrically compute gradient response within each of 8 orientation bins. Rather than decoding the gradient orientation at each pixel with $\tan^{-1}(I_y/I_x)$, we project each gradient vector ∇I onto eight 2-d orientation basis vectors $\mathbf{v}_i$ around the unit circle, corresponding to angles $\theta_i \in \{\pi/8, 3\pi/8, \dots, 13\pi/8, 15\pi/8\}$, i.e. gradients in $[0, \pi/4)$ have greatest projection onto the first orientation vector. In order to later weight the best-aligned orientation vector with the gradient's norm, we also create two identity filters that copy over I_x, I_y to the next layer. With this 1×1 convolution, we lift the feature map from $\mathbb{R}^{N \times 2 \times H \times W} \rightarrow \mathbb{R}^{N \times 10 \times H \times W}$.

3. *Parameter-Free Histogram Layer*: Compute the weighted histogram using a non-linearity. Computing a histogram requires knowing which basis bin to increment; although an `argmax` along the channel dimension is one non-differentiable way of doing so, a differentiable alternative is a softmax. Along the first 8 channel dimensions, at each pixel we have a vector of cosine similarity values $\mathbf{c}$ such that $\mathbf{c}_i = \mathbf{v}_i \cdot \nabla I = \cos(\theta_{\mathbf{v}_i, \nabla I}) \|\mathbf{v}_i\| \|\nabla I\|$, and since $\|\mathbf{v}_i\| = 1$, we seek the largest value $\mathbf{c}_i = \cos(\theta_{\mathbf{v}_i, \nabla I}) \|\nabla I\|$ along the channel dimension at each pixel. The per-pixel histogram $h \in \mathbb{R}^8$ can be computed as

$$h = \|\nabla I\| \cdot \text{softmax}\Big(\max\big\{0, \mathbf{c} - \cos(\pi/8) \|\nabla I\|\big\}^{1/\tau} \Big) \tag{3.21}$$

If the gradient vector falls within a basis vector $\mathbf{v})i$'s bin, it must lie within $\pi/8$ on either side. Subtraction by $\cos(\pi/8)\|\nabla I\|$ from all cosine similarities ensures that if any value falls below $\cos(\pi/8)$, it could not lie within $\mathbf{v}_i$'s bin with arc $\pi/4$. By applying the ReLU, all elements outside of the appropriate bin will be clamped to zero, and the only entry left will have a positive value. Such a positive value can be driven back up to 1 by a low-temperature softmax, effectively applying hard clamping to 0/1 values. A vectorized implementation will elementwise multiply a 4d binary occupancy tensor, one-hot at the appropriate orientation bin, with the per-pixel gradient magnitude, increment its histogram bin by a certain weight. This layer returns a per-pixel 8-d histogram $\mathbb{R}^{N \times 10 \times H \times W} \rightarrow \mathbb{R}^{N \times 8 \times H \times W}$.

4. *Sub-grid Accumulation Layer*: Accumulate per-pixel histograms among 4×4 sub-grids by group convolution (with 8 groups), meaning convolution with a 4×4 kernel, filled with unit weights, to sum 8-d vectors along the height,width dimensions. Dimensions are preserved $\mathbb{R}^{N \times 8 \times H \times W} \rightarrow \mathbb{R}^{N \times 8 \times H \times W}$.

5. *Feature Stacking Layer*: If every pixel represents a keypoint, we must pull out the 8-d histogram at the center of each 4×4 subgrid (representing subgrid summary statistic), and stack them. This can be accomplished as à trous (dilated) convolution with 128 filter banks, each with a different one-hot single cell inside a 8-channel, 4×4 kernel. We convert the feature map from $\mathbb{R}^{N \times 8 \times H \times W} \rightarrow \mathbb{R}^{N \times 128 \times H \times W}$.

6. *ℓ_2-normalization* Implemented by a standard local response normalisation layer to bring each pixelwise descriptor to unit length [194].

After passing images through the network, we produce densely populated 128-d feature vectors at each pixel. Postprocessing is limited to raising each feature vector to power-law normalization, i.e. raising each descriptor a power less than 1, as in RootSIFT [28].

3.16 HPSequences Qualitative Results

3.17 Tables of Results

In the earlier sections of this chapter, we provided only visual representations on results on five main benchmarks. We now provide tables of quantitative results on the h**D** (Table 3.6,

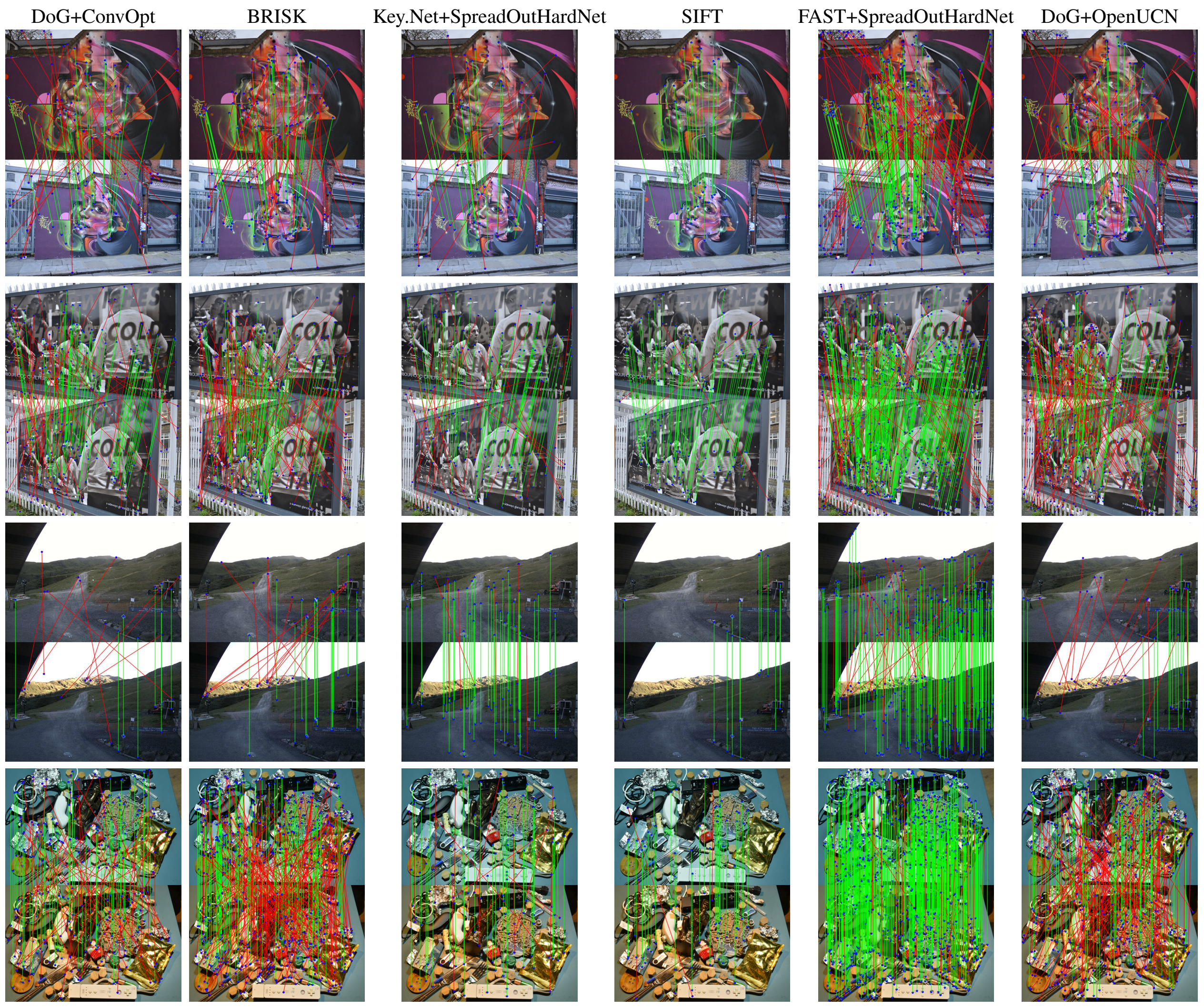


Figure 3.19: Performance of six **DM** systems on HPSequences. Rows 1-2: *Woman*, Rows 3-4: *Coffee House*, Rows 5-6: *Autannes*, Rows 7-8: *Objects*. The keypoints are randomly subsampled by 8.

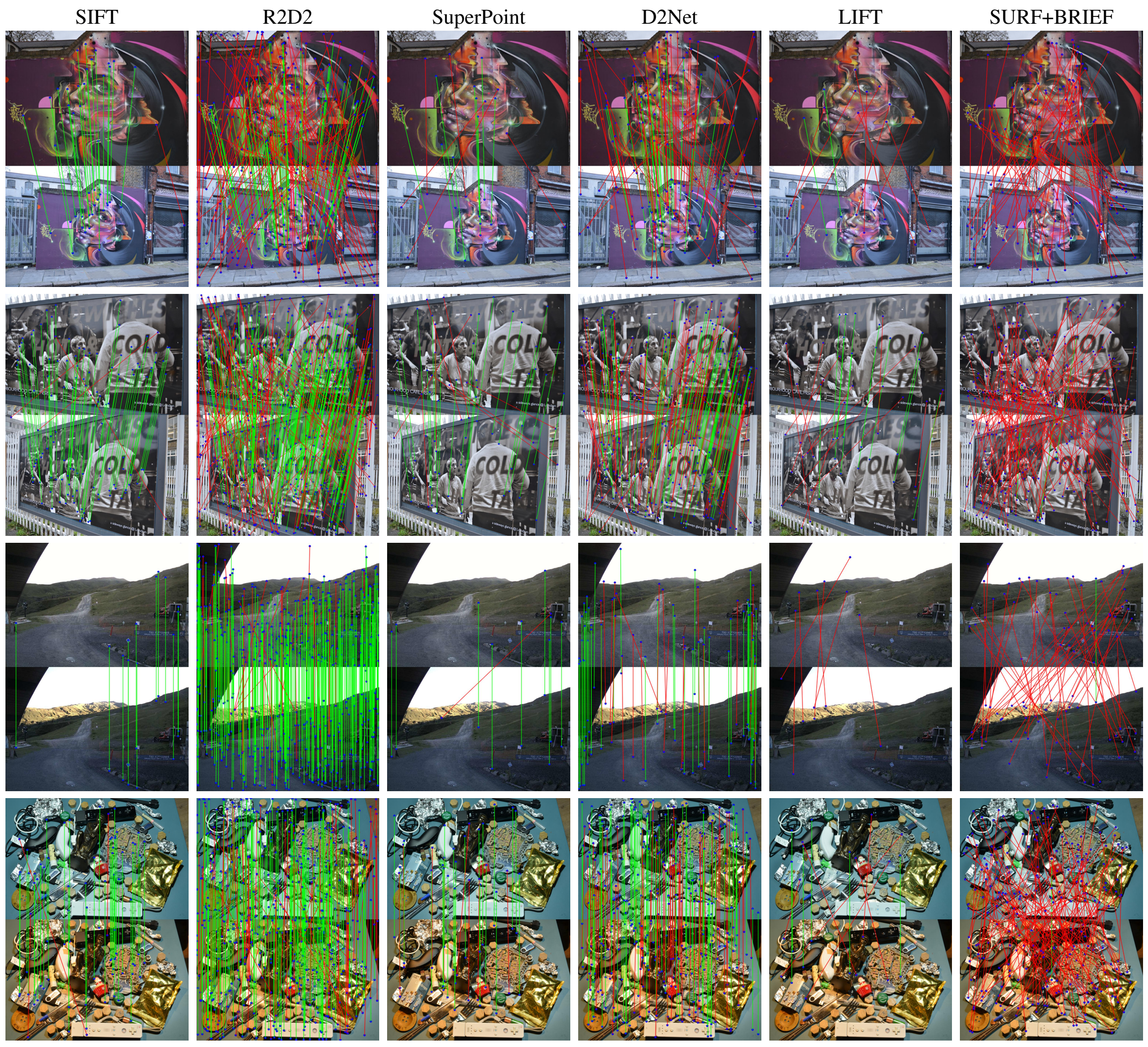


Figure 3.20: Performance of six **DM** systems on HPSequences. Rows 1-2: *Woman*, Rows 3-4: *Coffee House*, Rows 5-6: *Autannes*, Rows 7-8: *Objects*. The keypoints are randomly subsampled by 8.

Detector	**Rep @150**	**Rep @300**	**Rep @600**	**Rep @1200**	**Rep @2400**
HARRIS	0.54	0.58	0.58	0.59	0.59
HARRISNET	0.55	0.55	0.55	0.56	0.55
KEY.NET	0.54	0.55	0.56	0.56	0.56
FAST	0.52	0.57	**0.60**	**0.63**	**0.64**
KAZE	0.49	0.51	0.51	0.52	0.52
BRISK	0.49	0.53	0.55	0.56	0.57
SUPERPOINT	**0.57**	**0.59**	0.59	0.59	0.59
ORB	0.48	0.52	0.52	0.52	0.52
GLAMPOINTS	0.51	0.52	0.53	0.54	0.54
SURF	0.48	0.49	0.50	0.50	0.50
DOG	0.40	0.41	0.42	0.43	0.43
BRIEF	0.42	0.42	0.42	0.42	0.42
CONTEXTDESC	0.37	0.38	0.39	0.40	0.40
LF-NET	0.17	0.29	0.40	0.45	0.46
MSER	0.55	0.56	0.56	0.57	0.58
D2-NET	0.38	0.39	0.40	0.41	0.41
LIFT	0.33	0.34	0.35	0.35	0.35
HARRIS-LAPLACE	0.42	0.45	0.46	0.47	0.47
R2D2	0.20	0.29	0.39	0.48	0.52

Table 3.6: Results of h**D** - illumination variant scenes (HP-Sequences) as a function of the number of keypoints (150, 300, 600, 1200, 2400). The results are sorted by the repeatability @ 150 keypoints on the combined viewpoint variant and illumination variant scenes. For those methods that can only produce a limited number of keypoints, we consider the cutoff threshold a budget, and copy those numbers to the rightmost columns. (Corresponds to Figure 3.3 of section 3.3).

Table 3.7), w**D** (Table 3.8), h**DM** (Table 3.9,Table 3.10), w**DM** (Table 3.11), and w**DMV** (Table 3.12). The full tables are too large to display here, but are available in CSV format at https://github.com/deep-front-ends/deep-front-ends.

Detector	**Rep @150**	**Rep @300**	**Rep @600**	**Rep @1200**	**Rep @2400**
HARRIS	**0.58**	**0.61**	**0.63**	0.62	0.60
HARRISNET	0.55	0.56	0.54	0.54	0.54
KEY.NET	0.53	0.54	0.53	0.53	0.53
FAST	0.53	0.57	0.60	**0.63**	**0.65**
KAZE	0.48	0.51	0.54	0.55	0.55
BRISK	0.48	0.51	0.54	0.56	0.57
SUPERPOINT	0.38	0.41	0.42	0.42	0.42
ORB	0.45	0.48	0.48	0.48	0.48
GLAMPOINTS	0.39	0.43	0.43	0.43	0.43
SURF	0.42	0.44	0.45	0.45	0.45
DOG	0.43	0.45	0.47	0.47	0.48
BRIEF	0.35	0.34	0.34	0.34	0.34
CONTEXTDESC	0.36	0.40	0.44	0.46	0.46
LF-NET	0.17	0.29	0.40	0.45	0.46
MSER	0.33	0.35	0.36	0.36	0.37
D2-NET	0.22	0.24	0.28	0.30	0.30
LIFT	0.16	0.17	0.17	0.17	0.17
HARRIS-LAPLACE	0.03	0.05	0.08	0.09	0.10
R2D2	0.09	0.15	0.26	0.39	0.51

Table 3.7: Results of h**D** - viewpoint variant scenes (HP-Sequences) as a function of the number of keypoints (150, 300, 600, 1200, 2400) (Figure 3.3(b)). The results are sorted by the repeatability @ 150 keypoints on the combined viewpoint variant and illumination variant scenes. For those methods that can only produce a limited number of keypoints, we consider the cutoff threshold a budget, and copy those numbers to the rightmost columns. (Corresponds to Figure 3.3(b) of section 3.3).

Detector	Rep @150	Rep @300	Rep @600	Rep @1200	Rep @2400
HarrisNet	**0.61**	**0.68**	0.68	0.68	0.68
SuperPoint	0.59	0.59	0.59	0.59	0.59
Harris	0.58	0.65	**0.72**	**0.73**	**0.73**
ContextDesc	0.57	0.64	0.68	0.68	0.68
Key.Net	0.57	0.64	0.64	0.64	0.64
SURF	0.57	0.64	0.69	0.69	0.69
D2-Net	0.57	0.63	0.68	0.68	0.68
LIFT	0.56	0.56	0.56	0.56	0.56
GLAMPoints	0.55	0.55	0.55	0.55	0.55
FAST	0.54	0.61	0.66	0.70	0.72
KAZE	0.53	0.59	0.60	0.60	0.60
BRISK	0.52	0.60	0.64	0.66	0.66
LF-Net	0.52	0.58	0.65	0.66	0.66
DoG	0.51	0.60	0.66	0.66	0.66
R2D2	0.49	0.57	0.63	0.68	0.68
BRIEF	0.47	0.47	0.47	0.47	0.47
ORB	0.43	0.51	0.51	0.51	0.51
Harris-Laplace	0.34	0.41	0.44	0.44	0.44
MSER	0.33	0.35	0.35	0.35	0.35

Table 3.8: Results of w**D** (YFCC-100M) as a function of the number of keypoints (150, 300, 600, 1200, 2400) (Figure 3.3(c)). For those methods that can only produce a limited number of keypoints, we consider the cutoff threshold a budget, and copy those numbers to the rightmost columns. (Corresponds to Figure 3.3(c) of section 3.3).

Detector+Descriptor	**UIF-UB**	**PMR**	**Inlier Ratio**	**M-Coverage Fraction**	**M-Score**
FAST+OPENUCN	0.99	1.00	0.32	0.70	0.27
FAST+SPREADOUT-HARDNET	0.98	0.66	0.69	0.72	0.41
CONTEXTDESC	0.98	0.65	0.58	0.62	0.33
HARRIS+SPREADOUT HARDNET	0.97	0.60	0.52	0.75	0.34
SURF+SPREADOUT HARDNET	0.97	0.61	0.57	0.65	0.32
FAST+CONVOPT	0.97	0.64	0.63	0.69	0.37
R2D2	0.97	0.59	0.57	0.78	0.36
DOG+OPENUCN	0.97	1.00	0.37	0.60	0.31
HARRISNET+SPREADOUT-HARDNET	0.96	0.66	0.57	0.67	0.38
HARRIS+OPENUCN	0.96	1.00	0.22	0.73	0.22
⋮					
GLAMPOINTS+SIFTNET	0.80	0.62	0.45	0.53	0.30
SUPERPOINT	0.80	0.75	0.61	0.46	0.43
GLAMPOINTS+SPREADOUT-HARDNET	0.79	1.00	0.28	0.52	0.28
MSER+OPENUCN	0.79	1.00	0.33	0.27	0.26
DOG+SIFT	0.79	0.21	0.71	0.43	0.15
⋮					
MSER+SURF	0.21	0.10	0.47	0.10	0.04
MSER+TFeat	0.10	0.09	0.36	0.08	0.03
KAZE	0.00	0.36	0.01	0.02	0.00

Table 3.9: Results of the h**DM** (HP-Sequences) benchmark (corresponding to Figure 3.4(a) of section 3.4), sorted by the Usable Image Fraction-Upper Bound for Homography estimation. We show only the first 10 methods, and 8 representative baselines, from 101 methods.

Detector+Descriptor	**Inlier Ratio (All)**	**Inlier Ratio (V)**	**Inlier Ratio (I)**
SIFT	0.85	0.84	0.85
GLAMPOINTS+TFEAT	0.75	0.69	0.81
GLAMPOINTS+SIFT	0.75	0.68	0.82
HARRISNET+TFEAT	0.75	0.70	0.80
GLAMPOINTS+PCA-SIFT	0.75	0.67	0.82
FAST+SIFT	0.72	0.61	0.84
FAST+PCA-SIFT	0.72	0.62	0.83
HARRIS+TFEAT	0.72	0.67	0.78
HARRISNET+SIFT	0.71	0.59	0.84
DOG+SIFT	0.71	0.67	0.75
DOG+PCA-SIFT	0.71	0.66	0.75

Table 3.10: Results of the h**DM** (HP-Sequences) benchmark (corresponding to Figure 3.4 (b) of section 3.4), sorted by the inlier ratio over all the scenes. *V* and *I* denote the viewpoint and illumination variant scenes from HP-Sequences.

Detector+Descriptor	**UIF-UB**	**PMR**	**Inlier Ratio**	**Rank (UIF-UB)**	**Rank (Inlier Ratio)**
FAST+OPENUCN	1.00	1.00	0.16	**1**	64
CONTEXTDESC	0.98	0.58	0.38	2	20
DOG+OPENUCN	0.97	1.00	0.22	3	48
FAST+SIFTNET	0.97	0.57	0.27	4	38
FAST+SPREADOUTHARDNET	0.96	0.55	0.44	5	12
FAST+CONVOPT	0.96	0.52	0.36	6	24
FAST+ROOTSIFT	0.95	0.46	0.41	7	15
SURF+OPENUCN	0.95	1.00	0.16	8	61
HARRIS+OPENUCN	0.93	1.00	0.10	9	76
BRISK	0.92	0.65	0.18	10	58
R2D2	0.92	0.48	0.32	11	29
HARRIS+SPREADOUTHARDNET	0.92	0.51	0.30	12	31
SURF+SPREADOUTHARDNET	0.91	0.52	0.37	13	22
DOG+SPREADOUTHARDNET	0.91	0.51	0.36	14	23
DOG+CONVOPT	0.90	0.51	0.26	15	40
D2-NET	0.89	0.50	0.32	16	21
DOG+ROOTSIFT	0.89	0.49	0.29	17	34
LF-NET	0.89	0.45	0.23	18	44
SURF+CONVOPT	0.88	0.48	0.32	19	27
HARRIS+ROOTSIFT	0.87	0.55	0.16	20	65
HARRIS+SIFTNET	0.87	0.48	0.18	21	59
⋮					
SIFT	0.65	0.09	0.73	32	**1**
DOG+SIFT	0.44	0.07	0.58	40	5
DOG+PCA-SIFT	0.43	0.07	0.58	41	6
HARRIS+SIFT	0.38	0.04	0.43	43	16
HARRIS+TFEAT	0.38	0.03	0.60	44	3
HARRIS+PCA-SIFT	0.37	0.04	0.42	45	17
SURF+SURF	0.33	0.05	0.57	48	7
HARRISNET+TFEAT	0.29	0.08	0.62	50	2
SURF+TFEAT	0.26	0.04	0.56	54	4
DOG+TFEAT	0.22	0.04	0.51	55	8
FAST+PCA-SIFT	0.19	0.01	0.48	57	11
FAST+SIFT	0.19	0.01	0.50	58	10
HARRISNET+SIFT	0.18	0.07	0.45	59	13
HARRISNET+PCA-SIFT	0.18	0.07	0.44	60	14
KEY.NET+TFEAT	0.11	0.04	0.50	63	9
SURF+SIFT	0.07	0.01	0.40	65	19
SURF+PCA-SIFT	0.06	0.01	0.40	67	18

Table 3.11: Results of the w**DM** (YFCC-100M) benchmark (Figure 3.5). Top ranking methods by UIF-UB and Inlier Ratio are shown in the table with their ranks. Results are sorted by their UIF-UB score.

D+M+V	**UIF (E-matrix)**	**UIF (F-matrix)**
DoG+ConvOpt+OA-Net	0.42	0.39
DoG+RootSIFT+OA-Net	0.42	0.39
DoG+OpenUCN+OA-Net	0.33	0.33
DoG+SIFTnet+OA-Net	0.29	0.25
Key.Net+SIFTnet+OA-Net	0.26	0.16
HarrisNet+RootSIFT+OA-Net	0.26	0.17
BRISK+OA-Net	0.26	0.23
SIFT+RANSAC-0.5px	0.26	0.16
Key.Net+SpreadOutHardNet+OA-Net	0.25	0.18
SIFT+LMedS	0.24	0.20
⋮		
SIFT+RANSAC-3px	0.14	0.11
SIFT+NG-RANSAC	0.11	0.04
FAST+SpreadOutHardNet+NG-RANSAC	0.08	0.05
HarrisNet+TFeat+LMedS	0.06	0.05
FAST+SpreadOutHardNet+Eig-Free	0.04	0.04
SURF+OpenUCN+Eig-Free	0.04	0.04
DoG+SpreadOutHardNet+Learned-Corr	0.04	0.04
HarrisNet+SpreadOutHardNet+Learned-Corr	0.04	0.03
SURF+ConvOpt+N^3-Net	0.04	0.03
R2D2+N^3-Net	0.04	0.04
Harris+SIFT+RANSAC-3px	0.03	0.02
FAST+RootSIFT+MLESAC	0.00	0.00
FAST+ConvOpt+MLESAC	0.00	0.00

Table 3.12: Results of the w**DMV** (YFCC-100M) benchmark (Figure 3.10), sorted by Usable Image Fraction for Essential Matrix (UIF E-Matrix).

CHAPTER 4
VALIDATING OUTDOOR HD MAPS

In this chapter, we turn from a focus on building and validating *geometric* maps to a focus on validating *semantic* maps, especially in the self-driving domain. High-definition (HD) map change detection is the task of determining when sensor data and map data are no longer in agreement with one another due to real-world changes. We collect the first dataset for the task, which we entitle the *Trust, but Verify* (TbV) dataset, by mining thousands of hours of data from over 9 months of autonomous vehicle fleet operations. We present learning-based formulations for solving the problem in the bird's eye view and ego-view. Because real map changes are infrequent and vector maps are easy to synthetically manipulate, we lean on simulated data to train our model. Perhaps surprisingly, we show that such models can generalize to real world distributions. The dataset, consisting of maps and logs collected in six North American cities, is one of the largest AV datasets to date with more than 7.8 million images. We make the data[1] available to the public, along with code and models[2] under the the CC BY-NC-SA 4.0 license.

4.1 Problem Introduction

We live in a highly dynamic world, so much so that significant portions of our environment that we assume to be static are, in fact, in flux. Of particular interest to self-driving vehicle development is changing road infrastructure. Road infrastructure is often represented in an onboard map within a geo-fenced area. Geo-fenced areas have served as an operational design domain for self-driving vehicles since the earliest days of the DARPA Urban Challenge [277, 278, 279].

One way such maps could be used is to constrain navigation in all free space to a set of legal "rails" on which a vehicle can travel. Maps may also be used to assist in planning beyond the sensor range and in harsh environments. Besides providing routes for navigation, maps can ensure that the autonomous vehicle (AV) follows local driving laws when navigating through a city. They embody a representation of the world that the AV can understand, and contain valuable information about the environment.

However, maps assume a static world, an assumption which is violated in practice; although these changes are rare, they certainly occur and will continue to occur, and can have serious implications. Level 4 autonomy is defined as sustained performance by an autonomous driving system within an operational design domain, without any expectation that a user will respond to a request to intervene [280]. Thus, constant verification that the represented world, expressed as a map, matches the real world, a task which we name *map change detection*, is a clear requirement for L4 autonomy. Because dedicated mapping vehicles cannot traverse the world frequently enough to keep maps up to date [281], high-definition (HD) maps become "stale," with out of date information. If maps are used as hard priors, this could lead to confident but incorrect assumptions about the environment.

[1]Data is available at Argoverse.org.

[2]Code and models are available at github.com/johnwlambert/tbv.

In this work, we present the first dataset for urban map change detection based on actual, observed map changes, which we name *TbV*. Not only does no comparable dataset exist, there also has not even been an attempt to characterize how often map changes occur and what form they take. Collecting data for map change detection is challenging since changes occur randomly and infrequently. In addition, in order to use data corresponding to real changes to train and evaluate models, identified changes must be manually localized in both space and time. Concurrent work [282] presents qualitative results on a handful of real-world map changes, but depends upon synthetic test datasets for all quantitative evaluation.

HD map change detection is a difficult task even for humans, as it requires the careful comparison of all nearby semantic entities in the real world with all nearby map elements in the represented world. In an urban scene, there can be dozens of such entities, many with extended shapes. The task is sufficiently difficult that several have even questioned the viability of HD maps for long-term autonomy, opting instead to pursue HD-map-free solutions [283].

We concentrate on changes to two types of semantic entities – lane geometry and pedestrian crosswalks. We define the task as correctly classifying whether a change occurred at evenly spaced intervals along a vehicle's trajectory.

The task itself is relatively new, especially since HD maps were not made publicly available until the release of the Argoverse, nuScenes, Lyft Level5, and Waymo Open Motion datasets [4, 284, 285, 286]. We present the first entirely learning-based formulation for solving the problem in either a bird's eye view (BEV), as well as a new formulation for the ego-view (i.e. front camera frustum), eliminating several heuristics that have defined prior work. We pose the problem as learning a representation of maps, sensor data, or the combination of the two.

Our contributions are as follows:

- We present a novel AV dataset, with 799 vehicle logs in our train and synthetic validation splits, and over 200 vehicle logs with real-world map-changes in our real val and test splits.
- We implement various learning-based approaches as strong baselines to explore this task for the first time with real data. We also demonstrate how gradients flowing through our networks can be leveraged to localize map changes.
- We analyze the advantages of various data viewpoint by training both models operating on the ego-view and others on a bird's eye view.
- We show that synthetic training data is useful for detecting real map changes. At the same time, we identify a considerable domain gap between synthetic and real data, with significant performance consequences.

4.2 Related Work

HD Maps. HD maps include lane-level geometry, as well as other geometric data and semantic annotations [287, 288, 289, 2, 290, 4, 105, 10, 291, 6, 292, 7, 13, 11, 12]. The Argoverse [4], nuScenes [284], Lyft Level 5 [285], and Waymo Open Motion [286] datasets are the only publicly available sources of HD maps today, all with different semantic entities. Argoverse [4] includes a ground surface height map, rasterized driveable area, lane centerline geometry, connectivity, and other attributes. nuScenes [284] followed by also releasing centerline geometry, pedestrian crossing polygons, parking areas, and sidewalk polygons, along with rasterized driveable area. Lyft Level 5 [285] later provided a dataset with many more map entities, going beyond lane marking boundaries, crosswalks to provide traffic signs, traffic lights, lane restrictions, and speed bumps. Most recently, the Waymo Open Motion Dataset [286] released motion forecasting scenario data with associated HD maps. Their yet-richer HD map representation includes crosswalk polygons, speed bump polygons, lane boundary polylines with marking type, lane speed limits, lane types, and stop sign positions and their corresponding lane associations; their map data is most comparable with our HD maps.

HD Map Change Detection. HD map change detection is a recent problem, with limited prior work. Pannen *et al.* [293] introduce one of the first approaches; two particle filters are run simultaneously, with one utilizing only Global Navigation Satellite System (GNSS) and odometry measurements, and the other filter using only odometry with camera lane and road edge detections. These two distributions and sensor innovations are then fed to weak classifiers. Other prior work in the literature seeks to define hand-crafted heuristics for associating online lane and road detections with map entities [294, 112, 281]. These methods are usually evaluated on a single vehicle log [294].

Instead of comparing vector map elements, Ding *et al*. [114] use 2d BEV raster representations of the world; first, IMU-motion-compensated LiDAR odometry is used to build a local online "submap". Afterwards, the submap is projected to 2d and overlaid onto a prebuilt map; the intensity mean, intensity variance, and altitude mean of corresponding cells are compared for change detection. Rather than pursuing this approach, which requires creating and storing high-resolution reflectance maps of a city, we pursue the alignment of *vector maps* with sensor data. Vector maps can be encoded cheaply with low memory cost and are the more common representation, being used in all four public HD map datasets.

In concurrent work, Heo *et al*. [282] introduce an adversarial metric learning-based formulation for HD map change detection, but access to their dataset is restricted to South Korean researchers and performance is measured on a synthetic dataset, rather than on real-world changes. They employ a forward-facing ego-view representation, and require training a second, separate U-Net model to localize changed regions in 2d, whereas we show changed entity localization can come for free via examination of the gradients of a single model.

Mapping Dynamic Environments. While "HD maps" are a relatively new entity, dynamic map construction is a more mature field of study. Semi-static environments are not limited to urban streets; households, offices, warehouses, and parking lots are relatively fixed environments that a robot may navigate, with changing cars, furniture, and goods [295]. Mapping dynamic environments has been an area of study within the SLAM community for decades [296, 297, 298]. However, we focus purely on change detection, rather than map updates.

Recently, machine learning for online mapping has generated interest. An alternative to using an HD map prior is to rebuild the map on-the-fly during robot operation; however, such an approach cannot map occluded objects or entities. In addition, these methods are limited to producing raster map layers, such as a driveable area mask, with an output resembling semantic segmentation. Raster data is significantly less useful than vector data for path planning and generating vector map data with machine learning is generally an unsolved problem. Raster map layers may be generated from LiDAR [2], accumulated from networks operating on ego-view images over multiple cameras and timesteps [299, 300, 301], or from a single image paired with a depth map or LiDAR [302]. They all show that automatic mapping is quite challenging.

Image-to-Image Change Detection. Image-to-image scene change detection over the temporal axis is a well-studied problem [303, 304]. Scenes are dynamic over time in numerous ways, and those ways are mostly nuisance variables for our purposes. We wish to develop models invariant to season, lighting, the fading of road markings, and occlusion because these variables don't actually change the lane geometry. Wang *et al.* [303] introduced the CDnet benchmark, a collection of videos with frame pixels annotated as static, shadow, non-ROI, unknown, or moving. Alcantarilla [304] *et al.* introduce the VL-CMU-CD street view change detection benchmark, from a subset of the Visual Localization CMU dataset.

4.3 The TbV Dataset

We curate a novel dataset of autonomous vehicle driving data comprising 1043 logs, over 200 of which contain map changes. The vehicle logs are on average 54 seconds in duration, collected in six North American cities: Austin, TX, Detroit, MI, Miami, FL, Palo Alto, CA, Pittsburgh, PA, and Washington, D.C. (See Table 4.1).

Our training set and validation set consist of real data with accurate corresponding onboard maps ("positives"). Accordingly, synthetic perturbation of positives to create plausible "negatives" is required for training. We release the data, code and API to generate them. However, in the spirit of other datasets meant for testing only (i.e. not training) such as the influential WildDash dataset [305], we curate our test dataset from the real-world distribution. We do so since map changes are difficult to mine [282], thus we save their limited quantity for the test set. We provide a few examples from our 133 "real" test logs and 111 real "val" logs in Figure 4.1. Statistics of the test split are described in Table 4.1. We separate 10% of the training data into the held-out "synthetic" validation split.

4.3.1 Annotation

In order to label map changes, we use three rounds of human filtering, where changes are identified, confirmed, and characterized by three independent reviewing panels. We assign spatial coordinates to each changed object within a city coordinate system. Crosswalk changes are denoted by a polygon, and lane geometry changes by polylines. We use egovehicle-to-changed-map-entity distance (point-to-polygon or point-to-line) to determine whether or not a sensor and map rendering should be classified as a map-sensor match or mismatch.

Analysis of Map Change Frequency We use our annotated map changes, along with 5 months of fleet data, to analyze the frequency of map changes on a city-scale across several cities. Two particular questions are of interest: (1) *how often will an autonomous vehicle encounter a map change as part of its day-to-day operation?* and (2) *what percentage of*

Table 4.1: We describe the statistics of the map deviation data in our test set, and the types of deviations we observe. We define each BEV frame as a pose where the egovehicle has moved at least 5 meters since the previous pose. Lane geometry changes extend over far more frames than crosswalk changes.

	Data Split	
	Train/Val	Test
Num Images	6,991,006	1,008,134
Avg. number of images per log (@20 Hz)	8,129	7,201
Num LiDAR Sweeps	511,208	74,937
Avg. number of LiDAR sweeps per log (@10 Hz)	594	535
Num. rendered BEV frames (once every 5 meters of trajectory)	25,363	4,945
Num. BEV frames with no changes	25,363	2,159
Num. BEV frames with changes	0	2,786
Num. BEV frames with crosswalk changes only	0	201
Num. BEV frames with lane geometry changes only	0	2,105
Num. BEV frames with both lane geometry and crosswalk changes	0	120

Table 4.2: Probability of a 30m × 30m region that has been visited at least 5 times in 5 months undergoing a lane geometry or crosswalk change within the same time period. These statistics apply only to surface-level urban streets, not highways.

	City Name					
	Pittsburgh	Detroit	Washington, D.C.	Miami	Austin	Palo Alto
Probability of Change	0.0068	0.0056	0.0046	0.0038	0.0009	0.0007
Up to T / 1000 tiles will change in 5 mo.	7	6	5	4	0.9	0.7

Table 4.3: Entities included in our HD map representation.

HD Map Entity	Corresponding attributes
Pedestrian Crossings	2 edges oriented along its principal axis
Lanes	Boundaries: 3D left and right polylines Color: yellow, white, or implicit Boundary marking type Connectivity Lane type: bike or vehicle lane In intersection: true or false
Driveable area	polygons
Ground surface height	floating point height values at 30 centimeter resolution

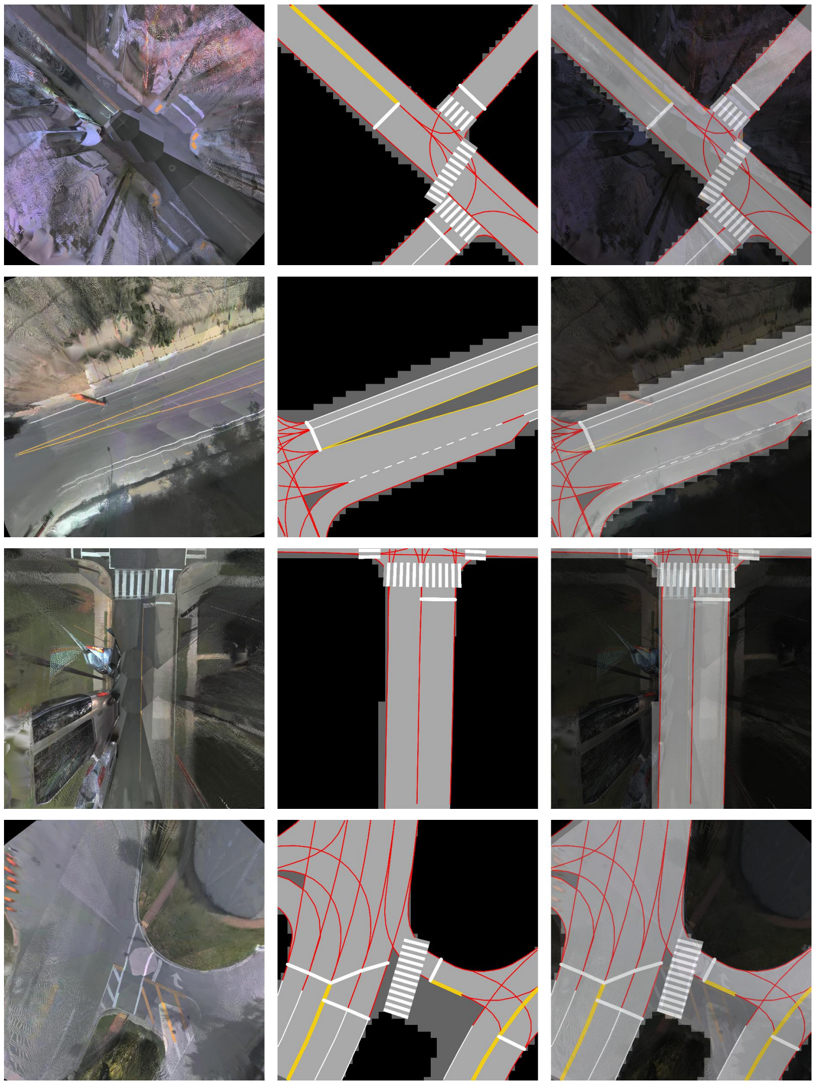
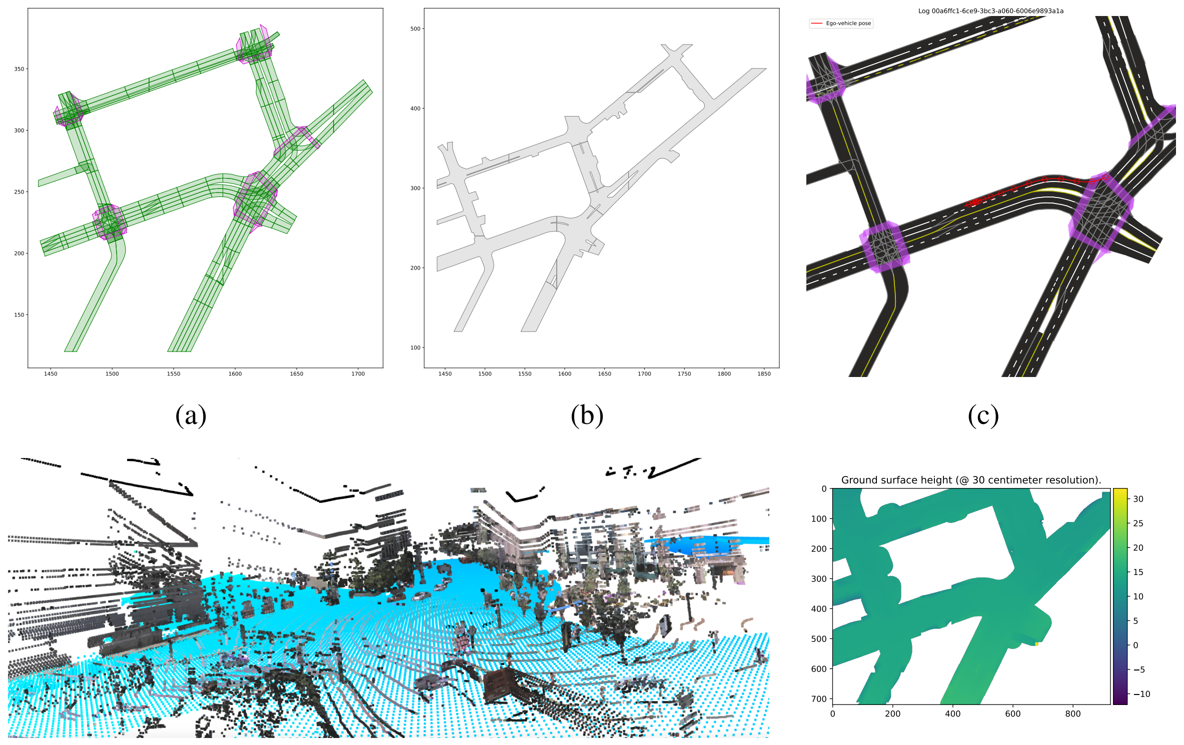


Figure 4.1: Examples from the test split of our TbV dataset. Left to right: BEV sensor representation, onboard map representation, blended map and sensor representations. Rows, from top to bottom: deleted crosswalk (top row), and painted lane geometry changes (bottom three rows).

map elements in a city will change each month or each year? For our analysis, we subdivide a city's map into square spatial units of dimension 30 meters $\times$ 30 meters, often referred to as "tiles" in the mapping community. We find the probability p of an encounter at any given time with a tile with changed lane geometry or crosswalk to be $p \approx 5.5174 \times 10^{-5}$. Given the 3.225 trillion miles driven in the U.S. per year [306], this could amount to *billions* of such encounters per year. We determine that up to 7 of every 1000 map tiles may change in a 5-month span (see Table 4.2), a significant number. More details are provided in the Appendix.

4.3.2 Sensor Data

Our TbV dataset includes LiDAR sweeps collected at 10 Hz, along with 20 fps imagery from 7 cameras positioned to provide a fully panoramic field of view. In addition, camera


Log 00a6ffc1-6ce9-3bc3-a060-6006e9893a1a
Ego-vehicle pose
Ground surface height (@ 30 centimeter resolution).

(a) (b) (c)

(d) (e)

Figure 4.2: Examples of lane graphs and pedestrian crossings **(a)**, drivable areas **(b)**, lane marking annotations **(c)** raster ground surface height data **(d, e)**, found in both TbV and the Argoverse 2.0 Sensor Datasets.

intrinsics, extrinsics and 6 d.o.f. AV pose in a global coordinate system are provided.

LiDAR returns are captured by two 32-beam LiDARs, spinning at 10 Hz in the same direction ("in phase"), but separated in time by $180°$. The cameras trigger in-sync with both of them, leading to a 20 Hz framerate. The 7 global shutter cameras are synchronized to the LiDAR to have their exposure centered on when the LiDAR sweeps through the middle of their fields of view. The top LiDAR spins clockwise in its frame, while the bottom LiDAR spins counter-clockwise in its frame; in the ego-vehicle frame, they both spin clockwise.

4.3.3 Map Data

In Table 4.3, we list the semantic map entities we include in the TbV dataset. Previous AV datasets have released sensor data localized within a single map per city [4, 284, 285]. This is not a viable solution for TbV, since the maps change over our long period of data gathering. We instead release local maps with all semantic entities within 20 meters of the egovehicle featured. Accordingly, single, incremental changes can be identified and tested. We release many maps, one per vehicle log; the corresponding map is the map used on-board at time of capture. Lane segments within our map are annotated with boundary annotations for both the right and left marking (including against curbs) and are marked as implicit if there is no corresponding paint (See Figure 4.2).

4.3.4 Dataset Taxonomy

Our dataset's taxonomy is intentionally oriented towards lane geometry and crosswalk changes. In general, we focus on permanent changes, which are far less frequent in urban areas than temporary map changes. Temporary map changes often arise due to construction and road blockades.

We postulate that temporary map changes – temporarily closed lanes or roads, or temporary lanes indicated by barriers or cones, should be relegated to onboard object recognition and detection systems. Indeed, recent datasets such as nuScenes [284] include 3d labeling for traffic cones and movable road barriers, such as Jersey barriers (see Appendix for examples). Even certain types of permanent changes are object-centered (e.g. changes to traffic signs). Accordingly, a natural division arises between "things" and "stuff' in map change detection, just as in general scene understanding [307, 308]. We focus on the "stuff" aspect, corresponding to entities which are often spatially distributed in the BEV; we find lane geometry and crosswalks to be more frequent than other "stuff"-related changes.

4.4 Approach

4.4.1 Learning Formulation

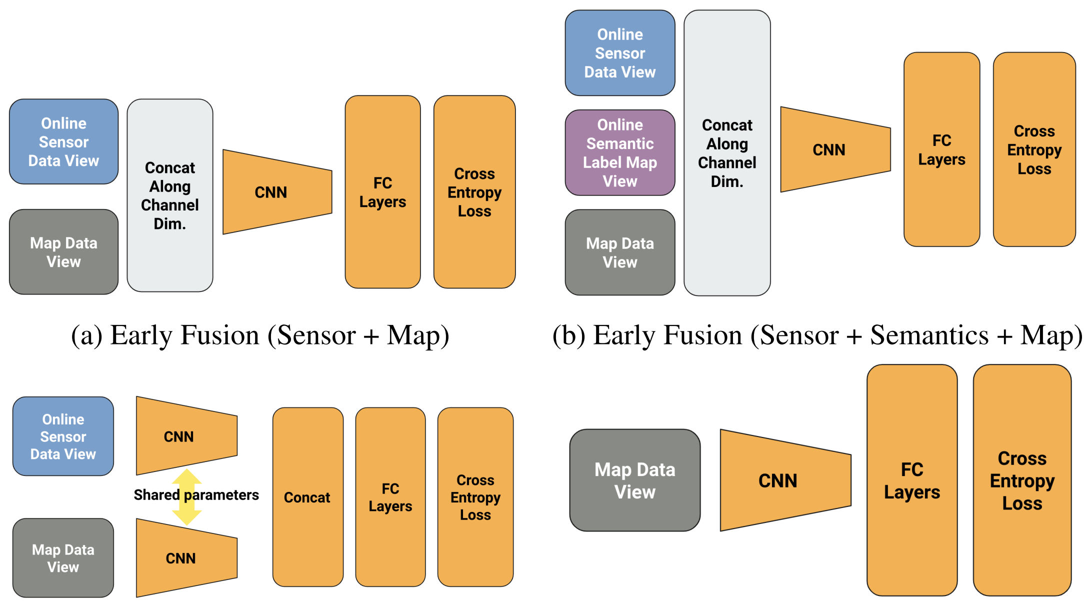
We formulate the learning problem as predicting whether a map is stale by fusing local HD map representations and incoming sensor data. We assume accurate pose is known. At training time, we assume access to training examples in the form of triplets (x, x^*, y), where x is a local region of the map, x^* is an online sensor sweep, and y is a binary label suggesting whether a "significant" map change occurred. (x, x^*) should be captured in the same location.

We explore a number of architectures to learn a shared map-sensor representation, including early fusion and late fusion (see Figure 4.3). The late fusion model uses a siamese network architecture with two input towers, and then a sequence of fully connected layers. We utilize a two-stream architecture [47, 309] with shared parameters, which has been shown to still be effective even for multi-modal input [310]. We also explore an early-fusion architecture, where the map, sensor, and/or semantic segmentation data are immediately concatenated along the channel dimension before being fed to the network. We take no credit for these convnet architectures, which are well studied.

4.4.2 Synthesis of Mismatched Data

Real negatives are difficult to obtain; because their location is difficult to predict a priori, they cannot be captured in a deterministic way by driving around an urban area on any particular day. Therefore, rather than using real negatives for training, we synthesize fake negatives. While sensor data is difficult to simulate, requiring synthesis of sensor measurements from the natural image manifold [311, 13], manipulating vector maps is relatively straightforward.

Synthetic data generation via randomized rendering pipelines can be highly effective for synthetic-to-real transfer [312]. In order to synthesize fake negatives from true positives, one must be able to trust the fidelity of labeled true positives. In other words, one must



(a) Early Fusion (Sensor + Map) (b) Early Fusion (Sensor + Semantics + Map)

(c) Late Fusion (Siamese) (d) Map-Only Input

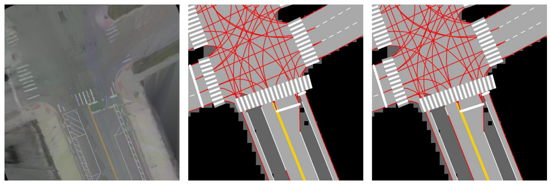
Figure 4.3: Learning architectures we explore for the map change detection problem.

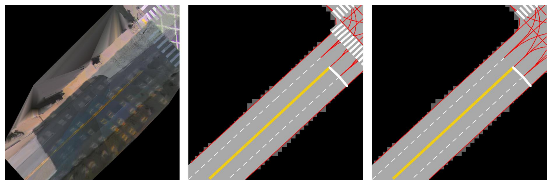
trust that for true positive logs, the map is completely accurate for the corresponding sensor data. We perturb the data in a number of ways (See Table 4.4). If such fidelity is assured, vector map manipulation is trivial because map elements are vector entities which can be perturbed, deleted, or added.

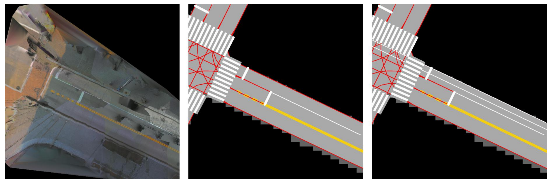
While synthesizing random vector elements is trivial, sampling from a realistic distribution requires conformance to priors, including the lane graph, drivable area, and intersection. We aim for synthetic map/sensor deviations to resemble real world deviations, and real world deviations tend to be subtle, e.g. a single lane is removed or painted a different color, or a single crosswalk is added, while 90% of the scene is still a match. In order to generate realistic-appearing synthetic map objects, we hand-design a number of priors that must be respected for a perturbed example to enter our training set as a valid training example (see Appendix). Figure 4.4 and Table 4.4 enumerate a full list of the 6 types of synthetic changes we employ.

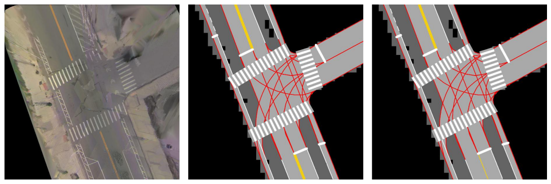
Table 4.4: Training dataset statistics and types of synthetic changes generated from 799 logs. Not all scenes can support all synthetic change types. For example, in order to delete a crosswalk from a local map, a crosswalk must be present of local vicinity of the egovehicle.

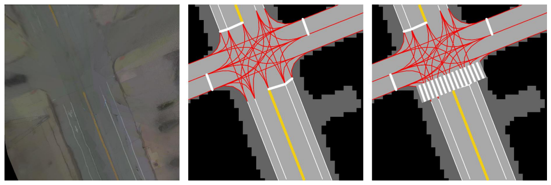
Change Category	Description of Change	Quantity of Examples
BEV Sensor Images	N/A	25,393
No change	None	25,263
Lane Geometry Changes	Delete Lane Marking	19,870
	Change Lane Marking Color	25,098
	Change Lane Boundary Dash-Solid	19,875
	Add Bike Lane	21,529
Crosswalk Changes	Delete Crosswalk	9,627
	Insert Crosswalk	23,166


(a) Lane marking color is changed from implicit to solid white (see bottom-center of image)


(b) A crosswalk is deleted from the map. Reflections off of windows create illumination variation on the road surface.


(c) A bike lane is added to the map (see center-right of image)


(d) The structure of a lane boundary marking is changed, from double-solid yellow to single-solid yellow (see bottom-center of image). Its color is preserved.


(e) A crosswalk is synthetically inserted into the map.

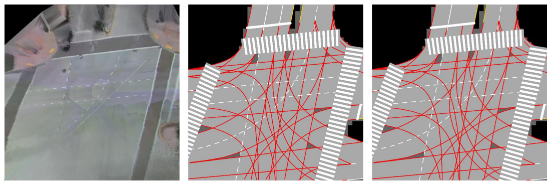

(f) A solid white lane boundary marking is deleted (see top-center of image).

Figure 4.4: Examples of our 6 types of synthetic map changes (zoom in for detail). Each row represents a single scene. Left: bird's eye view (BEV) sensor data representation. Center: rasterized onboard map representation (positive). Right: synthetic perturbation of onboard map (negative). We use red to denote implicit lane boundaries.

Table 4.5: Controlled evaluation of the influence of fusion architecture and scene rendering viewpoint (ego-view vs. BEV). Rows marked with an asterisk represent an expected mean accuracy based on randomly flipped labels, rather than results from a trained model.

			Modalities			Visibility-based Eval. @ 20m	BEV proximity Eval. @20m			Visibility-based Eval. @20m		
Backbone	Arch.	Viewpoint	RGB	Semantics	Map	val mAcc	test mAcc	Is Changed Acc	No Change Acc	test mAcc	Is Changed Acc	No Change Acc
-	Random Chance*	-	-	-	-	0.5000	0.5000	0.50	0.50	0.5000	0.50	0.50
ResNet-18	Early Fusion	Ego-View	✓	✓	✓	0.8417	**0.6724**	0.57	0.77	**0.7234**	**0.67**	0.78
ResNet-18	Late Fusion	Ego-View	✓		✓	0.8108	0.4930	0.13	**0.85**	0.4956	0.13	**0.86**
ResNet-50	Early Fusion	BEV	✓	✓	✓	**0.9130**	**0.6728**	**0.58**	0.77	-	-	-
ResNet-50	Late Fusion	BEV	✓		✓	0.8697	0.5761	0.43	0.72	-	-	-

Table 4.6: Controlled evaluation of the influence of data modalities. Rows marked with an asterisk represent an expected mean accuracy based on randomly flipped labels, rather than results from a trained model.

			Modalities			Visibility-based Eval. @ 20m	BEV proximity Eval. @20m			Visibility-based Eval. @20m		
Backbone	Arch.	Viewpoint	RGB	Semantics	Map	val mAcc	test mAcc	Is Changed Acc	No Change Acc	test mAcc	Is Changed Acc	No Change Acc
-	Random Chance*	-	-	-	-	0.5000	0.5000	0.50	0.50	0.5000	0.50	0.50
ResNet-18	Single Modality*	Ego-View	✓			0.5000	0.5000	0.50	0.50	0.5000	0.50	0.50
ResNet-18	Single Modality	Ego-View			✓	0.8444	0.5333	0.36	0.70	0.5431	0.38	0.71
ResNet-18	Early Fusion	Ego-View	✓		✓	0.8599	0.6463	0.52	**0.77**	0.6824	0.60	0.77
ResNet-18	Early Fusion	Ego-View		✓	✓	0.8632	0.6082	0.36	0.85	0.6363	0.42	**0.85**
ResNet-18	Early Fusion	Ego-View	✓	✓	✓	0.8417	**0.6724**	0.57	**0.77**	**0.7234**	**0.67**	0.78
ResNet-50	Single Modality*	BEV	✓			0.5000	0.5000	0.50	0.50	-	-	-
ResNet-50	Single Modality	BEV			✓	0.8900	0.5754	0.50	0.65	-	-	-
ResNet-50	Early Fusion	BEV	✓		✓	0.9007	0.6543	0.57	0.74	-	-	-
ResNet-50	Early Fusion	BEV		✓	✓	0.9153	0.6615	0.60	0.72	-	-	-
ResNet-50	Early Fusion	BEV	✓	✓	✓	**0.9130**	**0.6728**	**0.58**	**0.77**	-	-	-

4.4.3 Sensor Data Representation

We experiment with two sensor data representations – *ego-view* (the front center camera image) and *bird's eye view* (BEV). Rather than using Inverse Perspective Mapping (IPM) [313, 314, 315], we generate the BEV representation (i.e. orthoimagery) by ray-casting image pixels to a ground surface triangle mesh. For ray-casting, we use a set of camera sensors with a panoramic field of view, mounted onboard an autonomous vehicle. The temporal aspect is exploited as pixel values from 70 ego-view images are aggregated to render each BEV image (10 timesteps from 7 frustums) in order to reduce sparsity (see Appendix).

4.4.4 Map Data Learning Representation

We render our map inputs as rasterized images; Entities are layered from the back of the raster to the front in the following order: driveable area, lane segment polygons, lane boundaries, pedestrian crossings (i.e. crosswalks). We will release the API to generate and render these map images. Vector map entities are synthetically perturbed before rasterization.

4.5 Experimental Results

We frame the map change detection task as: *given a buffer of all past sensor data at timestamp* t, *including camera intrinsics and extrinsics, 6 d.o.f. egovehicle pose* ${}^{city}T_{egovehicle}$ which we denote as $T_{i=0 \dots t}$, *image data* $I^{c}_{i=0 \dots t}$ *where* c *is a camera index, lidar sweeps* $L_{i=0 \dots t}$, *onboard map data* $M_{k=0 \dots K}$, *estimate whether the map is in agreement with the sensor data.*

Table 4.7: Controlled evaluation of the benefit and influence of dropout of data modalities. Rows marked with an asterisk represent an expected mean accuracy based on randomly flipped labels, rather than results from a trained model.

			Modalities			Visibility-based Eval @ 20m	BEV proximity Eval. @20m			Visibility-based Eval @20m		
Backbone	Arch.	Viewpoint	RGB	Semantics	Map	val mAcc	test mAcc	Is Changed Acc	No Change Acc	test mAcc	Is Changed Acc	No Change Acc
-	Random Chance*	-	-	-	-	0.5000	0.5000	0.50	0.50	0.5000	0.50	0.50
ResNet-18	Early Fusion	Ego-View	✓	✓	✓	0.8417	0.6724	0.57	0.77	0.7234	0.67	0.78
ResNet-18	Early Fusion	Ego-View	dropout	dropout	✓	**0.8605**	0.6581	0.51	0.81	0.6926	0.58	**0.81**
ResNet-18	Early Fusion	Ego-View	✓	dropout	dropout	0.8384	**0.6850**	**0.63**	0.74	0.**7342**	**0.72**	0.74
ResNet-18	Early Fusion	Ego-View	✓	dropout	✓	0.8474	0.6483	0.51	0.78	0.6914	0.6	0.79
ResNet-18	Early Fusion	Ego-View	✓	✓	dropout	0.8429	0.6617	0.51	**0.82**	0.6994	0.58	**0.81**

4.5.1 Implementation Details

Ego-view Models. Our ego-view models that operate on front-center camera images leverage both LiDAR and RGB sensor imagery. We use LiDAR information to filter out occluded map elements from the rendering. We linearly interpolate a dense depth map from sparse LiDAR measurements, and then compare the depth of individual map elements against the interpolated depth map; elements found behind the depth map are not rendered. In our early fusion architecture, we experiment with models that also have access to semantic label masks from the semantic head of a publicly-available *seamseg* ResNet-50 panoptic segmentation model [316]. For those models with access to the semantic label map modality, we append 224×224 binary masks for 5 semantic classes ('road', 'bike-lane', 'marking-crosswalk-zebra', 'lane-marking-general', and 'crosswalk-plain') as additional channels in early fusion.
Bird's Eye View Models. We also implement camera-based models that accept orthoimagery as input, relying only upon RGB imagery and a pre-generated ground height field, without utilizing online LiDAR. We generate new orthoimagery each time the ego-vehicle moves at least 5 meters, and use each orthoimagery example as a training or test example. We use 2 cm per pixel resolution for orthoimagery; all pixels corresponding to 3d points up to 20 meters in ℓ_∞-norm from the ego-vehicle are included, generating a 2000×2000 px image.
Training. We use a ResNet-18 or ResNet-50 [159] backbone, with ImageNet-pretrained weights, where a corresponding weight parameter's size is applicable. We use a crop size of 224×224 from images resized to 234×234 px. Please refer to the Appendix for additional implementation details and an ablation experiment on the influence of input crop size on performance.

4.5.2 Evaluation

Comparable evaluation of the ego-view and bird's eye view models is challenging since they operate on different portions of the scene. The ego-view model should not be penalized for ignoring changes outside of its field of view, especially those located behind the ego-vehicle. Thus, we provide results for visibility-based evaluation (when the change is visible in the ego-view), and a purely proximity-based comparison (when it is within 20 m. by ℓ_∞ norm). The area about which a model should reason is somewhat arbitrary; changes behind and to the side may matter for fleet operation, but changes directly ahead of the AV are arguably most important for path-planning [317]. In addition, changes visible to the rear at some timestamp are often visible directly in front of the AV at a prior timestamp. We consider the visibility-based evaluation to be most fair for ego-view models.

We use a mean of per-class accuracies to measure performance on a two-class problem: predicting whether the real world is changed (i.e. map and sensor data are mismatched), or unchanged (i.e. a valid match).

Comparison of Performance from Different Data Viewpoints and Architectures. Our first finding is that models that operate on an ego-view scene perspective are more effective than those operating in the bird's eye view (5% more effective over their own respective field of view), achieving 72.3% mAcc (see Table 4.5). We found a simpler architecture (ResNet-18) to outperform ResNet-50 in the ego-view.

For both BEV and ego-view, the early fusion models significantly outperform the late fusion models (+22.8% mAcc in the ego-view and +9.7% mAcc in BEV). This may be surprising, but we attribute this to the benefit of early alignment of map and sensor image channels for the early fusion models. Instead, the late-fusion model performs alignment with greatly reduced spatial resolution in a higher-dimensional space, and is forced to make decisions about both data streams *independently*, which may be suboptimal. While the map and sensor images represent different modalities, a shared feature extractor is useful for both.

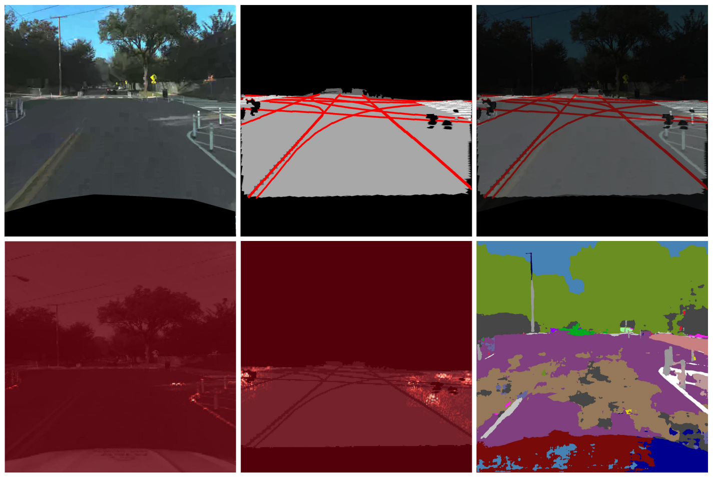
Comparison of Performance from Different Input Modalities. We compare validation and test set performance of various input modalities in Table 4.6. Early fusion of map and sensor data is compared with models that have access to only sensor or map data, or a combination of the two, with or without semantics. All models suffer a significant performance drop on the test set compared to the validation set. While a gap between validation and test performance is undesirable, better synthesis heuristics and better machine learning models can close that gap.

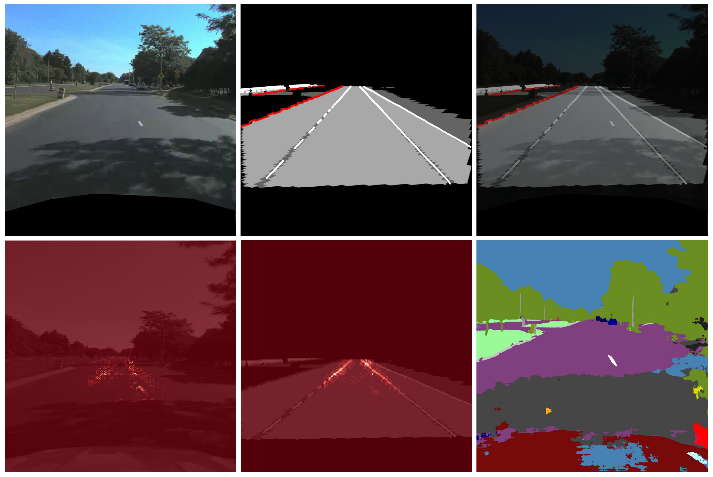
We find semantic segmentation label map input to be quite helpful even, although it places a dependency upon a separate model at inference time, increasing latency for a real-time system. Mean accuracy improves by 4% in the ego-view and 2% in the BEV when sensor and map data is augmented with semantic information in early fusion. In fact, early fusion of the map with the semantic stream alone (without sensor data) is 1% more effective than using corresponding sensor data for the BEV.

The map-stream-only models perform slightly better than random chance. Inspection via Guided GradCAM demonstrates that the map-only baseline attends to onboard map areas that are not in compliance with real-world priors, such as symmetry in crosswalk layout, paint patterns, and lane subdivisions (see Appendix).

Ablation on Modality Dropout. We find random drop-out of certain combinations of modalities to regularize the training in a beneficial way, improving accuracy by more than 1% of our best model (See Table 4.7). Given the wide array of modalities available to solve the task, from RGB sensor data, semantic label maps, rendered maps, and LiDAR reflectance data, we experiment with methods to force the network to learn to extract useful information from multiple modalities. Specifically, we perform random dropout of modalities, an approach developed in the self-supervised learning literature [318, 319, 320].

Perhaps the most intuitive approach would be to apply modality dropout to one of the sensor or semantic streams, forcing the network to extract useful features from both modalities during training. However, we find this is in fact detrimental. More effective, we dis-


(a) New paint constricting the intersection and bollards are added.


(b) A 3-lane road has been converted to a 2-lane road.

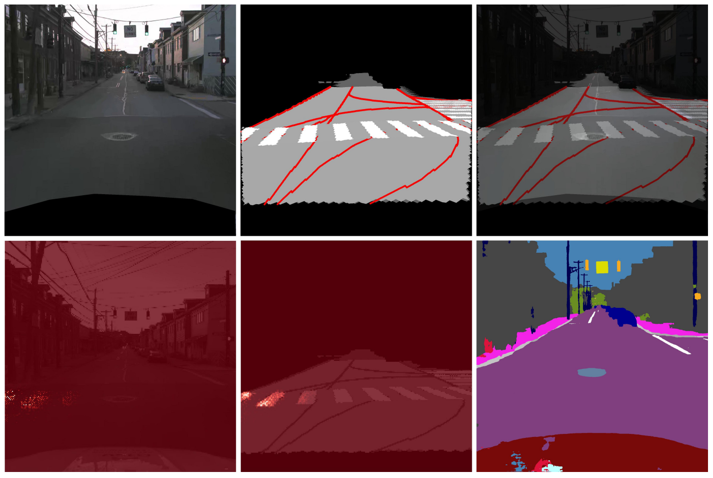

(c) Crosswalk paint has been removed.

Figure 4.5: Guided GradCAM. 6 figures are shown for frames from various test set logs. Clockwise, from top-left: ego-view sensor image, rendered map in ego-view, blended combination of sensor and map, *seamseg* label map, GradCAM activations for the map input, GradCAM activations for the sensor input. White color shows maximal activation, and red color shows zero activation in the heatmap palette. Label maps from *seamseg* are at times quite noisy.

cover, is to randomly drop out either the map or semantic streams. In theory, meaningful learning should be impossible without access to the map; however, since we drop-out each example in a batch with 50% probability, in expectation 50% of the examples should yield useful gradients in each batch. This approach improves accuracy by more than 1% of our best model. We zero out all data from a specific modality as our drop out technique.
Interpretability and Localizing Changes. While accurately perceiving changes is important, the ability to localize them would also be helpful. We use Guided Grad-CAM [321] to identify which regions of the sensor, map, and semantic input are most relevant to the prediction of the 'is-changed' class. In Figure 4.5, we show qualitative results on frames for which our best model predicts real-world map changes have occurred.

4.6 Conclusion

Discussion. In this work, we have introduced the first dataset for a new and challenging problem, map change detection. Our dataset is one of the largest AV datasets at the present time, featuring 1043 logs with an average duration of 54 seconds. We implement various approaches as strong baselines to explore this task for the first time with real data. Perhaps surprisingly, we find that comparing maps in a metric representation (a bird's eye view) is inferior to operating directly in the ego-view. We attribute this to a loss of texture during the projection process, and to a more difficult task of reasoning about a much larger spatial area (85° f.o.v. instead of 360° f.o.v.). In addition, we provide a new method for localizing changed map entities, thereby facilitating efficient updates to HD maps.

We identify a significant gap between validation accuracy and test accuracy – 10-20% less on the test split – which supports the importance of testing on real data. If performance is only measured on fake changes that resemble one's training distribution, performance can appear much better than what occurs in reality. Real changes can be subtle, and we hope the community will use this dataset to further push the state-of-the-art we introduce. Our data, models, and code to generate our dataset and reproduce our results will be made publicly available.
Limitations. Rendering time. A second limitation of our work is that real-time rendering requires GPU hardware; in the ego-view, map entity tesselation and rasterization are costly, whereas in the BEV, ray-casting is computationally intensive. **Perturbation diversity.** In our work, we introduce just 6 types of possible map perturbations, of which far more types are possible; nonetheless, we prove that they are surprisingly useful. **Accuracy.** Perhaps last of all, although our baselines have reasonable performance and by inspection we demonstrate they are learning to attend to meaningful regions, a large gap still exists before such a model would be accurate enough to be used on-vehicle.

4.7 Appendix

In this appendix, we provide additional details about our dataset and experiments. In Section (A), we provide an ablation study on the influence of input crop size on model performance. In Section (B), we discuss additional implementation details about our training, data augmentation, and occlusion-based map rendering process. In Section (C), we discuss the paired positive-negative logs we include. In Section (D), we describe our evaluation

Table 4.8: Controlled evaluation of the influence of input crop size (for ego-view and BEV).

				Modalities			Visibility-based Eval. @ 20m	BEV proximity Eval. @20m			Visibility-based Eval. @20m		
Resolution	Backbone	Arch.	Viewpoint	RGB	Semantics	Map	val mAcc	test mAcc	Is Changed Acc	No Change Acc	test mAcc	Is Changed Acc	No Change Acc
224x224	ResNet-18	Early Fusion	Ego-View	✓	dropout	dropout	0.8384	**0.6850**	**0.63**	0.74	**0.7342**	**0.72**	0.74
448x448	ResNet-18	Early Fusion	Ego-View	✓	dropout	dropout	0.8713	0.6331	0.38	**0.88**	0.6644	0.45	**0.88**
224x224	ResNet-50	Early Fusion	BEV	✓	no	✓	0.9007	0.6543	0.57	0.74			
448x448	ResNet-50	Early Fusion	BEV	✓	no	✓	**0.9072**	0.6749	**0.63**	0.72			

metric. In Section (E), we provide additional experimental analysis of different models and rendering viewpoints. In Section (F), we provide additional details about how we generate orthoimagery. In Section (G), we offer additional examples from our test set. In Section (H), we give examples of other types of temporary map changes which we do not annotate or evaluate within our dataset. In Section (I) we provide further analysis of the frequency of map changes. Finally, in Section (J), we give additional details about our synthetic map perturbation protocol.

3D point locations are quantized to float16. Ground height maps are quantized to 0.3 meter resolution from their full resolution. HD map polygon vertex locations are quantized to .01 meter resolution.

Appendix A: Influence of Input Crop Size

In this section, we perform an ablation on input crop size, as discussed in subsection 4.5.1 of this chapter. Earlier in this chapter, we set our input crop size to 224×224 px for all experiments mentioned therein. In this section, we present an ablation to measure the influence of input crop size. Again, we find the ego-view model is the best-performing model, as measured on its own field of view.

Perhaps surprisingly, we find that an RGB image at 234×234 px resolution ($\sim$ 164K pixel values/image) is sufficient to capture significant detail. In Table 4.8, we present an ablation where we find that for BEV models, higher resolution (i.e. 468×468 px) does improve mAcc by 2% mAcc, although requiring almost 4x the GPU memory during training and significantly longer training times. However, for ego-view models, a higher crop size is quite detrimental, reducing visibility-based mAcc by around 7%.

Appendix B: Additional Implementation Details

B.1. Training

We train our models for 90 epochs with the Adam [322] optimizer. We use a polynomial learning rate decay strategy, starting at 1×10^{-3}. We use a batch size of 1024 examples. We start with pretrained ImageNet weights for ResNet-18 or ResNet-50 [159].

We train with multiple negative examples per sensor image, which we found to be more beneficial than randomly sampling a single negative example (i.e. a synthetically perturbed map). In other words, we perform multiple types of perturbations for a given scene, and feed them to the network as separate negative examples (not necessarily in the same mini-batch).

B.2. Data Augmentation

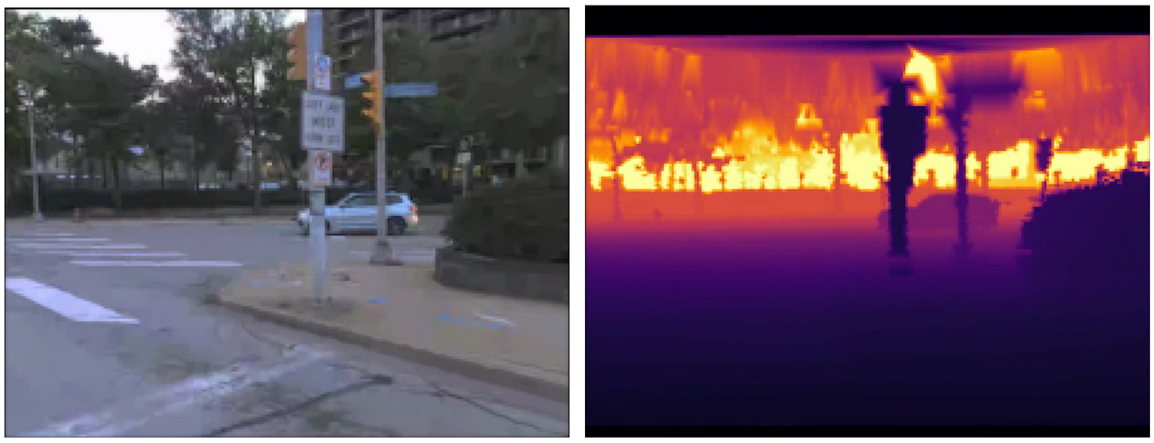
We employ a number of data augmentation techniques to improve the generalization of our models and prevent overfitting. Input images are of dimension 2048×1550 for the front-center camera, and 1550×2048 for all other 6 cameras. For the ego-view models, we first take a square crop from the bottom 1550×1550 of an ego-view image. Afterwards, we resize to 234×234, perform a random horizontal flip with 50% probability, take a random 224×224 crop, divide pixel intensities by 255, and then normalize both sensor and map RGB channels by the ImageNet mean $(\mu_r, \mu_g, \mu_b) = (0.485, 0.456, 0.406)$ and standard deviation $(\sigma_r, \sigma_g, \sigma_b) = (0.229, 0.224, 0.225)$

For BEV models, we resize input images from 2000×2000 px to 234×234 px, perform a random horizontal and/or vertical flip with 50% probability each (independently), choose a random 224×224 crop, and normalize as described above.

We find other traditional data augmentation techniques from the semantic segmentation literature [323], such as applying a random rotation to the input or randomly blurring the input with a small kernel, to be ineffective.

B.3. Occlusion Reasoning

As discussed in subsection 4.5.1 earlier in this chapter, we use map occlusion reasoning when generating the input for our ego-view models. Occluded map elements and map elements that have been removed in the real world ("deleted") are both not visible in camera imagery. While the former is an expected everyday occurrence, and the latter is of interest to us, we use occlusion reasoning in order to separate the two phenomena. We generate a dense depth map from sparse LiDAR returns (see Figure 4.6) and the depth of map entities is compared against the corresponding depth of its projection in the depth map.



(a) RGB Image (b) Interpolated Depth Map

Figure 4.6: Example of a dense depth map interpolated from sparse LiDAR returns.

B.4. Details about Semantic Label Map Input

As discussed in subsection 4.5.1 earlier in this chapter, we use semantic label maps generated from the semantic head of a publicly-available seamseg ResNet-50 panoptic segmenta-

tion model [316] [3]. We create 5 binary mask channels from the semantic label map, for the 'road', 'bike-lane', 'marking-crosswalk-zebra', 'lane-marking-general', and 'crosswalk-plain' classes. These are optionally provided as additional channels to the 3 RGB sensor channels and 3 RGB map channels via early fusion. Seamseg's semantic label maps on their own do not capture sufficient granularity for the map change detection task we define, since the Mapillary Vistas public dataset's taxonomy does not differentiate between lane color and or different marking types (e.g. double-solid, solid, dashed-solid), which are of interest to autonomous vehicle operation.
Unsuitability of Per-Pixel Semantic Comparison. Directly comparing rendered map and semantic label maps at a *per-pixel* level is not always useful since our HD map representation does not provide paint annotation for every single dashed longitudinal lane marking, but rather provides a description lane marking pattern, polyline boundary, and other corresponding attributes (See Table 4.3 earlier in this chapter). Thus, we can simulate the pattern of dashed lane markings, but not their exact, pixel-perfect location. As we show earlier in this chapter, the network can abstract away the per-pixel details to provide more meaningful features.

Appendix C: Data Selection

For a subset of the 'negative' logs in our TbV dataset, we provide a corresponding 'positive' log captured before the change occurred. Example images from pair positive-negative logs are provided in Figure 4.7. This allows for non-learning based approaches (e.g. based upon comparison of 3d reconstructed world models) for a limited amount of the test set.

Appendix D: Evaluation

As our primary accuracy metric, we use a mean of class accuracies over two classes. This accounts for both precision and recall. If a confusion matrix is computed with predicted entries on the rows and actual classes as the columns, and normalized by dividing by the sum of each column, 2-class accuracy can be simply calculated as the mean of the diagonal of the confusion matrix.

More formally, let $n_{cl} = 2$ be the number of classes, $\hat{y}_i$ be the prediction for the i'th test example, and y_i be the ground truth label for the i'th test example. We define per-class accuracy (Acc$_c$) and mean accuracy (mAcc) as:

$$\text{mAcc} = {}^1\!/\!_{n_{cl}} \sum_{c=0}^{n_{cl}} Acc_c, \quad \text{Acc}_c = \frac{\sum\limits_{i=0}^{N} \mathbb{1}\{\hat{y}_i = y_i\} \cdot \mathbb{1}\{y_i = c\}}{\sum\limits_{i=0}^{N} \mathbb{1}\{y_i = c\}} \tag{4.1}$$

Appendix E: Additional Experimental Analysis

Advantages of BEV. In principle, the bird's eye view (BEV) representation (orthoimagery) offers two main advantages: a single, dense, accumulated metrically-accurate representa-

[3]Available at https://github.com/mapillary/seamseg.

tion for a single pass through a network, rather than passing in 7 images through 7 separate networks, trained on each frustum, in order to detect changes to the sides and rear of the vehicle. This approach can be costly at inference time given the number of camera frustums required to achieve a panoramic view with traditional cameras. Second, the BEV is generally free of distortion, compared to the ego-view. The ego-view can be seen as "spoiling" the map data's metric nature.

Advantages of Ego-view. However, an ego-view perspective also presents clear advantages over the BEV. Rendering data in the BEV can be seen as "spoiling" the sensor data's texture. Importantly, there is less distraction and less overall content to reason about in the egoview. Therefore, the ego-view task is arguably easier than the BEV task, needing only to detect changes in a $85°$ f.o.v. instead of $360°$ f.o.v.

Analysis of Map-Only Baseline. The map-only baseline performs quite poorly when predicting real-world lane geometry changes, slightly over random chance (2% or 3% over random chance in the ego-view and 7% over random change in the BEV). While the map-only stream may seem doomed to fail without access to real-world sensor information, we observe that a certain number of map changes exist to bring the real world into compliance with certain priors, which are already encapsulated in the map. For example, we find that upgrading a 4-way intersection from a single crosswalk to 4 crosswalks, or from a single crosswalk to 0 crosswalks (after repaving) is a common map change, which would agree with priors. Indeed, our experimental results suggest that the map-only baseline, which is completely blind to the real-world, can occasionally succeed at predicting real-world crosswalk changes by learning powerful priors. Inspection via Guided GradCAM demonstrates that the map-only models attends to asymmetric paint patterns along the left and right boundaries of a road, or asymmetric lane subdivisions along two sides of a road; modifications to such map asymmetry which are common real-world map updates.

Analysis of Sensor-Only Baseline. The sensor-only model (see Table 4.6) sees randomly perturbed labels, with only "positive" training data, and therefore is not a meaningful baseline.

Appendix F: Orthoimagery Generation Implementation Details

In this section, we provide additional details about the orthoimagery generation process described in subsection 4.4.3 and subsection 4.5.1 earlier in this chapter. In order to create a metrically-accurate sensor data representation that is free of perspective distortion, we generate orthoimagery using ray-casting. Orthoimagery from LiDAR suffers from extreme sparsity, leading to an impoverished representation. To generate dense panoramic orthoimagery, we use a set of high-definition camera sensors with a panoramic field of view, mounted onboard an autonomous vehicle. We generate the BEV representation (i.e. orthoimagery) by ray-casting image pixels to a ground surface triangle mesh. Our ground height maps exploit LiDAR offline, and in this way our ego-view method incorporates the strengths of LiDAR.

CUDA Ray-Casting Routine. We tesselate quads from a ground surface mesh with 1 meter resolution to triangles; rays are cast to triangles up to 25 m away from the egovehicle. For acceleration, we cull triangles outside of the left and right cutting planes of each camera's view frustum. We implement the Moller-Trombore ray-triangle intersection routine

[324] in CUDA.
Density. Ray-casting yields a vastly more dense set of image rays than LiDAR, on the order of 2 orders of magnitude greater density; for a 1550×2048 image, one can obtain ~ 3.17 million rays per image, and across 7 camera frustums, this translates to over 22.19 million rays with available RGB values per second. With 20 fps imagery per camera frustum, this amounts to 440 million rays per second. Most conventional 10 Hz LiDAR sensors can provide little more than 100k returns per sweep, and thus at most 1 million rays per second.
Aggregation. In order to prevent holes in the orthoimagery in the area underneath the egovehicle, we aggregate pixels in ring buffer of length 10 sweeps, and wait 10 sweeps before starting rendering. Future sensor data is not used to render the sensor data representation. We use linear interpolation to account for sparsity at range.
Comparison with IPM. While Inverse Perspective Mapping (IPM) is the dominant approach in the literature, it is inaccurate as it cannot account for ground surface variation. Geiger [313] model the image-to-ground plane mapping as a homography (IPM) and mosaics together monocular images, but requires scenes with an approximately-planar ground surface. Zhang *et al.*[314] generate orthophoto ground imagery using fisheye cameras and IPM. Rapo [315] explored the use of dashboard-mounted cell phones without access to LiDAR or known calibration, instead relying upon SfM, optical flow, and vanishing point estimation for online calibration and also use IPM for pixel-to-world correspondence.

Appendix G: Additional Examples from Test Set

In Figure 4.8, we show additional examples from our test set, as seen from a bird's eye view.

Appendix H: Map Changes from Construction

In Figure 4.9, we show examples of object-centric map changes inside our TbV dataset, which we do not annotate and are not the focus of our work.

Appendix I: Additional Analysis of Map Change Frequency

In subsection 4.3.1 and Table 4.2 earlier in this chapter, we present an analysis of map change frequency. In this section, we provide additional analysis, an extended table, and derivations of our estimates. Map changes occur at random as part of a stochastic process. While some changes are coordinated at a city-administration level, it is still difficult to predict to a specific date or time when construction crews will complete changes. As discussed earlier in the chapter, we reason about square spatial areas of size 30 m $\times$ 30 m, which we refer to as tiles, which cover 900 m^2 each.
Derivation: Probability of an Encounter We consider the probability of entering a spatial area that has undergone a crosswalk or lane geometry within it. In other words, it is the probability of encountering a changed area, and thus we name it p_{eca}. In order to estimate the probability of encountering a changed area, rather than computing the ratio $\left(\frac{\text{num change-discovery miles}}{\text{num fleet miles}}\right)$, we compute the ratio $\left(\frac{\text{num. tiles where change is observed}}{\text{num. tiles entered by fleet}}\right)$. We do

Table 4.9: Across six particular cities, we analyze the probability of change for a $30m \times 30m$ spatial area. Since we can likely only catch changes for spatial areas that are somewhat frequently visited, we require that an area is visited by fleet at least $n = 5$ times. We provide $n = 1$ as well as a lower bound.

City Name	≥ 5 visits by fleet		≥ 1 visit by fleet	
	Probability of Change per tile	Up to T tiles in a thousand will change in 5 months	Probability of Change per tile	Up to T tiles in a thousand will change in 5 months
Pittsburgh	0.0068	7	0.0052	5
Detroit	0.0056	6	0.0049	5
Washington, D.C.	0.0046	5	0.0037	4
Miami	0.0038	4	0.0027	3
Austin	0.0009	0.9	0.0006	0.6
Palo Alto	0.0007	0.7	0.0006	0.6

not require that the autonomous vehicle directly drove over the changed tile, as an observed change can very well still affect driving behavior. We model the probability as a Bernoulli(p) r.v., with $p \approx 5.517 \times 10^{-5}$ across the more than 5 North American cities we analyze. A visit would occur once per every 18,124 times a vehicle enters such areas.

While the change percentage may seem inconsequential, one must consider that drivers in the United States are estimated to drive 3.225 trillion miles per year, according to the U.S. Department of Transportation [306]. If one were to consider our rate of change equal to the rate of change of any stretch of road within the United States, this would amount to an *upper bound* of 9B encounters of spatial areas with changed lane geometry or crosswalks, per year:

$$\frac{3.225 \cdot 10^{12}\text{miles}}{1 \text{ year}} \cdot \frac{1609 \text{ m}}{1 \text{ mile}} \cdot \frac{1 \text{ tile}}{30 \text{ m}} \cdot \frac{5.517 \cdot 10^{-5} \text{ changes}}{1 \text{ tile}} \approx 9.5B \tag{4.2}$$

This derivation assumes that all roads (including highways) are changed as often as urban roads (a generous estimate).

Derivation: Probability per Spatial Area We next estimate the probability of each unique tile in a city seeing a crosswalk or lane geometry change, which we also model as a Bernoulli(p) random variable, with p estimated as:

$$p = \frac{\text{\# unique changed tiles in city}}{\text{\# unique tiles in city visited at least n times by fleet}} \tag{4.3}$$

where the numerator and denominator are both measured over k months.

In Table 4.9, we analyze the probability of change for a $30m \times 30m$ spatial area across six particular cities. Since we can likely only catch changes for spatial areas that are somewhat frequently visited, we require that an area is visited by fleet at least $n = 5$ times over $k = 5$ months.

Appendix J: Synthetic Map Perturbation Technique

In subsection 4.4.2, Table 4.4, and Figure 4.4 earlier in this chapter, we enumerate a number of hand-designed priors we use to generate realistic-appearing synthetic maps. In this section, we provide detailed descriptions of the generation process.

J.1. Priors on the Crosswalk Perturbation Procedure

Our main observations from studying mapped data are that crosswalks are generally located near intersections, are orthogonal to lane segment tangents, and have little to no area overlap with other crosswalks. Accordingly, we first sample a random lane segment which will be spanned by the generated, synthetic crosswalk. We perform this random sampling from a biased but normalized probability distribution; lane segments within intersections achieve 4.5x the weight of non-intersection lane segments. In order to determine the orientation of the synthesized crosswalk's principal axis, we compute the normal to the centerline of the sampled lane segment at a randomly sampled waypoint. This waypoint is sampled from 50 waypoints that we interpolate along the centerline. We ensure that the sampled waypoint is not within the outermost 1/8 of pixels along any border of the rendered map image (i.e. within 15 m according to ℓ_∞ norm from the egovehicle). This measure is to allow some perturbation of the random crop for data augmentation, without losing visibility of the changed entity.

Next, in order to determine how many total lane segments the crosswalk must cross in order to span the entire road, we must determine the road extent. We approximate it as the union of all nearby lane segment polygons. The line representing the principal axis of the crosswalk may intersect with this road polygon in more than two locations, since it is often non-convex. We choose the shortest possible length segment that spans the road polygon to be valid, and thus find the closest two intersections to the sampled centerline waypoint. We randomly sample a crosswalk width w in meters from a normal distribution $w \sim \mathcal{N}(\mu = 3.5, \sigma = 1)$, but clip to the range $w \in [2, 4]$ meters afterwards, in accordance to our empirical observations of the real-world distribution.

If the rendered synthetic crosswalk has overlap with any other real crosswalk above a threshold of IoU $= 0.05$, we continue to sample until we succeed. The crosswalk is rendered as a rectangle, bounded between two long edges both extending along the principal axis of the crosswalk. We use alternating parallel strips of white and gray to color the object. Crosswalks are deleted by simply not rendering them in the rasterized image.

J.2. Lane Geometry Perturbation Procedure

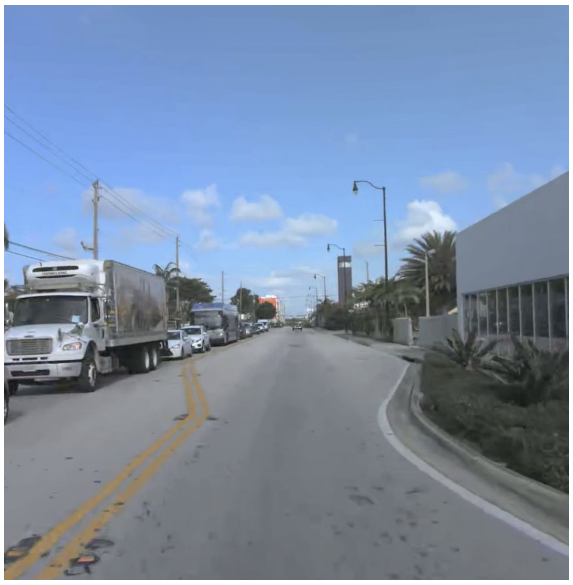
Our main observations from studying real-world map changes are that lane changes generally occur over a chain of lane segments, with combined length often over tens or hundreds of meters, although at times the combined length is far shorter. Accordingly, we use the directed lane graph to sample random connected sequences of lane segments, respecting valid successors. We then manipulate either the left or the right boundary only (not both) of this lane sequence.

Our general procedure is to start this sequence at a random lane well-within the field of view of the BEV image. As before, we ensure that the sampled marking is not entirely contained within the outermost 1/8 of pixels along any border of the rendered map image (i.e. within 15 m according to ℓ_∞ norm from the egovehicle).

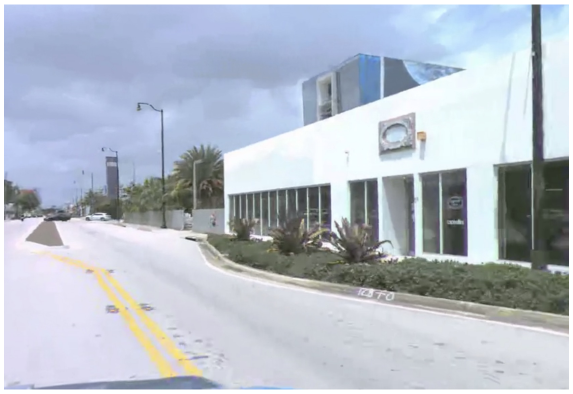
When deleting lane boundaries, we sample only painted yellow or white lane boundary markings. When changing the color or structure of lane boundaries, we sample lane boundary markings of any color (including those that are implicit). When adding a bike lane, we sample a sequence of 5 lane segments. For marking deletion and changes to lane marking color and structure, we sample a sequence of length 3.

We render these boundaries as colored polylines; we use red for implicit boundaries, and yellow and white for lane markings of their respective color. Lane boundary markings are deleted by simply not rendering them in the rasterized image.

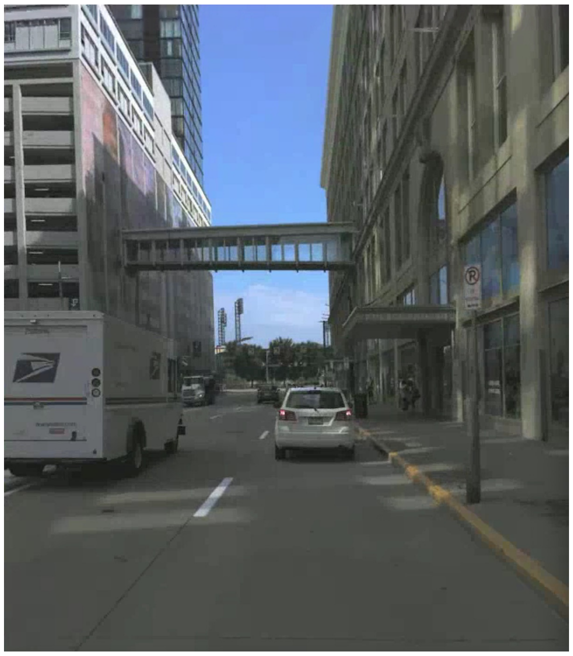
Bike lanes generally represent the rightmost lane in the United States. Accordingly, we synthesize a valid location for a new bike lane by iterating through the lane graph until there is no right neighbor; by dividing this rightmost lane into half, we can create two half-width lanes in place of one. We use solid white lines to represent their boundaries.



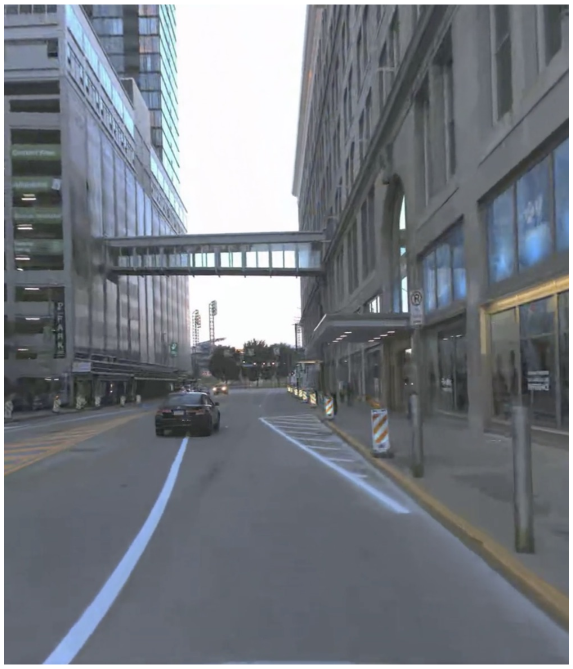
(a) Before



(b) After

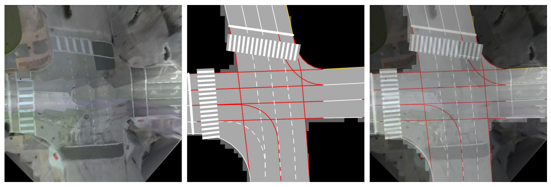


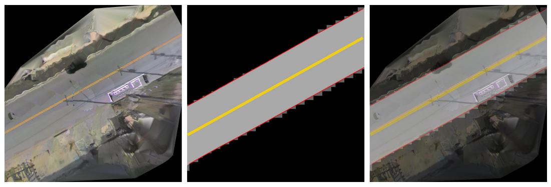
(c) Before

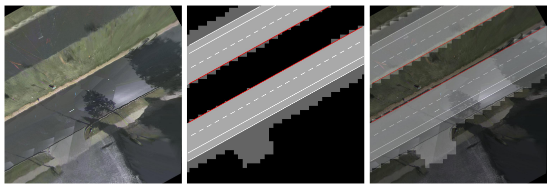


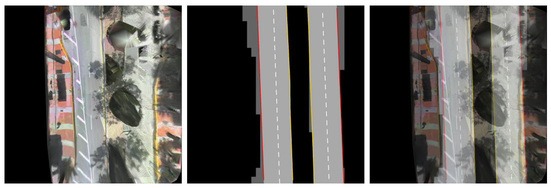
(d) After

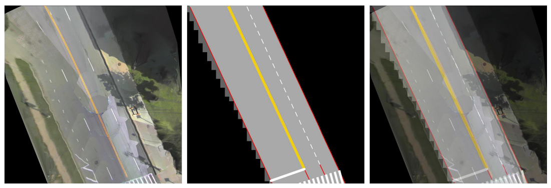
Figure 4.7: For a number of 'negative' logs, our TbV dataset includes corresponding logs captured before the map change was implemented, such that we obtain "before and after" imagery.

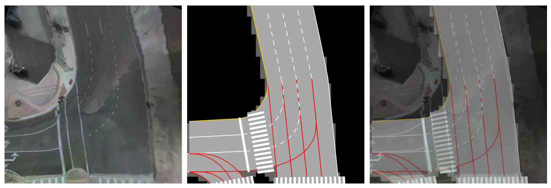

(a)

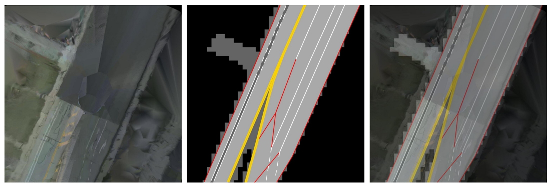

(b)


(c)


(d)


(e)


(f)


(g)

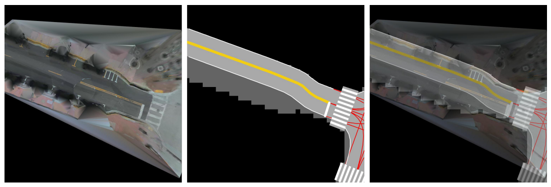
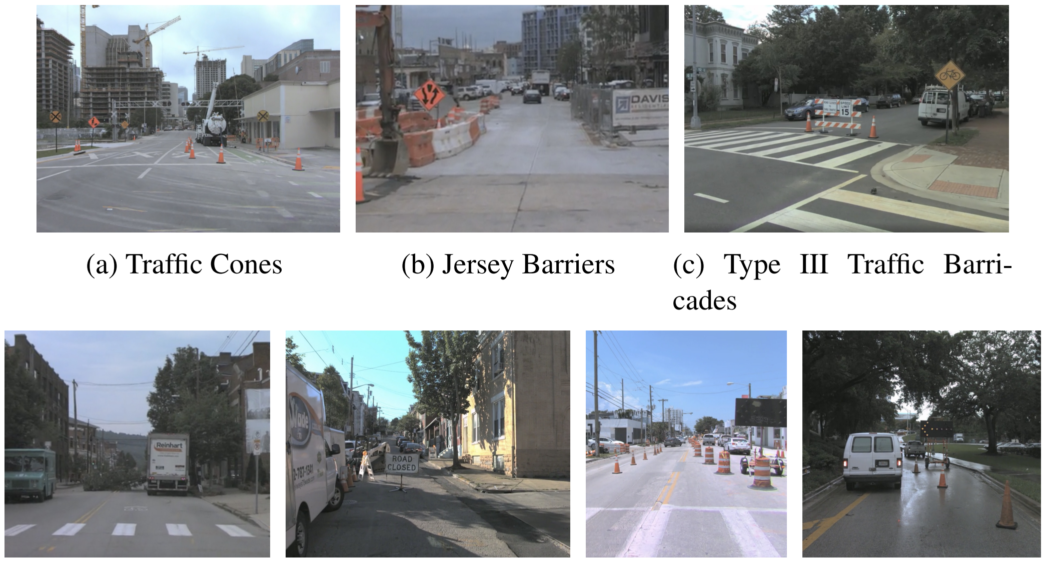

(h)

Figure 4.8: Examples from the test split of our TbV dataset. Left to right: BEV sensor representation, onboard map representation, blended map and sensor representations. Rows, from top to bottom: inserted crosswalks **(a)**, and painted lane geometry changes **(b-h)**.



(a) Traffic Cones (b) Jersey Barriers (c) Type III Traffic Barricades

(d) Fallen Trees (e) Construction Signs (f) Traffic Barrels/Drums (g) Arrowboard Trailers

Figure 4.9: Scenes with temporary object-related map changes collected in Argo AI's fleet data. Such scenes are not the focus of our work; rather, we believe such changes should be addressed by onboard object recognition systems.

CHAPTER 5
CONCLUSION

In this thesis, I have demonstrated how deep learning can unlock new capabilities for mapping, a crucial building block for spatial artificial intelligence.

In the indoor domain, we demonstrate dramatic improvements over the state of the art in completeness for 2d geometric indoor mapping, i.e. floorplan reconstruction. SALVe enables mapping with cheap hardware, few views, and very wide baselines, representing a significant breakthrough for virtual tours, architectural analysis, virtual staging, and autonomous navigation.

In the outdoor domain, our work exploring the "deep front-end" of SfM and SLAM demonstrates that the wide-baseline "correspondence problem" is far from solved in many real-world scenarios that arise in robotics applications, such as drone navigation and off-board scene reconstruction from drone photography. We introduce GTSFM, a new system for global SfM, to further analyze the benefits of a "deep front-end" on batch-mode SfM, and find that ensuing front-end signal-to-noise ratios are often insufficient to enable accurate batch-mode SfM on many real-world datasets.

Finally, in the domain of mapping for self-driving, I introduce learning-based formulations for solving the HD map change detection task in a bird's eye view and ego-view, demonstrating improvements over the state of the art in accuracy. Because real map changes are infrequent and vector maps are easy to synthetically manipulate, we lean on simulated data to train such models. Perhaps surprisingly, we show that such models can generalize to real world distributions. Our approach to building and validating HD maps can make such maps dynamic, an important missing part of mapping research today that can mitigate the danger associated with a host of unsafe scenarios for mobile robots and autonomous driving.

Along the way, in order to satisfy the demands of these data-driven, deep learning approaches, I contribute several large-scale datasets towards solving these problems via "programming by data" – the Argoverse 1.0 Datasets [4], the MSeg Dataset [325], the Trust but Verify (TbV) Dataset [120], and the Argoverse 2.0 Datasets [326].

5.1 Reflection and Lessons Learned

5.1.1 Contrasting Indoor and Outdoor Reconstruction

In this work, we have explored vastly different algorithms for indoor versus outdoor reconstruction. Indoor, man-made environments offer a large number of valuable priors that can simply reconstruction and remove degrees of freedom. For example, vertically aligned lines allow us to straighten panoramas; in an upright configuration, global pose estimation can be done in 2D, rather than in 3D. Furthermore, planar surfaces such flat floors and generally flat ceilings allow us to texture map onto orthographic views through backprojection. In additiona, common structures such as windows, doors, openings that determine connectivity in residential buildings dramatically simplify the search space for features;

for example, semantic features can be used instead of keypoint features for panoramic imagery, eliminating up to 3 orders of magnitude of matches, as we move from thousands of potential keypoint features per panorama, to a handful of door, window, and opening detections. Using traditional keypoint-based SfM reconstruction pipelines that were designed for narrow-f.o.v. outdoor imagery does not make sense for highly constrained indoor environments, captured by panoramas.

5.1.2 SLAM vs. SfM

Today's robotics research shows an emphasis on streams of data [327]. Mapping from diverse internet user photo collections has become a less and less common use case in computer vision [199, 208]. Nonetheless, wide-baseline still has a few unique use cases, for examples in human-driven image capture certain scenarios, such as indoor capture by real estate photographers. In other scenarios, SLAM a more frequent scenario for robotics; SLAM has many distinct advantages over SfM today, as its smaller baselines can dramatically simplify the correspondence problem, allowing the valuable use of photometric signals such as optical flow [327], and access to additional sensors such as IMU measurements further eliminate uncertainty.

5.2 Future Work

In this work, we have explored some of the current limits for building and validating maps, and shown avenues for progress via data-driven deep learning methods. However, the opportunities for future work in this domain are unbounded. Below, I share a few potentially fruitful ideas for exploration.

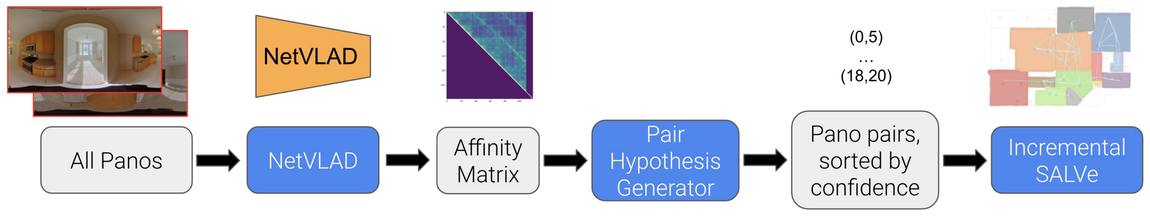


Figure 5.1: A possible incremental variant of SALVe. Rather than exhaustively evaluating $\mathcal{O}(n^2 k^2)$ alignments from n panoramas, each panorama containing k W/D/O's, one could sort the pairs by image similarity, and keep any alignment if SALVe's confidence estimates exceeds a certain threshold.

5.2.1 Accelerating Semantic SfM Indoors

In SALVe, I introduced a new semantic SfM algorithm for automatic floorplan reconstruction from sparse panoramas. However, our implementation of SALVe is largely based around exhaustive enumeration of all possible hypotheses, leading to difficult computational complexity. However, this matching is embarrassingly parallel, and could be performed in a smart and efficient incremental fashion. For example, the pose graph could be grown by selecting image pairs by relative likelihood, for example via pairwise image similarity using techniques such as NetVLAD [328] (see Figure 5.1).

5.2.2 End-to-End Optimization via Differentiable Rendering

In this work, we demonstrated how 2 separately trained deep networks – HorizonNet [117] and SALVe – can be used to first determine adjacency of a pose graph and initialize it, prior to optimization. This was done because the adjacency matrix is unknown a priori, and requires hard choices about which edges are reliable and should be thrown out. However, in future work, these choices could be made by estimating weights on the edges, in order to leverage differentiable rendering and differentiable alignment, like UnsupervisedR&R [329]. One potential loss would be directly on the camera rotations and translations, using backpropagation on transformation groups, using tools such as LieTorch [330]. This would allow us to learn the entire system end-to-end (see Figure 5.2).

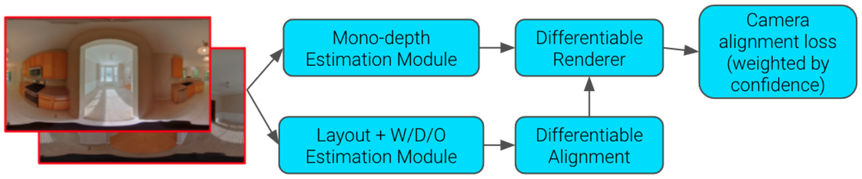

Mono-depth Estimation Module
Differentiable Renderer
Camera alignment loss (weighted by confidence)
Layout + W/D/O Estimation Module
Differentiable Alignment

Figure 5.2: A potential architecture for an end-to-end trainable floorplan reconstruction system.

5.2.3 Semantic SfM: Joint Optimization

Another key opportunity for floorplan reconstruction is to incorporate landmarks into the global optimization of window, door, and opening features, rather than only optimizing a relative pose graph. Not only do these landmarks represent crucial pieces of information for a floorplan, but they may also be used to further improve accuracy.

5.2.4 Learning for SfM

While dramatic progress is being made for multi-view stereo, largely accelerated by NeRF [217], learning-based methods for camera geometry estimation still lag behind. Even state-of-the art methods such as SuperGlue [76] and LoFTR [82] can fail spectacularly under repetitive features (see Figure 5.3). A critical missing capability today for deep features is knowing when to trust them. Using learning to estimate this trustworthiness could be used to eliminate heavily engineered methods in track management and outlier rejection, such as those found in COLMAP [216]. A popular open-source tool, `cloc`, counts 304,993 lines of code in the COLMAP repository, scattered across 681 files. Fundamental ways of re-thinking the learning formulation may be required. Replacing these lines in a "Software 2.0" framework, with a single model or multiple trained models, represents a grand challenge of vision research. A bitter lesson of SfM research is that decades of research have yet to yield methods that can work with human-level robustness; under wide baselines or repetitive structures, modern deep-feature matching techniques do not approach, let alone surpass, a human-level ability to identify correspondences between two images (see Figure 5.3). Why is this? The `programming by data` paradigm suggests that the solution lies in massively scaling up training datasets, and transitioning fully to attention-like architectures.

In Global SfM, a key opportunity is understanding which relative pose measurements are trustworthy, and which represent false positives from over-confident matches on pairs with no true covisibility. There are many features available per edge – inlier ratios, cycle errors, inlier counts, point features, point errors – and more. Graph neural networks such as Transformers [331] could be used to aggregate these features to discard erroneous edges, or instead to learn other types of features. While Phillips and Daniilidis [237] explore this direction for the keypoint match graph, there are many opportunities to explore this direction on the relative pose graph.

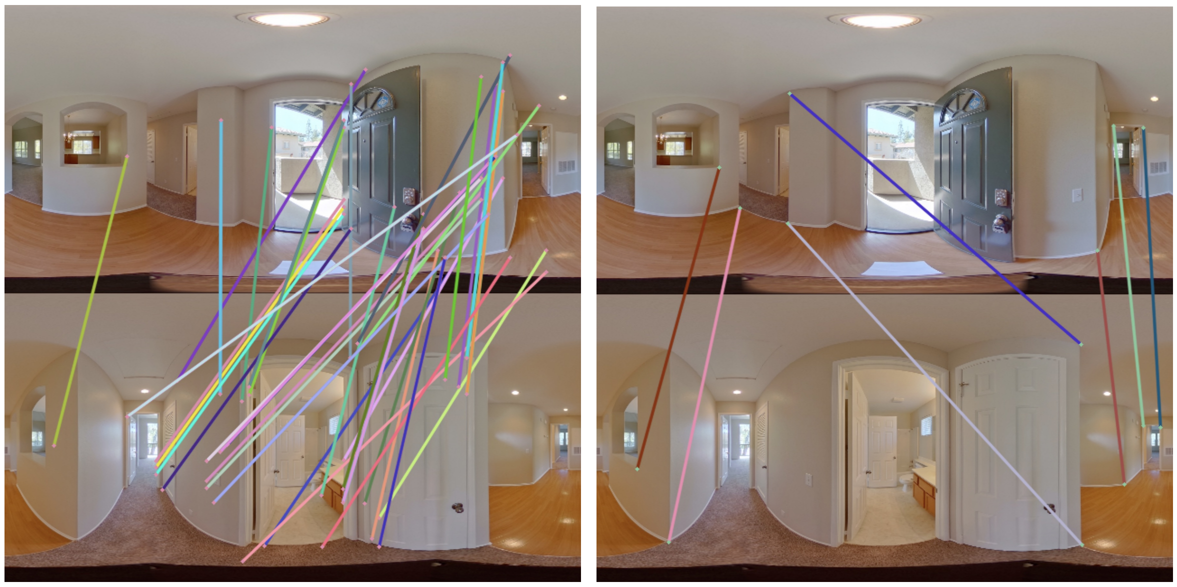

(a) LoFTR (b) Ground Truth

Figure 5.3: **(Left)**: 50 most confident keypoint matches from LoFTR [82], a state-of-the-art image matching system, on an image pair from ZInD [14]. **(Right)**: Ground-truth (hand-annotated) keypoint matches for the same image pair. 50 out of 50 LoFTR matches are incorrect, each with estimated confidence over 0.9.

5.2.5 Mapping with SfM + MVS

Exciting new works in real-time neural radiance fields [246], and large-scale NERF [332] especially for entire city blocks, suggest that classical methods for Multi-view stereo will become obsolete. Dramatic additional improvements are needed to lessen the requirement for the number of input images, 3M images. However, obtaining accurate camera poses across wide baselines is still very challenging, suggesting that visual inertial odometry (VIO) and LiDAR-based pose estimation methods will stand the test of time. No global

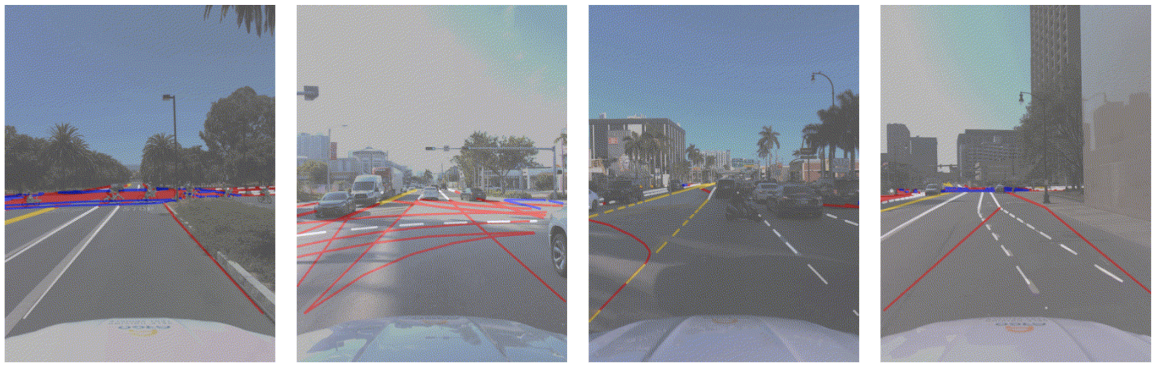


Figure 5.4: Annotated HD maps overlaid on front-center camera imagery captured in the Argoverse 2.0 Sensor Dataset.

SfM approach will be successful without some incremental reasoning, as the raw signal-to-noise ratio even for learned descriptors and matchers remains limited.

5.2.6 Improving the Accuracy of Map Change Detection

In this thesis, we have largely explored image-centric map change detection algorithms. However, LiDAR centric learning-based map-change detection is also a promising approach. In addition, we have experimented with accumulating sensor data over the temporal dimension in the bird's eye view, but not in the ego-view. Using a buffer of data in the ego-view could further improve its accuracy, and there are many potential architectures that could be explored.

5.2.7 Scalable HD Map Creation and Maintenance: Looking to the future

All who are familiar with self-driving know the immense logistical challenges associated with long-term maintenance of HD maps. The tedious nature of maintaining maps as the world changes around us suggests we should invest further into techniques for automatic map updates, albeit with a trained annotator who can certify the final state of the map before placed into production. Promising new architectures [108, 109, 110] suggest a path forward for an order of magnitude increase in automation. Training such algorithms, however, requires tapping into the "programming by data" paradigm. Our new datasets – the Argoverse 2.0 Sensor Dataset, Argoverse 2.0 LiDAR Dataset, and the *Trust but Verify* Dataset – provide tens of thousands of vehicle logs with high-quality HD map annotations in 3D to make this possible (see Figure 5.4).

Appendices

APPENDIX A
GEOMETRY FUNDAMENTALS

A.1 Lie Groups and Algebras

Lie groups are essential and helpful tools for modeling rigid body kinematics and transformations. Rigid bodies have a state which consists of position and orientation. When sensors are placed on a rigid body (e.g. a robot), they provide measurements in the body frame. Suppose we wish to take a measurement y_b from the body frame and move it to the world frame, yielding y_w. We can do this via left multiplication with a transformation matrix ${}^wT_b$, a member of the matrix Lie groups, that transports the point from one space to another space: $y_w = {}^wT_b\, y_b$.

Group Definition

A group is a set G, with an operation $\circ$ of (binary) multiplication on elements of G which is:

1. closed: If $g_1, g_2 \in G$ then also $g_1 \circ g_2 \in G$.
2. associative: $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$, for all $g_1, g_2, g_3 \in G$;
3. unit element e: $e \circ g = g \circ e = g$, for all $g \in G$;
4. invertible: For every element $g \in G$, there exists an element $g^{-1} \in G$ such that $g \circ g^{-1} = g^{-1} \circ g = e$.

Lie Groups

When we are working with pure rotations, we work with Special Orthogonal groups, $SO(\cdot)$. When we are working with a rotation and a translation together, we work with Special Euclidean groups $SE(\cdot)$.

Lie Groups are unique because they are both a group *and* a manifold. They are continuous manifolds in high-dimensional spaces, and have a group structure.

The Special Orthogonal Group SO(N)

Membership in the Special Orthogonal Group $SO(N)$ requires two matrix properties: $R^T R = I$ and $\det(R) = +1$.

This gives us a very helpful property: $R^{-1} = R^T$, obtaining the matrix's inverse is as simple as taking its transpose. We will generally work with $SO(N)$, where $N = 2, 3$, meaning the matrices are rotation matrices $R \in \mathbf{R}^{2\times 2}$ or $R \in \mathbf{R}^{3\times 3}$.

These rotation matrices R are not commutative.

Two-dimensional Special Orthogonal Group SO(2)

$SO(2)$ is a 1-dimensional manifold living in a 2D Euclidean space, e.g. moving around a circle. In other words, $SO(2)$ is the space of orthogonal matrices that corresponds to rotations in the plane. We will be stuck with singularities if we use 2 numbers to parameterize it, which would mean kinematics break down at certain orientations.
A simple example: Let's move from the body frame b to a target frame t:

$$P_t = {}^tR_b(\theta)P_b$$

$$\begin{bmatrix} 5\cos(\theta) \\ 5\sin(\theta) \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

As described by Lavalle in [333], another way to think of this is to consider that a robot can be rotated counterclockwise by some angle $\theta \in [0, 2\pi)$ by mapping every (x, y) as:

$$(x, y) \rightarrow (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta). \tag{A.1}$$

SO(3)

There are several well-known parameterizations of $R \in SO(3)$:

1. $R \in \mathbb{R}^{3\times3}$ full rotation matrix, 9 parameters – there must be 6 constraints.
2. Euler angles, e.g. (ϕ, θ, ψ), so 3 parameters.
3. Angle-Axis parameters $(\vec{a}, \phi)$, which is 4 parameters and 1 constraint (unit length).
4. Quaternions (q_0, q_1, q_2, q_3), 4 parameters and 1 constraint (unit length).

There are only 3 degrees of freedom in describing a rotation. But this object doesn't live in 3D space. It is a 3D manifold, embedded in a 4-D Euclidean space. Parameterizations 1,3,4 are overconstrained, meaning they employ more parameters than we strictly need. With overparameterized representations, we have to do extra work to make sure we satisfy the constraints of the representation. As it turns out, $SO(3)$ cannot be parameterized by only 3 parameters in a non-singular way.

Rotation Matrices

3D rotation matrices do not form a vector space. An easy way to see this is to try to add the following two rotation matrices, I and R, where R is a 180° rotation about the z-axis:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$I + R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{A.2}$$

which is not a rotation (as it squashes flat the x- and y- components).

Euler Angles

One parameterization of $SO(3)$ is to imagine three successive rotations around different axes. The Euler angles encapsulate yaw-pitch-roll: first, a rotation about the x-axis by ϕ (roll). Then, a rotation about the pitch axis by θ (via right-hand rule), and finally we perform a yaw via a rotation about the z-axis (yaw, ψ).

The sequence of successive rotations is encapsulated in ${}^{w}R_b$:

$$ {}^{w}R_b = R_z(\psi)R_y(\theta)R_x(\phi) \tag{A.3}$$

As outlined by Lavalle [333], these successive rotations by (ψ, θ, ϕ) are defined by:

$$R_{yaw} = R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$R_{pitch} = R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{A.5}$$

$$R_{roll} = R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{A.6}$$

Each rotation matrix $\in \mathbb{R}^{3\times3}$ above is a simple extension of the 2D rotation matrix from $SO(2)$. For example, the yaw matrix R_{yaw} performs a 2D rotation with respect to the x and y coordinates while leaving the z coordinate unchanged [333].

The Two-Dimensional Special Euclidean Group SE(2)

The real space $SE(2)$ are 3×3 matrices, moving a point in homogenous coordinates to a new frame. It is important to remember that this represents a rotation followed by a translation (not the other way around). A rigid body which translates and rotates on a 2D plane has 3 DOF, e.g. a ground robot.

$$T = \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \begin{bmatrix} R_{2\times2} & & t_{2\times1} \\ & \ddots & \vdots \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix} \tag{A.7}$$

By adding an extra dimension to the input points and transformation matrix T, the translational part of the transformation is absorbed.

The Three-Dimensional Special Euclidean Group SE(3)

The set of rigid body motions, or special Euclidean transformations, is a (Lie) group, the so-called special Euclidean group, typically denoted as SE(3). The real space $SE(3)$ is a

6-dimensional manifold. Its dimensions is exactly the number of degrees of freedom of a free-floating rigid body in space [3]. $SE(3)$ can be parameterized with a 4×4 matrix as follows:

$$\begin{bmatrix} & R_{3\times 3} & & t_{3\times 1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What is the inverse of an SE(3) object? Consider a transformation of a point in the body frame p_b to a point in the world frame p_w. Both points p_b, p_w must be in homogeneous coordinates. We can invert it as follows:

$$\begin{aligned} p_w &= {}^wT_b p_b \\ p_w &= \begin{bmatrix} {}^wR_b & \mid & {}^wt_b \end{bmatrix} p_b \\ p_w &= {}^wR_b p_b + {}^wt_b \\ p_w - {}^wt_b &= {}^wR_b p_b \\ ({}^wR_b)^{-1}(p_w - {}^wt_b) &= ({}^wR_b)^{-1w}R_b p_b \\ ({}^wR_b)^T(p_w - {}^wt_b) &= p_b \\ ({}^wR_b)^T p_w - ({}^wR_b)^{Tw}t_b &= p_b \\ \begin{bmatrix} ({}^wR_b)^T & \mid & -({}^wR_b)^{Tw}t_b \end{bmatrix} p_w &= p_b \\ p_b &= \begin{bmatrix} ({}^wR_b)^T & \mid & -({}^wR_b)^{Tw}t_b \end{bmatrix} p_w \\ p_b &= {}^wT_b^{-1} p_w \end{aligned} \tag{A.8}$$

Thus if $T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$, then $T^{-1} = \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix}$.

A.1.1 Axis-Angle Representation, Lie Algebra, and Exponential and Log Maps

One of the most powerful representations of a 3d rotation is the axis-angle representation, that will allow us to compute averages of multiple rotation matrices, smoothly interpolate between two rotations, and much more. These tasks are hard to perform directly because rotation matrices don't form a vector space—for instance, adding two rotation matrices doesn't give you another rotation matrix.

We can convert rotation matrices into an axis-angle form where rotations are represented by ordinary vectors.The axis-angle representation consists of an angle $\theta \in \mathbb{R}$ and an axis $\mathbf{u} \in \mathbb{R}^n$. The direction of the vector gives the axis of rotation, and the magnitude of the vector gives the angle of the rotation. They define a type of "screw" motion (spinning around an axis).

We know that rotation matrices form a Lie group $SO(n)$, and the associated vector space is the Lie algebra $\mathfrak{so}(3)$.

Logarithmic Map Mapping from manifold to tangent space: $\log : SO(3) \rightarrow \mathfrak{so}(3)$. The *logarithmic map* can be used to turn a rotation matrix $\mathbf{R}$ into an axis and angle.

Exponential Map Mapping from tangent space to manifold: $\exp : \mathfrak{so}(3) \rightarrow SO(3)$. The *exponential map* can be used to turn an angle θ and unit-length axis $\mathbf{u}$ into a rotation matrix $\mathbf{R}$.

The Exponential Map

The matrix exponential

$$e^{\hat{\omega}t} = I + \hat{\omega}t + \frac{(\hat{\omega}t)^2}{2!} + \cdots + \frac{(\hat{\omega}t)^n}{n!} + \cdots \tag{A.9}$$

defines a map from the space $\mathfrak{so}(3)$ to $SO(3)$, which we often call the exponential map [334].

For any rotation matrix $R \in SO(3)$, there exists a $\omega \in \mathbb{R}^3$, $\|\omega\| = 1 and t \in \mathbb{R}$ such that $R = e^{\hat{\omega}t}$. This theorem is quite powerful: it means that any rotation matrix can be realized by rotating around some fixed axis by a certain angle. This map is not one-to-one.

Exponential Map Application: Interpolation of Rotation Matrices

A problem that arises in computer vision and robotics is interpolation of rigid body motions.

We can smoothly interpolate 3d rotations using the exponential and logarithmic maps for 3D rotations. Euler angles are an inferior choice for interpolation. Given two rotations $\mathbf{R}_0$ and $\mathbf{R}_1$, the smallest rotation between them (in axis-angle form) is given by

$$\boldsymbol{\omega} = \log(\mathbf{R}_1 \mathbf{R}_0^{-1}) \tag{A.10}$$

Hence, we can interpolate via $\mathbf{R}(t) = \exp(t\boldsymbol{\omega})\mathbf{R}_0$.

As before, this family of rotations starts at $t = 0$ with $\mathbf{R}_0$, and interpolates to $\mathbf{R}_1$ at $t = 1$ with the minimal amount of additional "twisting" in-between.

A.1.2 Log / Exponential Map Application: Rotation Averaging

Another problem that arises in computer vision and robotics is single rotation averaging.

The Single Rotation Averaging problem is defined as follows: given a collection of rotation matrices $\mathbf{R}_1, \dots, \mathbf{R}_n \in \mathbb{R}^{3\times 3}$, find the average rotation $\bar{\mathbf{R}}$. However, we can't just take an average of the matrices $\frac{1}{n}\sum_{i=1}^{n} \mathbf{R}_i$, since the average of rotation matrices will not in general be a rotation matrix.

Analogy: Averaging Points. Instead, let's first think about an unusual way to average a bunch of points $\mathbf{x}_1, \dots, \mathbf{x}_n$ in the 2d plane. Instead of just directly taking the average (which we can't do with rotations), we'll pick some initial guess $\bar{\mathbf{x}}$ for the average. We'll then compute, for each point, the smallest vector $\mathbf{u}_i$ that takes us from $\bar{\mathbf{x}}$ to $\mathbf{x}_i$. In the case of points in the 2d plane, this vector is just $\mathbf{u}_i = \mathbf{x}_i \bar{\mathbf{x}}$. We'll then compute the average vector $\mathbf{u} := \frac{1}{n}\sum_{i=1}^{n} \mathbf{u}_i$ that tries to pull us toward all the points, and take a little step in this direction of size $\tau \in [0, 1]$. If we reach a point where $\mathbf{u} = 0$ – or at least, below some very small $\epsilon > 0$—then the algorithm stops and we know we've reached the average (see Algorithm 2).

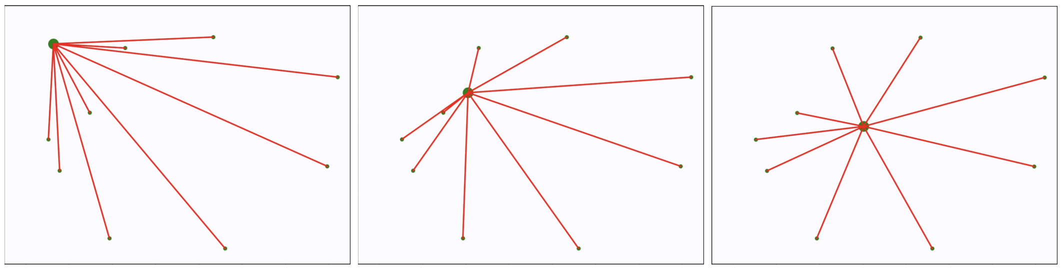


Figure A.1: Iterations towards convergence of the Weiszfeld algorithm. **(Left)**: (1) initialization. **(Center)**: intermediate iteration. **(Right)**: convergence.

Algorithm 2 Weiszfeld Algorithm

```
**Inputs:**
$\{\mathbf{x}_i\}_{i=1}^{N}$: Point measurements.

**Output:**
$\bar{\mathbf{x}}$: Average point.

**Solution:**
Pick an initial guess $\bar{\mathbf{x}} \in \mathbb{R}^2$
**while** true **do**
    $\mathbf{u}_i \leftarrow \mathbf{x}_i\bar{\mathbf{x}}$
    $\mathbf{u} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \mathbf{u}_i$
    $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \tau\mathbf{u}$
    **if** $\|\mathbf{u}\| > \epsilon$ **then**
        break
    **end if**
**end while**
```

Single Rotation Averaging.

The algorithm for averaging rotations is nearly identical (see Algorithm 3).

The initial guess could be any rotation matrix – for example, the identity $\bar{\mathbf{R}} = I$. Rather than computing differences of rotation matrices, we compute the smallest rotation from $\bar{\mathbf{R}}$ to each of the $\mathbf{R}_i$ via the log map, yielding a bunch of skew-symmetric matrices $\boldsymbol{\omega}_i$.

Since these matries belong to a common vector space, we can average them to produce another skew-symmetric matrix, i.e., another axis-angle representation of a rotation. Applying the exponential map moves $\bar{\mathbf{R}}$ a little bit toward the average, and we repeat until convergence.

This algorithm is essentially known as the *Weiszfeld algorithm*, and the notion of average we get in the end is referred to as the *Karcher mean* of the rotations. Multiple rotation averaging uses the same sorts of ideas, as can be seen in [335] or [119].

Algorithm 3 Single Rotation Averaging

```
Inputs:
$\{\mathbf{R}_i\}_{i=1}^{N}$: Rotation measurements.

Output:
$\bar{\mathbf{R}}$: Mean rotation.

Solution:
Pick an initial guess $\bar{\mathbf{R}} \in \mathbb{R}^{3\times 3}$
while true do
    $\boldsymbol{\omega}_i \leftarrow \log\big(\mathbf{R}_i \bar{\mathbf{R}}^{-1}\big)$
    $\boldsymbol{\omega} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\omega}_i$
    $\bar{\mathbf{R}} \leftarrow \exp(\tau\, \boldsymbol{\omega}) \bar{\mathbf{R}}$
    if $\|\boldsymbol{\omega}\| > \epsilon$ then
        break
    end if
end while
```

Twists

A 4×4 matrix of the form $\hat{\xi}$ is called a twist. The set of all twists is denoted as $\mathfrak{se}(3)$ [336, 334, 337]:

$$\mathfrak{se}(3) = \left\{ \hat{\xi} = \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} \mid \omega \in \mathfrak{so}(3), v \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times 4} \tag{A.11}$$

$\mathfrak{se}(3)$ is called the tangent space (or Lie algebra) of the matrix group $SE(3)$.

Why do we care about twists? It turns out that a rigid body can be moved from one position to any other by a movement consisting of (1) a rotation about a straight line (2) followed by a translation parallel to that line. This type of motion is *screw motion*, and its infinitesimal version is called a *twist*. The beauty of a twist is that it describes the instantaneous velocity of a rigid body in terms of its linear and angular components, i.e. the linear velocity v and angular velocity ω [334]. It is the matrix exponential that maps a twist into its corresponding screw motion.

A.2 Epipolar Geometry

Epipolar geometry is the geometry of stereo vision. In this section, we will review the two-view case. Many modern approaches for solving robotic computer vision tasks such as Structure from Motion (SfM) and Visual Simultaneous Localization and Mapping (VS-LAM) rely heavily on "feature matching," or the ability to find accurate keypoint correspondences between pairs of images. Tools from Epipolar geometry are a simple, and often effective, way to discard outliers in feature matching and are widely used.

A.2.1 The Essential Matrix

Given point correspondences $\{(\mathbf{x}_0, \mathbf{x}_1)\}$ respectively from two images I_0, I_1, and camera intrinsics K, the 5-point algorithm solves for an Essential matrix ${}^1\mathbf{E}_0$:

$$\begin{bmatrix} \mathbf{x}_1 \\ 1 \end{bmatrix} K^{-T} \left({}^1\mathbf{E}_0 \right) K^{-1} \begin{bmatrix} \mathbf{x}_0 \\ 1 \end{bmatrix} = 0 \tag{A.12}$$

This equation can be derived from first principles if we consider a 3D point $\mathbf{p}$ being viewed from two cameras (see Longuet-Higgins [272] or Szeliski [338], p. 704):

$$d_1 \hat{\mathbf{x}}_1 = \mathbf{p}_1 = {}^1\mathbf{R}_0 \mathbf{p}_0 + {}^1\mathbf{t}_0 = {}^1\mathbf{R}_0 (d_0 \hat{\mathbf{x}}_0) + {}^1\mathbf{t}_0 \tag{A.13}$$

where $\hat{\mathbf{x}}_j = \mathbf{K}_j^{-1} \mathbf{x}_j$ are the (local) ray direction vectors. Note that ${}^1\mathbf{R}_0$ and ${}^1\mathbf{t}_0$ define an SE(3) object that transforms $\mathbf{p}_0$ from camera 0's frame to camera 1's frame. We'll refer to these just as $\mathbf{R}$ and $\mathbf{t}$ for brevity in the following derivation.

We can eliminate the $+\mathbf{t}$ term by a cross-product. This can be achieved by multiplying with a skew-symmetric matrix as $[\mathbf{t}]_\times \mathbf{t} = 0$. Then:

$$d_1 [\mathbf{t}]_\times \hat{\mathbf{x}}_1 = d_0 [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_0. \tag{A.14}$$

Swapping sides and taking the dot product of both sides with $\hat{\mathbf{x}}_1$ yields

$$d_0 \hat{\mathbf{x}}_1^T ([\mathbf{t}]_\times \mathbf{R}) \hat{\mathbf{x}}_0 = d_1 \hat{\mathbf{x}}_1^T [\mathbf{t}]_\times \hat{\mathbf{x}}_1 = 0, \tag{A.15}$$

Since the cross product $[\mathbf{t}]_x$ returns 0 when pre- and post-multiplied by the same vector, we arrive at the familiar epipolar constraint, where $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$:

$$\hat{\mathbf{x}}_1^T \mathbf{E} \hat{\mathbf{x}}_0 = 0 \tag{A.16}$$

Recovering relative camera motion In order to recover $(\mathbf{R}, \mathbf{t})$ from $\mathbf{E}$, we must verify possible pose hypotheses by triangulating 3d points from each 2-view correspondence, and by checking the cheirality of each 3d point (whether 3D points have positive depth). We cannot recover the SE(3) object ${}^1T_0 = ({}^1\mathbf{R}_0, {}^1\mathbf{t}_0)$ from the decomposed E matrix, as ${}^1\mathbf{t}_0$ provides a unit translation direction, not an actual translation direction with any meaningful magnitude or scale.

A.2.2 The Fundamental Matrix and DLT

Suppose we don't have access to camera intrinsic matrices K_0, K_1. The Fundamental matrix allows us still to verify correspondences. We can use Equation A.12, and define $\mathbf{F} = K_1^{-T} ({}^1\mathbf{E}_0) K_0^{-1}$, leaving a new epipolar constraint of $\mathbf{x}_1^T F \mathbf{x}_0 = 0$, where $\mathbf{x}_0, \mathbf{x}_1$ are 2D corresponding points in images I_0, I_1. To simplify notation in a later system of equations , we'll use slightly different notation: let $\mathbf{x}_0 = (u, v)$ and $\mathbf{x}_1 = (u', v')$:

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \tag{A.17}$$

Longuet-Higgins' 8-Point Algorithm [272] provides the solution for estimating F if at least 8 point correspondences are provided. A system of linear equations is formed as follows:

$$Af = \begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & u_nv_n' & u_n & v_nu_n' & v_nv_n' & v_n & u_n' & v_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{A.18}$$

The matrix-vector product above can be driven to zero by minimizing the norm, and avoiding the degenerate solution that $x = 0$ with a constraint that the solution lies upon the unit ball, e.g.

$$\underset{\|x\|=1}{\text{minimize}} \quad \|Ax\|_2^2 = x^T A^T A x = x^T B x \tag{A.19}$$

By the Courant-Fisher characterization, it is well known that if B is a $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and corresponding eigenvectors $v_1, \dots, v_n$, then

$$v_n = \text{arg} \min_{\|x\|=1} x^T B x \tag{A.20}$$

meaning the eigenvector associated with the smallest eigenvalue λ_n of $B = A^T A$ is the solution $x^\star$, which is equivalent to the smallest right singular vector v from the SVD of $A = U\Sigma V^T$. The vector $x^\star$ contains the 9 entries of the Fundamental matrix $F^\star$.

DLT Proof: the SVD provides the solution

The proof is almost always taken for granted, but we will provide it here for completeness. This is a specific instance of the extremal trace [339] (or trace minimization on a unit sphere) problem, with $k = 1$, i.e.

$$\begin{aligned} &\text{minimize} && \text{Tr}(X^T B X) \\ &\text{subject to} && X^T X = I_k \end{aligned} \tag{A.21}$$

where I_k denotes the $k \times k$ identity matrix. The unit ball constraint avoids the trivial solution when all eigenvalues λ_i are zero instead of a single zero eigenvalue.

In our case, since U, V are orthogonal matrices (with orthonormal columns), then $U^T U = I$. Thus, the SVD of B yields

$$A^T A = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma U^T U \Sigma V^T = V\Sigma^2 V^T. \tag{A.22}$$

Since $B = A^T A$, B is symmetric, and thus the columns of $V = \begin{bmatrix} v_1 \dots v_n \end{bmatrix}$ are eigenvectors of B. V can equivalently be computed with the SVD of A or B, since V appears in both decompositions: $A = U\Sigma V^T$ and $B = V\Sigma^2 V^T$.

Because B is symmetric, there exists a set of n orthonormal eigenvectors, yielding an eigendecomposition $B = V^T \Lambda V$. Thus,

$$\arg\min_{\|x\|=1} \quad x^T B x = \arg\min_{\|x\|=1} \quad x^T V^T \Lambda V x = \arg\min_{\|x\|=1} \quad (Vx)^T \Lambda (Vx) \tag{A.23}$$

Since V is orthogonal, $\|Vx\| = \|x\|$, thus minimizing $(Vx)^T\Lambda(Vx)$ is equivalent to minimizing $x^T\Lambda x$. Since Λ is diagonal, $x^T B x = \sum\limits_{i=1}^{n} \lambda_i x_i^2$ where $\{\lambda_i\}_{i=1}^n$ are the eigenvalues of B. Let $q_i = x_i^2$, meaning $q_i \geq 0$ since it represents a squared quantity. Since $\|x\| = 1$, then $\sqrt{\sum\limits_i x_i^2} = 1$, $\sum\limits_i x_i^2 = 1$, $\sum\limits_i q_i = 1$. Thus,

$$\min_{\|x\|=1} x^T B x = \min_{\|x\|=1} \sum_{i=1}^{n} \lambda_i x_i^2 = \min_{q_i} \sum_i \lambda_i q_i = \min_{q_i} \lambda_n \sum_i q_i = \lambda_n \tag{A.24}$$

where λ_n is the smallest eigenvalue of B. The last line follows since $q_i \geq 0$ and $\sum\limits_i q_i = 1$, therefore we have a convex combination of a set of numbers $\{\lambda_i\}_{i=1}^n$ on the real line. By properties of a convex combination, the result must lie in between the smallest and largest number. Now that we know the minimum is λ_n, we can obtain the argmin by the following observation:

If v is an eigenvector of B, then

$$Bv = \lambda_n v \tag{A.25}$$

Left multiplication with v^T simplifies the right side because $v^T v = 1$, by our constraint that $\|x\| = 1$. We find:

$$v^T(Bv) = v^T(\lambda_n v) = v^T v \lambda_n = \lambda_n \tag{A.26}$$

Thus the eigenvector v associated with the eigenvalue λ_n is $x^\star$. $\square$

A.2.3 Normalized 8-Point Algorithm

Hartley [175] proposes a simple modification to the classical 8-Point algorithm to make it robust to noise. By preceding the algorithm with a very simple normalization (translation and scaling) of the coordinates of the matched points, results are obtained comparable with the best iterative algorithms.

The main idea is to replace coordinates $\mathbf{u}_a$ in image a with $\hat{\mathbf{u}}_a = T_a \mathbf{u}_a$, and coordinates $\mathbf{u}_b$ in image b with $\hat{\mathbf{u}}_b = T_b \mathbf{u}_b$.

If T is chosen to be invertible, then we can recover the original coordinates from the transformed ones, as

$$\begin{aligned} \hat{\mathbf{u}}_a &= T_a \mathbf{u}_a \\ T_a^{-1}\hat{\mathbf{u}}_a &= T_a^{-1} T_a \mathbf{u}_a \\ T_a^{-1}\hat{\mathbf{u}}_a &= \mathbf{u}_a \end{aligned} \tag{A.27}$$

Substituting in the equation $\mathbf{u}_b^T F \mathbf{u}_a = 0$, we derive the equation

$$\begin{aligned} \mathbf{u}_b^T F \mathbf{u}_a &= 0 \\ (T_b^{-1}\hat{\mathbf{u}}_b)^T F T_a^{-1} \hat{\mathbf{u}}_a &= 0 \\ \hat{\mathbf{u}}_b^T T_b^{-T} F T_a^{-1} \hat{\mathbf{u}}_a &= 0 \end{aligned} \tag{A.28}$$

If we use the normalized points $\mathbf{u}_a, \mathbf{u}_b^T$ when fitting the Fundamental matrix, then we will end up estimating $\hat{F} = T_b^{-T} F T_a^{-1}$. In other words, $\mathbf{u}_b^T F \mathbf{u}_a = \hat{\mathbf{u}}_b^T \hat{F} \hat{\mathbf{u}}_a$. If we want to find out the original F that corresponded to raw (unnormalized) point coordinates, than we need to transform backwards:

$$\begin{aligned} \hat{F} &= T_b^{-T} F T_a^{-1} \\ T_b^T \hat{F} &= T_b^T T_b^{-T} F T_a^{-1} \\ T_b^T \hat{F} T_a &= F T_a^{-1} T_a \\ T_b^T \hat{F} T_a &= F \end{aligned} \tag{A.29}$$

A.3 3d Geometry for Panoramas

Panorama

A panoramic image offers a $360°$ view of the area. Panoramas are often mapped to an image that covers a $360°$ field-of-view horizontally, but only $180°$ vertically (see Figure A.2). This is often what we are used to when seeing a map of the world, as if a globe had been distorted to a cylinder, and then unrolled to be flat. It is called an "equirectangular projection". In other words, it maps meridians to vertical straight lines of constant spacing.

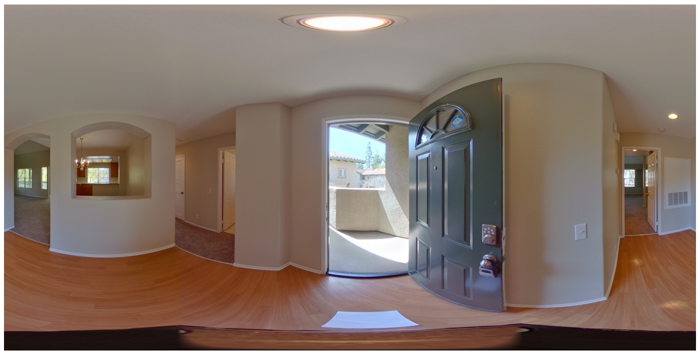


Figure A.2: A panorama provided in the Zillow Indoor Dataset (ZInd) [14].

Spherical Coordinate System

In the spherical coordinate system, we use an ordered triple to describe the location of a point in space. In this case, the triple (ρ, θ, φ) describes one distance and two angles.

If $\mathbf{P} = (x, y, z)$ is a point in space, and the x-y axes form the ground plane, and the z-axis points upwards, then:

- ρ is the distance between $\mathbf{P}$ and the origin.
- θ is the angle component of a polar coordinate in the x-y plane
- φ is the angle formed by the positive z-axis and line segment $\overline{OP}$, where O is the origin.

Conversion of rectangular coordinates to spherical coordinates Since ρ is the distance of a point on the sphere to the origin, then:

$$\begin{aligned}
\rho &= \| \begin{bmatrix} x \\ y \\ z \end{bmatrix} \|_2 \\
\rho^2 &= x^2 + y^2 + z^2 \\
\tan\theta &= \frac{y}{x} \\
\frac{z}{\rho} &= \frac{z}{\sqrt{x^2+y^2+z^2}} = \cos(\varphi) \\
\varphi &= \arccos\Big(\frac{z}{\sqrt{x^2+y^2+z^2}}\Big)
\end{aligned} \tag{A.30}$$

Conversion of spherical coordinates to rectangular coordinates We'll now discuss how to go in the opposite direction – from spherical to rectangular coordinates.

The sine of an angle is the length of the opposite leg, divided by the length of the hypotenuse: $sin(\varphi) = \frac{r}{\rho}$. Therefore $\rho \sin\varphi = r$.

At this point, we can consider only triangles inside the x-y plane. If r is the hypotenuse of a triangle in the x-y plane, then $x = r\cos\theta$ and $y = r\sin\theta$.

To compute z, we can notice one more right triangle above, where $\cos\varphi = \frac{z}{\rho}$, meaning $z = \rho\cos\varphi$.

We have derived the following relationships:

1. $x = \rho\sin\varphi\cos\theta = r\cos\theta$.
2. $y = \rho\sin\varphi\sin\theta = r\sin\theta$.
3. $z = \rho\cos\varphi$.

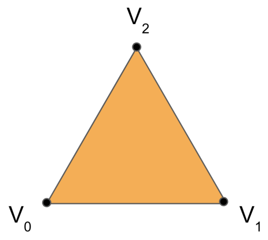
A.4 Raycasting

Unlike raytracing, raycasting seeks to find only the first intersection between a ray and a surface (if it exists), rather than continuing to trace the path. It turns out the math for this is quite simple to do naively, but the interesting part is figuring out how to make the implementation very cheap and fast [324]. We'll walk through the derivation from first principles.

A.4.1 System of Equations for Ray/Triangle Intersection

Suppose we model the surface as a triangle mesh. In order to find this intersection point P, we need a parametric equation of a ray, and of a triangle. Suppose we define the ray by its origin O and its direction R. The ray parametric equation is $P = O + tR$ where t is the distance from the ray origin O to point P. To find P, we need to compute distance t.

The three 3d vertices (v_0, v_1, v_2) of a triangle define a plane, and we can compute the plane's normal vector with a single cross product. If the vertices are ordered counter-



clockwise, then we can compute the unit length plane normal vector $N = (N_x, N_y, N_z)$ as:

$$\begin{aligned} N' &= (v_1 - v_0) \times (v_2 - v_0) \\ N &= \frac{N'}{\|N'\|_2} \end{aligned} \tag{A.31}$$

Now, we can form a system of equations. Suppose we have $P = (x, y, z)$. We know ray origin O and direction R, along with the plane normal $N = (A, B, C)$ and distance D, computed as $D = N \cdot v_0$. Our only unknowns are $P = (x, y, z)$ and t:

$$\begin{aligned} P &= O + tR \\ Ax + By + Cz + D &= 0 \\ A * P_x + B * P_y + C * P_z + D &= 0 \rightarrow \text{ rewrite } P = (x, y, z) \text{ as } P = (P_x, P_y, P_z) \end{aligned} \tag{A.32}$$

We'll now substitute P (from the first equation) to (x, y, z) in equation 2 and solve for distance t. Let $O = (O_x, O_y, O_z)$ and let $R = (R_x, R_y, R_z)$:

$$\begin{aligned} A * (O_x + tR_x) + B * (O_y + tR_y) + C * (O_z + tR_z) + D &= 0 \\ A * O_x + B * O_y + C * O_z + A * tR_x + B * tR_y + C * tR_z + D &= 0 \\ t * (A * R_x + B * R_y + C * R_z) + A * O_x + B * O_y + C * O_z + D &= 0 \\ t * (A * R_x + B * R_y + C * R_z) &= -\Big(A * O_x + B * O_y + C * O_z + D\Big) \\ t &= -\frac{A * O_x + B * O_y + C * O_z + D}{A * R_x + B * R_y + C * R_z} \\ t &= -\frac{N(A, B, C) \cdot O + D}{N(A, B, C) \cdot R} \end{aligned} \tag{A.33}$$

The expression simplifies nicely, and shows how to find where the ray intersects a 3d plane. There are at least 3 more conditions we need to check for, though:

1. Is the intersection point inside of the triangle's perimeter, or outside?

2. Are the ray direction and the 3d plane parallel to one another?

3. Is the triangle "behind" the ray?

For condition (1), we'll need another sub-routine, the "inside-outside test". We can determine condition (2) easily. If $|N \cdot R| < \epsilon$, where ϵ is close to zero, then the ray and plane are parallel.
Condition (3) is also easy to verify: after solving for t, we check the sign of t and ensure that it's non-negative.

A.5 Rotation averaging

The rotation averaging problem, also known as "rotations averaging", concerns the estimation of one or more rotations in a global frame, given pairwise rotation measurements. This problem is equivalent to the synchronization problem over the 3-dimensional Special Orthogonal Group $SO(3)$, and can serve as a useful initialization for bundle adjustment in Global Structure from Motion (SfM).

Let ${}^{w}\mathbf{R}_i$, sometimes written as $\mathbf{R}_i$ for brevity.

Often we we speak of rotation averaging, we are referring to the "multiple rotation averaging" problem [243]:

$$\underset{\mathbf{R_1},\dots\mathbf{R_n} \in SO(3)}{\text{argmin}} \sum_{(i,j)\in\mathcal{N}} d({}^{i}\overline{\mathbf{R}}_j, {}^{i}\mathbf{R}_w {}^{w}\mathbf{R}_j) \tag{A.34}$$

In Shonan averaging [119], the maximum likelihood problem is posed as

$$\max_{\mathbf{R}\in SO(d)^n} \sum_{(i,j)\in\mathcal{E}} \kappa_{ij} \text{tr}({}^{w}\mathbf{R}_i {}^{i}\overline{\mathbf{R}}_j {}^{j}\mathbf{R}_w)$$

Rotation averaging is a least squares problem in the tangent space of the $SO(3)$ manifold. Because $i\omega_j \approx \omega j - \omega i$ (the equality only holds for Baker-Campbell-Hausdorff form), a simple iterative algorithm [335] arises, by initializing global rotations $\{{}^{w}\mathbf{R}_i\}_{i=1}^{N}$

1. Compute residual: $\Delta^{j}\mathbf{R}_i = {}^{w}\mathbf{R}_j {}^{j}\mathbf{R}_i {}^{i}\mathbf{R}_w$

2. Apply logarithmic map: $\Delta^{j}\omega_i = \log(\Delta^{j}\mathbf{R}_i)$

3. Solve concatenated linear system: $\mathbf{A}\Delta\omega_{global} = \omega_{rel}$

4. Apply exponential map: $\forall k \in [1, N], \mathbf{R}_k = \mathbf{R}_k \exp(\Delta\omega_k)$

A.6 Translation Averaging

The translation averaging (also known as "translations averaging") problem estimates global translations of cameras in a world frame, given a number of pairwise 3 d.o.f. translation directions and global rotation estimates.

Given translation direction measurements (rays) ${}^{j}\hat{\mathbf{t}}_i$, and global rotation estimates ${}^{w}\mathbf{R}_j$, we can rotate these rays into a global frame via ${}^{w}_{j}\hat{\mathbf{t}}_i = {}^{w}\mathbf{R}_j{}^{j}\hat{\mathbf{t}}_i$. We can then establish an optimization problem between directions (rays) in the global frame, and global camera positions. 1dsfm [209] uses a sum of squared chordal distance:

$$E(\cdot) = \sum_{(i,j)\in E} d_{ch}\left({}^{w}_{j}\hat{\mathbf{t}}_i, \frac{{}^{w}\mathbf{t}_j - {}^{w}\mathbf{t}_i}{\|{}^{w}\mathbf{t}_j - {}^{w}\mathbf{t}_i\|}\right)^2 \tag{A.35}$$

where $d_{ch}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2$.

APPENDIX B
HIGH DEFINITION (HD) MAP FUNDAMENTALS

High-definition (HD) maps are representations of the world that are designed for use by autonomous vehicles. They include lane-level geometry, as well as other geometric data and semantic annotations [287, 288, 289, 2, 290, 4, 105, 10, 291, 6, 292, 7, 13, 11, 12]. In this section, we describe a one possible representation of an HD map, as used in the Argoverse Datasets [4].

An HD map may include many distinct map components – such as (1) a vector map of lane centerlines and their attributes; (2) a rasterized map of ground height, (3) a rasterized map of driveable area and region of interest (ROI), as well as other possible components.

B.0.1 Vector Map of Lane Geometry

A *vector map* consists of semantic road data represented as a localized graph rather than rasterized into discrete samples. In a vector map, the lane graph may be represented by lane boundaries or instead by lane centerlines, split into lane segments. Vehicle trajectories generally follow the center of a lane, so a centerline is a useful prior for tracking and forecasting.

A lane segment is a segment of road where cars drive in single-file fashion in a single direction. Multiple lane segments may occupy the same physical space (e.g. in an intersection). Turning lanes which allow traffic to flow in either direction are represented by two different lanes that occupy the same physical space.

For each lane centerline, a number of semantic attributes are possible. In Argoverse v1.0 [4], these lane attributes describe whether a lane is located within an intersection or has an associated traffic control measure (Boolean values that are not mutually inclusive). Other semantic attributes include the lane's turn direction (*left, right, or none*) and the unique identifiers for the lane's predecessors (lane segments that come before) and successors (lane segments that come after) of which there can be multiple (for merges and splits, respectively). Centerlines are provided as "polylines", i.e. an ordered sequence of straight segments. Each straight segment is defined by 2 vertices: (x_i, y_i, z_i) start and $(x_{i+1}, y_{i+1}, z_{i+1})$ end. Thus, curved lanes are approximated with a set of straight lines.

In the United States, lane segments adhere to certain priors. For example, in Miami, lane segments that could be used for route planning are on average 3.84 ± 0.89 m wide. In Pittsburgh, the average width is 3.97 ± 1.04 m. Other types of lane segments that would not be suitable for self-driving, e.g. bike lanes, can be as narrow as 0.97 m in Miami and as narrow as 1.06 m in Pittsburgh [4].

B.0.2 Rasterized Driveable Area Map

An HD map may include binary driveable area labels at some regular grid resolution, e.g. 1 meter resolution for Argoverse 1.0 and 30 centimeter resolution for Argoverse 2.0. A driveable area is an area where it is possible for a vehicle to drive (though not necessarily legal). Driveable areas can encompass a road's shoulder in addition to the normal driveable area that is represented by a lane segment. A dilated isocontour of the driveable area, e.g. all areas within 5 meters of the driveable area, can be useful for reasoning about entities of interest for self-driving. This larger area is sometimes referred to as a *region of interest* (ROI).

B.0.3 Rasterized Ground Height Map

HD maps may also include real-valued ground height at some regular grid resolution, e.g. 1 meter resolution for Argoverse 1.0. Knowledge of ground height can be used to remove LiDAR returns on static ground surfaces and thus makes the 3D detection of dynamic objects easier. Figure B.1 shows a cross section of a scene with uneven ground height.

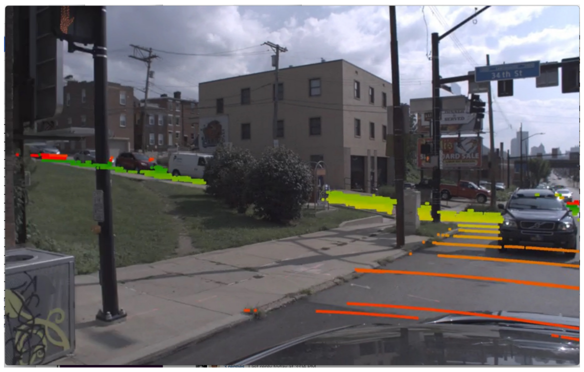


Figure B.1: A scene with non-planar ground surface. The colored LiDAR returns have been classified as belonging to the ground, based on the map. Points outside the driveable area are also discarded. This simple distance threshold against a map works well, even on the road to the left which goes steeply uphill.

B.0.4 Coordinate System

HD maps require a global, world coordinate system. In Argoverse 1.0 and Argoverse 2.0, the model of the world used within the maps is a local tangent plane centered at a central point located within each city [4, 326]. This model has a flat earth assumption which is approximately correct at the scale of a city. In Argoverse 1.0 and 2.0, map objects are defined in city coordinates. City coordinates can be converted to the UTM (Universal Transverse Mercator) coordinate system by simply adding the city's origin in UTM coordinates to the object's city coordinate pose. The UTM model divides the earth into 60 flattened, narrow zones, each of width 6 degrees of longitude. Each zone is segmented into 20 latitude bands.

We favor a city-level coordinate system because of its high degree of interpretability when compared with geocentric reference coordinate systems such as the 1984 World Geodetic System (WGS84). While WGS84 is widely used by the Global Positioning System, the model is difficult to interpret at a city-scale; because its coordinate origin is located at the Earth's center of mass, travel across an entire city corresponds only to pose value changes in the hundredth decimal place. The conversion back and forth between UTM and WGS84 is well-known and is documented in detail in [340].

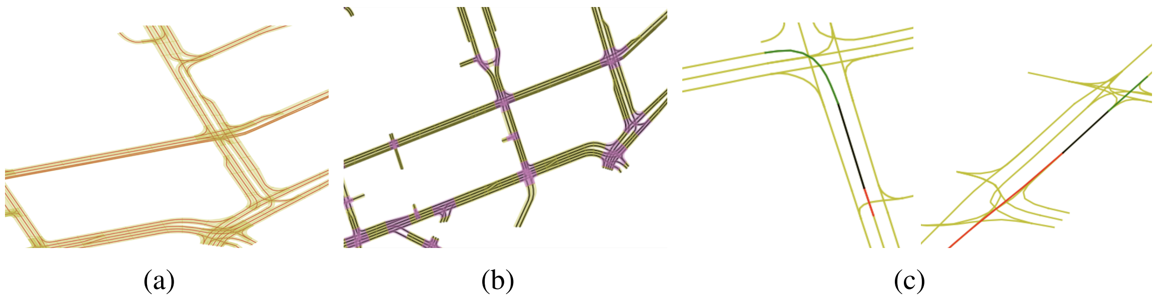
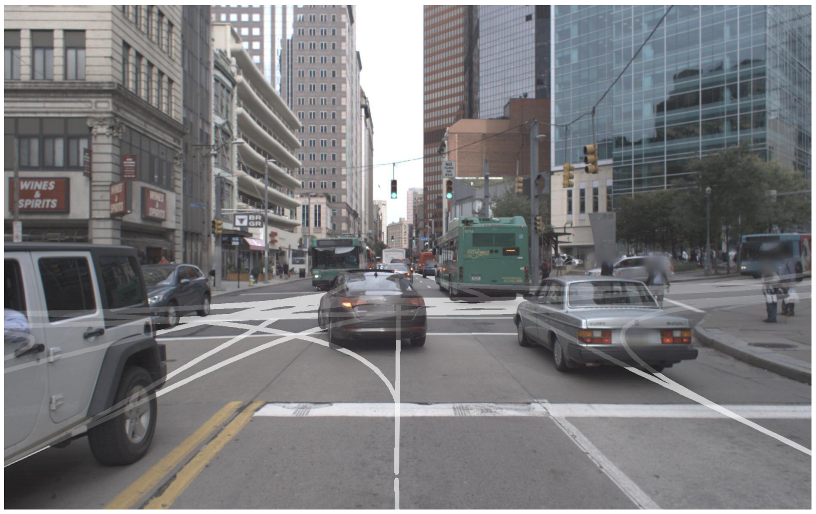
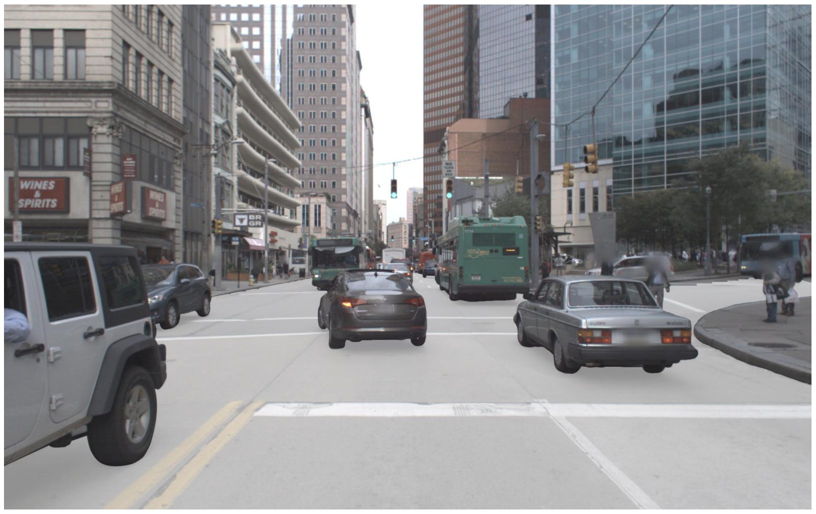


Figure B.2: (a) Lane centerlines and hallucinated area are shown in red and yellow, respectively. Argoverse 1.0 provides *lane* centerlines because simple *road* centerline representations cannot handle the highly complicated nature of real world mapping, as shown above with divided roads. (b) Lane segments within intersections are shown in pink, and all other lane segments in yellow. Black shows lane centerlines. (c) Example of a specific lane centerline's successors and predecessors. Red shows the predecessor, green shows the successor, and black indicates the centerline segment of interest.

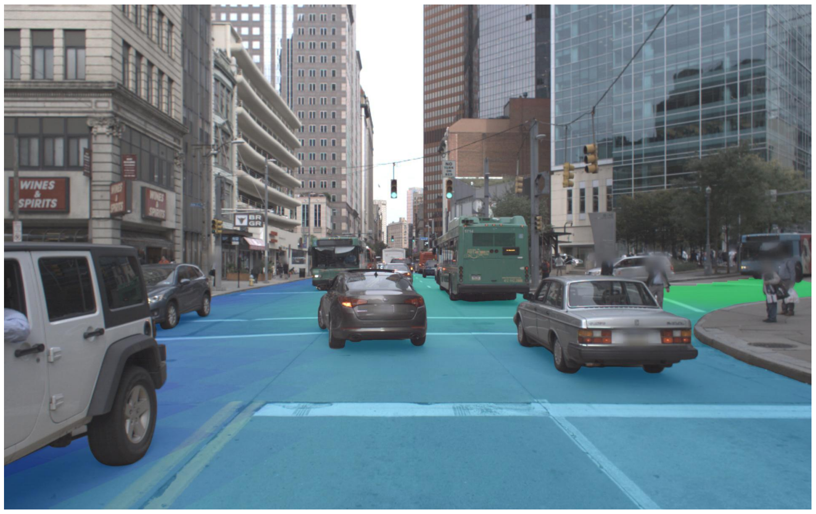
Figure B.2 shows examples of the centerlines which are the basis of the Argoverse 1.0 vector map. Centerline attributes include whether or not lane segments are in an intersection, and which lane segments constitute their predecessors and successors. Figure B.3 shows examples of centerlines, driveable area, and ground height projected onto a camera image.



(a) Lane geometry and connectivity



(b) Driveable area



(c) Ground height

Figure B.3: Examples of centerlines, driveable area, and ground height projected onto a camera image.

REFERENCES

[1] P. Lindenberger, P.-E. Sarlin, V. Larsson, and M. Pollefeys, "Pixel-Perfect Structure-from-Motion with Featuremetric Refinement," in *ICCV*, 2021.

[2] B. Yang, M. Liang, and R. Urtasun, "HDNET: exploiting HD maps for 3d object detection," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, ser. Proceedings of Machine Learning Research, vol. 87, PMLR, 2018, pp. 146–155.

[3] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, ser. Proceedings of Machine Learning Research, vol. 87, PMLR, 2018, pp. 947–956.

[4] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[5] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *CVPR*, Jun. 2020.

[6] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *CVPR*, Jun. 2020.

[7] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.

[8] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov, "Tnt: Target-driven trajectory prediction," in *4th Annual Conference on Robot Learning, CoRL 2020*, 2020.

[9] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning (CoRL)*, 2019.

[10] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *CVPR*, Jun. 2019.

[11] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun, "Scenegen: Learning to generate realistic traffic scenes," in *CVPR*, Jun. 2021.

[12] S. Suo, S. Regalado, S. Casas, and R. Urtasun, “Trafficsim: Learning to simulate realistic multi-agent behaviors,” in *CVPR*, Jun. 2021, pp. 10 400–10 409.

[13] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun, “Geosim: Realistic video simulation via geometry-aware composition for self-driving,” in *CVPR*, Jun. 2021.

[14] S. Cruz, W. Hutchcroft, Y. Li, N. Khosravan, I. Boyadzhiev, and S. B. Kang, “Zillow indoor dataset: Annotated floor plans with 360deg panoramas and 3D room layouts,” in *CVPR*, Jun. 2021, pp. 2133–2143.

[15] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, “Matterport3d: Learning from rgb-d data in indoor environments,” *International Conference on 3D Vision (3DV)*, 2017.

[16] A. Cohen, J. L. Schönberger, P. Speciale, T. Sattler, J.-M. Frahm, and M. Pollefeys, “Indoor-outdoor 3D reconstruction alignment,” in *ECCV*, vol. 9907, 2016, pp. 285–300.

[17] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, “Tanks and temples: Benchmarking large-scale scene reconstruction,” *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[18] J. L. Schonberger and J.-M. Frahm, “Structure-from-motion revisited,” in *CVPR*, Jun. 2016.

[19] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, “Openmvg: Open multiple view geometry,” in *International Workshop on Reproducible Research in Pattern Recognition*, Springer, 2016, pp. 60–74.

[20] C. Sweeney, T. Hollerer, and M. Turk, “Theia: A fast and scalable structure-from-motion library,” in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 693–696.

[21] D. G. Lowe, “Object recognition from local scale-invariant features,” in *ICCV*, Ieee, vol. 2, 1999, pp. 1150–1157.

[22] ——, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[23] H. Aanæs, A. L. Dahl, and K. Steenstrup Pedersen, “Interesting interest points,” *Int. J. Comput. Vision*, vol. 97, no. 1, pp. 18–35, Mar. 2012.

[24] A. Albarelli, E. Rodolà, and A. Torsello, “Robust game-theoretic inlier selection for bundle adjustment,” in *International Symposium on 3D Data Processing, Visualization and Transmission*, 2010.

[25] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, “Kaze features,” in *ECCV*, 2012.

[26] J. Aldana-Iuit, D. Mishkin, O. Chum, and J. Matas, “In the saddle: Chasing fast and repeatable features,” in *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, 2016, pp. 675–680.

[27] H. Altwaijry, A. Veit, and S. Belongie, “Learning to detect and match keypoints with deep architectures,” in *British Machine Vision Conference (BMVC)*, York, UK, 2016.

[28] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *CVPR*, IEEE, 2012, pp. 2911–2918.

[29] B. Babenko, P. Dollár, and S. Belongie, “Task specific local region matching,” in *2007 IEEE 11th International Conference on Computer Vision*, IEEE, 2007, pp. 1–8.

[30] D. P. Vassileios Balntas Edgar Riba and K. Mikolajczyk, “Learning local feature descriptors with triplets and shallow convolutional neural networks,” in *Proceedings of the British Machine Vision Conference (BMVC)*, E. R. H. Richard C. Wilson and W. A. P. Smith, Eds., BMVA Press, Sep. 2016, pp. 119.1–119.11.

[31] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk, “Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters,” in *ICCV*, 2019.

[32] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[33] A. Bhowmik, S. Gumhold, C. Rother, and E. Brachmann, “Reinforced feature points: Optimizing feature detection and description for a high-level task,” in *CVPR*, 2020, pp. 4948–4957.

[34] J. Bian, W.-Y. Lin, Y. Matsushita, S.-K. Yeung, T.-D. Nguyen, and M.-M. Cheng, “Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[35] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, “Dsac - differentiable ransac for camera localization,” in *CVPR*, Jul. 2017.

[36] E. Brachmann and C. Rother, “Neural-guided ransac: Learning where to sample model hypotheses,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019.

[37] G. H. M. Brown and S. Winder, “Discriminative learning of local image descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 43–57, Feb. 2010.

[38] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ser. ECCV’10, Heraklion, Crete, Greece: Springer-Verlag, 2010, pp. 778–792.

[39] T. Cieslewski, M. Bloesch, and D. Scaramuzza, “Matching features without descriptors: Implicitly matched interest points,” in *British Machine Vision Conference (BMVC)*, 2019.

[40] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, “Universal correspondence network,” in *Advances in Neural Information Processing Systems 29*, 2016.

[41] Z. Dang, K. Moo Yi, Y. Hu, F. Wang, P. Fua, and M. Salzmann, “Eigendecomposition-free training of deep networks with zero eigenvalue-based losses,” in *ECCV*, Sep. 2018.

[42] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2018.

[43] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, “D2-Net: A Trainable CNN for Joint Detection and Description of Local Features,” in *CVPR*, 2019.

[44] P. Ebel, A. Mishchuk, K. M. Yi, P. Fua, and E. Trulls, “Beyond cartesian representations for local descriptors,” in *ICCV*, Oct. 2019.

[45] P. Di Febbo, C. Dal Mutto, K. Tieu, and S. Mattoccia, “Kcnn: Extremely-efficient hardware keypoint detection with a compact convolutional neural network,” in *CVPR*, Jun. 2018.

[46] P. Fischer, A. Dosovitskiy, and T. Brox, “Descriptor matching with convolutional neural networks: A comparison to sift,” *arXiv preprint arXiv:1405.5769*, 2014.

[47] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, “Matchnet: Unifying feature and metric learning for patch-based matching,” in *CVPR*, Jun. 2015.

[48] W. Hartmann, M. Havlena, and K. Schindler, “Predicting matchability,” in *CVPR*, 2014.

[49] K. He, Y. Lu, and S. Sclaroff, “Local descriptors optimized for average precision,” in *CVPR*, Jun. 2018.

[50] M. Jahrer, M. Grabner, and H. Bischof, “Learned local descriptors for recognition and matching,” in *Proceedings of the Computer Vision Winter Workshop 2008*, 2008, pp. 39–46.

[51] Y. Ke and R. Sukthankar, “Pca-sift: A more distinctive representation for local image descriptors,” in *CVPR*, IEEE, vol. 2, 2004, pp. II–II.

[52] W. Kienzle, F. A. Wichmann, B. Scholkopf, and M. O. Franz, “Learning an interest operator from human eye movements,” in *CVPRW*, 2006.

[53] K. Lenc and A. Vedaldi, “Learning covariant feature detectors,” in *ECCV Workshops (3)*, ser. Lecture Notes in Computer Science, vol. 9915, 2016, pp. 100–117.

[54] S. Leutenegger, M. Chli, and R. Y. Siegwart, “Brisk: Binary robust invariant scalable keypoints,” in *ICCV*, ser. ICCV ’11, USA: IEEE Computer Society, 2011, pp. 2548–2555.

[55] W.-Y. Lin, F. Wang, M.-M. Cheng, S.-K. Yeung, P. H. S. Torr, M. N. Do, and J. Lu, “CODE: coherence based decision boundaries for feature correspondence,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 34–47, 2018.

[56] Z. Luo, T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan, “Geodesc: Learning local descriptors by integrating geometry constraints,” in *ECCV*, 2018, pp. 168–183.

[57] Z. Luo, T. Shen, L. Zhou, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan, “Contextdesc: Local descriptor augmentation with cross-modality context,” in *CVPR*, Jun. 2019.

[58] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[59] K. Mikolajczyk and C. Schmid, “Indexing based on scale invariant interest points,” in *ICCV*, vol. 1, Vancouver, Canada, Jul. 2001, pp. 525–531.

[60] ——, “An affine invariant interest point detector,” in *ECCV*, Springer, 2002, pp. 128–142.

[61] ——, “Scale & affine invariant interest point detectors,” *Int. J. Comput. Vision*, vol. 60, no. 1, pp. 63–86, Oct. 2004.

[62] ——, “A performance evaluation of local descriptors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[63] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, “A comparison of affine region detectors,” *Int. J. Comput. Vision*, vol. 65, no. 1-2, pp. 43–72, Nov. 2005.

[64] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, “Working hard to know your neighbor’s margins: Local descriptor learning loss,” in *Advances in Neural Information Processing Systems*, 2017, pp. 4826–4837.

[65] D. Mishkin, F. Radenovic, and J. Matas, “Repeatability is not enough: Learning affine regions via discriminability,” in *ECCV*, Sep. 2018.

[66] A. Mukundan, G. Tolias, and O. Chum, “Explicit spatial encoding for deep local descriptors,” in *CVPR*, Jun. 2019.

[67] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, “Large-scale image retrieval with attentive deep local features,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[68] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, “Lf-net: Learning local features from images,” in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 6236–6246.

[69] T. Plötz and S. Roth, “Neural nearest neighbors networks,” in *NeurIPS*, 2018, pp. 1087–1098.

[70] R. Ranftl and V. Koltun, “Deep fundamental matrix estimation,” in *The European Conference on Computer Vision (ECCV)*, Sep. 2018.

[71] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm, “Usac: A universal framework for random sample consensus,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 2022–2038, Aug. 2013.

[72] J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel, “R2d2: Reliable and repeatable detector and descriptor,” in *NeurIPS*, 2019, pp. 12 405–12 415.

[73] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *ECCV*, 2006.

[74] E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010.

[75] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *ICCV*, 2011, pp. 2564–2571.

[76] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “SuperGlue: Learning feature matching with graph neural networks,” in *CVPR*, 2020.

[77] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys, “Quad-networks: Unsupervised learning to rank for interest point detection,” in *CVPR*, Jul. 2017.

[78] K. Simonyan, A. Vedaldi, and A. Zisserman, “Learning local feature descriptors using convex optimisation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[79] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, “Discriminative learning of deep convolutional feature point descriptors,” in *ICCV*, IEEE Computer Society, 2015, pp. 118–126.

[80] C. Strecha, A. Lindner, K. Ali, and P. Fua, “Training for task specific keypoint detection,” in *Joint Pattern Recognition Symposium*, Springer, 2009, pp. 151–160.

[81] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, “LDAHash: Improved matching with smaller descriptors,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 1, pp. 66–78, 2011.

[82] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, “LoFTR: Detector-free local feature matching with transformers,” in *CVPR*, Jun. 2021, pp. 8922–8931.

[83] J. Tang, H. Kim, V. Guizilini, S. Pillai, and R. Ambrus, “Neural outlier rejection for self-supervised keypoint learning,” in *International Conference on Learning Representations*, 2020.

[84] Y. Tian, B. Fan, and F. Wu, “L2-net: Deep learning of discriminative patch descriptor in euclidean space,” in *CVPR*, Jul. 2017.

[85] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, “Sosnet: Second order similarity regularization for local descriptor learning,” in *CVPR*, Jun. 2019.

[86] E. Tola, V. Lepetit, and P. Fua, “Daisy: An efficient dense descriptor applied to wide-baseline stereo.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, 2010.

[87] P. Truong, S. Apostolopoulos, A. Mosinska, S. Stucky, C. Ciller, and S. D. Zanet, "Glampoints: Greedily learned accurate match points," in *ICCV*, Oct. 2019.

[88] T. Tuytelaars and L. v. Gool, "Wide baseline stereo matching based on local, affinely invariant regions," in *Proceedings of the British Machine Vision Conference*, BMVA Press, 2000, pp. 38.1–38.14.

[89] T. Tuytelaars, K. Mikolajczyk, *et al.*, "Local invariant feature detectors: A survey," *Foundations and trends® in computer graphics and vision*, vol. 3, no. 3, pp. 177–280, 2008.

[90] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "TILDE: A temporally invariant learned detector.," in *CVPR*, IEEE Computer Society, Jun. 2015, pp. 5279–5288.

[91] S. A. J. Winder, G. Hua, and M. A. Brown, "Picking the best DAISY," in *CVPR*, IEEE Computer Society, 2009, pp. 178–185.

[92] S. Winder and M. Brown, "Learning local image descriptors," in *CVPR*, Minneapolis, Jun. 2007.

[93] G. Yang, T. Malisiewicz, and S. Belongie, "Learning data-adaptive interest points through epipolar adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2019.

[94] K. Moo Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, "Learning to find good correspondences," in *CVPR*, Jun. 2018.

[95] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *ECCV*, vol. 9910, 2016, pp. 467–483.

[96] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *CVPR*, Jun. 2015.

[97] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao, "Learning two-view correspondences and geometry using order-aware network," in *ICCV*, Oct. 2019.

[98] X. Zhang, F. X. Yu, S. Karaman, and S.-F. Chang, "Learning discriminative and transformation covariant local feature detectors," in *CVPR*, Jul. 2017.

[99] X. Zhang, F. X. Yu, S. Kumar, and S.-F. Chang, "Learning spread-out local feature descriptors," in *ICCV*, Oct. 2017.

[100] L. Zhang and S. Rusinkiewicz, "Learning local descriptors with a cdf-based dynamic soft margin," in *ICCV*, Oct. 2019.

[101] ——, “Learning to detect features in texture images,” in *CVPR*, Jun. 2018.

[102] C. Zhao, Z. Cao, C. Li, X. Li, and J. Yang, “Nm-net: Mining reliable neighbors for robust feature correspondences,” in *CVPR*, Jun. 2019.

[103] V. M. Govindu, “Lie-algebraic averaging for globally consistent motion estimation,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, IEEE, vol. 1, 2004, pp. I–I.

[104] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, and E. Trulls, “Image matching across wide baselines: From paper to practice,” *International Journal of Computer Vision*, vol. 129, no. 2, pp. 517–547, 2021.

[105] N. Homayounfar, W.-C. Ma, J. Liang, X. Wu, J. Fan, and R. Urtasun, “DAGMapper: Learning to map by discovering lane topology,” in *ICCV*, Oct. 2019.

[106] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “Nuscenes: A multimodal dataset for autonomous driving,” in *CVPR*, 2020, pp. 11 621–11 631.

[107] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov, *Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset*, 2021. arXiv: 2104.10133 [cs.CV].

[108] Q. Li, Y. Wang, Y. Wang, and H. Zhao, *HDMapNet: An online hd map construction and evaluation framework*, 2021. arXiv: 2107.06307 [cs.CV].

[109] A. Karpathy. (2021). “Tesla AI Day Presentation,” Tesla.

[110] Y. B. Can, A. Liniger, D. P. Paudel, and L. Van Gool, “Structured bird’s-eye-view traffic scene understanding from onboard images,” in *ICCV*, Oct. 2021, pp. 15 661–15 670.

[111] D. Pannen, M. Liebner, W. Hempel, and W. Burgard, “How to keep HD maps for automated driving up to date,” in *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, IEEE, 2020, pp. 2288–2294.

[112] D. H. Silver and D. I. F. Ferguson, *Change detection using curve alignment*, US Patent 9,321,461, Apr. 2016.

[113] K. Jo, C. Kim, and M. Sunwoo, “Simultaneous localization and map change update for the high definition map-based autonomous driving car,” *Sensors*, vol. 18, no. 9, 2018.

[114] W. Ding, S. Hou, H. Gao, G. Wan, and S. Song, “Lidar inertial odometry aided robust lidar localization system in changing city scenes,” in *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, IEEE, 2020, pp. 4322–4328.

[115] J. Hawke, H. E, V. Badrinarayanan, and A. Kendall, “Reimagining an autonomous vehicle,” *CoRR*, vol. abs/2108.05805, 2021. arXiv: 2108.05805.

[116] F. Dellaert, “Factor graphs and GTSAM: A hands-on introduction,” Georgia Institute of Technology, Tech. Rep., 2012.

[117] C. Sun, C.-W. Hsiao, M. Sun, and H.-T. Chen, “Horizonnet: Learning room layout with 1D representation and pano stretch data augmentation,” in *CVPR*, Jun. 2019.

[118] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard, “SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group,” *Intl. J. of Robotics Research*, vol. 38, no. 2–3, pp. 95–125, Mar. 2019.

[119] F. Dellaert, D. M. Rosen, J. Wu, R. E. Mahony, and L. Carlone, “Shonan rotation averaging: Global optimality by surfing $SO(p)^{\mathrm{n}}$,” in *ECCV*, vol. 12351, 2020, pp. 292–308.

[120] J. W. Lambert and J. Hays, “Trust, but Verify: Cross-modality fusion for hd map change detection,” in *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[121] P. Gargallo, Y. Kuang, *et al.*, *OpenSfM*, 2016.

[122] M. A. Shabani, W. Song, M. Odamaki, H. Fujiki, and Y. Furukawa, “Extreme structure from motion for indoor panoramas without visual overlaps,” in *ICCV*, Oct. 2021, pp. 5703–5711.

[123] S. Y. Bao and S. Savarese, “Semantic structure from motion,” in *CVPR 2011*, 2011, pp. 2025–2032.

[124] A. Cohen, T. Sattler, and M. Pollefeys, “Merging the unmatchable: Stitching visually disconnected SfM models,” in *ICCV*, Dec. 2015.

[125] C. Lin, C. Li, and W. Wang, “Floorplan-jigsaw: Jointly estimating scene layout and aligning partial scans,” in *ICCV*, Oct. 2019.

[126] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” *ArXiv*, vol. abs/2110.14168, 2021.

[127] J. Shen, Y. Yin, L. Li, L. Shang, M. Zhang, and Q. Liu, “Generate & Rank: A multi-task framework for math word problems,” in *Findings of the Association for Computational Linguistics: EMNLP 2021*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2269–2279.

[128] G. Pintore, C. Mura, F. Ganovelli, L. Fuentes-Perez, R. Pajarola, and E. Gobbetti, “State-of-the-art in automatic 3D reconstruction of structured indoor environments,” *Computer Graphics Forum*, vol. 39, no. 2, pp. 667–699, 2020.

[129] G. Albanis, N. Zioulis, P. Drakoulis, V. Gkitsas, V. Sterzentsenko, F. Álvarez, D. Zarpalas, and P. Daras, “Pano3D: A holistic benchmark and a solid baseline for 360° depth estimation,” *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3722–3732, 2021.

[130] P. E. Debevec, C. J. Taylor, and J. Malik, “Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’96, 1996, pp. 11–20.

[131] D. Farin, W. Effelsberg, and P. H. de With, “Floor-plan reconstruction from panoramic images,” in *Proceedings of the 15th ACM international conference on Multimedia*, 2007, pp. 823–826.

[132] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, “Reconstructing building interiors from images,” in *ICCV*, 2009, pp. 80–87.

[133] R. Cabral and Y. Furukawa, “Piecewise planar and compact floorplan reconstruction from images,” in *CVPR*, Jun. 2014.

[134] G. Pintore, F. Ganovelli, R. Pintus, R. Scopigno, and E. Gobbetti, “3D floor plan recovery from overlapping spherical images,” *Computational visual media*, vol. 4, no. 4, pp. 367–383, 2018.

[135] G. Pintore, F. Ganovelli, A. J. Villanueva, and E. Gobbetti, “Automatic modeling of cluttered multi-room floor plans from panoramic images,” *Comput. Graph. Forum*, vol. 38, no. 7, pp. 347–358, 2019.

[136] C. Liu, J. Wu, and Y. Furukawa, “Floornet: A unified framework for floorplan reconstruction from 3D scans,” in *ECCV*, Sep. 2018.

[137] J. Chen, C. Liu, J. Wu, and Y. Furukawa, "Floor-SP: Inverse CAD for floorplans by sequential room-wise shortest path," in *ICCV*, Oct. 2019.

[138] S. Stekovic, M. Rad, F. Fraundorfer, and V. Lepetit, "Montefloor: Extending MCTS for reconstructing accurate large-scale floor plans," in *ICCV*, Oct. 2021, pp. 16 034–16 043.

[139] S. Purushwalkam, S. V. A. Garı, V. K. Ithapu, C. Schissler, P. Robinson, A. Gupta, and K. Grauman, "Audio-visual floorplan reconstruction," in *ICCV*, Oct. 2021, pp. 1183–1192.

[140] B. Okorn, X. Xiong, B. Akinci, and D. Huber, "Toward automated modeling of floor plans," in *3D DPVT*, 2010.

[141] Y. M. Kim, J. Dolson, M. Sokolsky, V. Koltun, and S. Thrun, "Interactive acquisition of residential floor plans," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3055–3062.

[142] H. Fang, C. Pan, and H. Huang, "Structure-aware indoor scene reconstruction via two levels of abstraction," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 178, pp. 155–170, 2021.

[143] H. Fang, F. Lafarge, C. Pan, and H. Huang, "Floorplan generation from 3D point clouds: A space partitioning approach," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 175, pp. 44–55, 2021.

[144] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer, "A survey of structure from motion," *Acta Numerica*, vol. 26, May 2017.

[145] P. Moulon, P. Monasse, and R. Marlet, "Global fusion of relative motions for robust, accurate and scalable structure from motion," in *ICCV*, 2013, pp. 3248–3255.

[146] O. Enqvist, F. Kahl, and C. Olsson, "Non-sequential structure from motion," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 264–271.

[147] C. Zach, M. Klopschitz, and M. Pollefeys, "Disambiguating visual relations using loop constraints," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1426–1433.

[148] K. Wilson and N. Snavely, "Robust global translations with 1DSfM," in *ECCV*, vol. 8691, Springer, 2014, pp. 61–75.

[149] S. Choudhary, A. J. Trevor, H. I. Christensen, and F. Dellaert, "SLAM with object discovery, modeling and mapping," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 1018–1025.

[150] Z. Yang, J. Z. Pan, L. Luo, X. Zhou, K. Grauman, and Q. Huang, "Extreme relative pose estimation for RGB-D scans via scene completion," in *CVPR*, Jun. 2019.

[151] Z. Yang, S. Yan, and Q. Huang, "Extreme relative pose network under hybrid representations," in *CVPR*, Jun. 2020.

[152] K. Chen, N. Snavely, and A. Makadia, "Wide-baseline relative camera pose estimation with directional learning," in *CVPR*, Jun. 2021, pp. 3258–3268.

[153] L. Jin, S. Qian, A. Owens, and D. F. Fouhey, "Planar surface reconstruction from sparse views," in *ICCV*, 2021.

[154] Z. Laskar, I. Melekhov, S. Kalia, and J. Kannala, "Camera relocalization by computing pairwise relative poses using convolutional neural network," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct. 2017.

[155] V. Balntas, S. Li, and V. Prisacariu, "Relocnet: Continuous metric learning relocalisation using neural nets," in *ECCV*, 2018, pp. 751–767.

[156] M. Ding, Z. Wang, J. Sun, J. Shi, and P. Luo, "CamNet: Coarse-to-fine retrieval for camera re-localization," in *ICCV*, 2019, pp. 2871–2880.

[157] F. Zhang, N. Nauata, and Y. Furukawa, "Conv-MPN: Convolutional message passing neural network for structured outdoor architecture reconstruction," in *CVPR*, 2020, pp. 2798–2807.

[158] C. Sun, M. Sun, and H.-T. Chen, "Hohonet: 360 indoor holistic understanding with latent horizontal features," in *CVPR*, Jun. 2021, pp. 2573–2582.

[159] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[160] Y. Zhang, S. Song, P. Tan, and J. Xiao, "Panocontext: A whole-room 3D context model for panoramic scene understanding," in *ECCV*, vol. 8694, Heidelberg: Springer, 2014, pp. 668–686.

[161] M. Aly and J.-Y. Bouguet, "Street view goes indoors: Automatic pose estimation from uncalibrated unordered spherical panoramas," in *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*, 2012, pp. 1–8.

[162] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *CVPR*, IEEE, vol. 2, 1999, pp. 588–594.

[163] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from RGB-D data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[164] C. Zou, J.-W. Su, C.-H. Peng, A. Colburn, Q. Shan, P. Wonka, H.-K. Chu, and D. Hoiem, "Manhattan room layout reconstruction from a single 360° image: A comparative study of state-of-the-art methods," *International Journal of Computer Vision*, vol. 129, no. 5, pp. 1410–1431, May 2021.

[165] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, and Z. Zhou, "Structured3d: A large photo-realistic dataset for structured 3D modeling," in *ECCV*, Springer, 2020, pp. 519–535.

[166] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *CVPR*, Jul. 2017.

[167] M. Oskarsson, "Two-view orthographic epipolar geometry: Minimal and optimal solvers," *Journal of Mathematical Imaging and Vision*, vol. 60, no. 2, pp. 163–173, 2018.

[168] S. Choi and J.-H. Kim, "Fast and reliable minimal relative pose estimation under planar motion," *Image and Vision Computing*, vol. 69, pp. 103–112, 2018.

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

[170] B. Reddy and B. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE Transactions on Image Processing*, vol. 5, no. 8, pp. 1266–1271, 1996.

[171] K. Son, D. Moreno, J. Hays, and D. B. Cooper, "Solving small-piece jigsaw puzzles by growing consensus," in *CVPR*, Jun. 2016.

[172] F. Dellaert, "Factor graphs: Exploiting structure in robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 141–166, 2021.

[173] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation," in *ECCV*, Springer, 2020, pp. 162–177.

[174] N. Nauata, S. Hosseini, K.-H. Chang, H. Chu, C.-Y. Cheng, and Y. Furukawa, "House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects," in *CVPR*, Jun. 2021, pp. 13 632–13 641.

[175] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 6, pp. 580–593, Jun. 1997.

[176] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–777, Jun. 2004.

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

[178] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR*, Ieee, vol. 2, 2006, pp. 2161–2168.

[179] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, "Semantic visual localization," in *CVPR*, Jun. 2018.

[180] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[181] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *ECCV*, 2016.

[182] Mapillary, *Open source structure from motion pipeline*, 2013.

[183] C. Wu, "Towards linear-time incremental structure from motion," in *Proceedings of the 2013 International Conference on 3D Vision*, ser. 3DV '13, 2013, pp. 127–134.

[184] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-80-03, Sep. 1980.

[185] P. Beaudet, "Rotationally invariant image operators," in *Proceedings of the 4th International Joint Conference on Pattern Recognition (ICPR)*, 1978, pp. 579–583.

[186] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[187] S. M. Smith and J. M. Brady, "Susan—a new approach to low level image processing," *International journal of computer vision*, vol. 23, no. 1, pp. 45–78, 1997.

[188] P. Dias, A. Kassim, and V. Srinivasan, "A neural network based corner detection method," in *Proceedings of ICNN'95-International Conference on Neural Networks*, IEEE, vol. 4, 2005, pp. 2116–2120.

[189] S. Honari, P. Molchanov, S. Tyree, P. Vincent, C. Pal, and J. Kautz, "Improving landmark localization with semi-supervised learning," in *CVPR*, Jun. 2018.

[190] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *Int. J. Comput. Vision*, vol. 37, no. 2, pp. 151–172, Jun. 2000.

[191] K. Lenc and A. Vedaldi, "Large scale evaluation of local image feature detectors on homography datasets," in *BMVC*, BMVA Press, 2018, p. 122.

[192] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 5, pp. 530–535, 1997.

[193] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," in *Advances in neural information processing systems*, 2001, pp. 831–837.

[194] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *CVPR*, 2015.

[195] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative Evaluation of Binary Features," in *ECCV*, 2012.

[196] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *CVPR*, 2017.

[197] C. Choy and J. Lee, *Open universal correspondence network*, https://github.com/chrischoy/open-ucn, 2019.

[198] P. Pritchett and A. Zisserman, "Wide baseline stereo matching," in *ICCV*, 1998, pp. 754–760.

[199] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06, Boston, Massachusetts: Association for Computing Machinery, 2006, pp. 835–846, ISBN: 1595933646.

[200] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, vol. 79, no. 388, pp. 871–880, 1984.

[201] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[202] W. Sun, W. Jiang, E. Trulls, A. Tagliasacchi, and K. M. Yi, "Attentive context normalization for robust permutation-equivariant learning," *CoRR*, vol. abs/1907.02545, 2019. arXiv: 1907.02545.

[203] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative Evaluation of Hand-Crafted and Learned Local Features," in *CVPR*, 2017.

[204] K. He, *Mask r-cnn: A perspective on equivariance*, URL: http://kaiminghe.com/iccv17tutorial/maskrcnn_iccv2017_tutorial_kaiminghe.pdf, Nov. 2017.

[205] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *CVPR*, 2005.

[206] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *CVPR*, IEEE, 2009, pp. 2599–2606.

[207] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "Yfcc100m: The new data in multimedia research.," *Commun. ACM*, vol. 59, no. 2, pp. 64–73, 2016.

[208] J. Heinly, J. L. Schonberger, E. Dunn, and J.-M. Frahm, "Reconstructing the world* in six days *(as captured by the yahoo 100 million image dataset)," in *CVPR*, Jun. 2015.

[209] K. Wilson and N. Snavely, "Robust global translations with 1dsfm," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[210] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Self-improving visual odometry," *CoRR*, vol. abs/1812.03245, 2018.

[211] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ser. ICCV '99, London, UK, UK: Springer-Verlag, 2000, pp. 298–372, ISBN: 3-540-67973-1.

[212] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.

[213] ——, "Toward geometric deep slam," *arXiv preprint arXiv:1707.07410*, 2017.

[214] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, and E. Trulls, *Image matching across wide baselines: From paper to practice*, 2020. arXiv: 2003.01587 [cs.CV].

[215] F. Wang, S. Galliani, C. Vogel, P. Speciale, and M. Pollefeys, "Patchmatchnet: Learned multi-view patchmatch stereo," in *CVPR*, Jun. 2021, pp. 14 194–14 203.

[216] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *CVPR*, Jun. 2016.

[217] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[218] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction," in *ICCV*, 2021.

[219] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman, "Learning the depths of moving people by watching frozen people," in *CVPR*, Jun. 2019.

[220] V. M. Govindu, "Combining two-view constraints for motion estimation," in *CVPR*, IEEE, vol. 2, 2001, pp. II–II.

[221] ——, "Robustness in motion averaging," in *Asian Conference on Computer Vision*, Springer, 2006, pp. 457–466.

[222] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher, "Discrete-continuous optimization for large-scale structure from motion," in *CVPR 2011*, 2011, pp. 3001–3008.

[223] C. Sweeney, T. Sattler, T. Hollerer, M. Turk, and M. Pollefeys, "Optimizing the viewing graph for structure-from-motion," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[224] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[225] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.

[226] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *Int. J. Comput. Vision*, vol. 80, no. 2, pp. 189–210, Nov. 2008.

[227] C. Wu, “Towards linear-time incremental structure from motion,” in *2013 International Conference on 3D Vision - 3DV 2013*, 2013, pp. 127–134.

[228] P. Lindenberger, P.-E. Sarlin, V. Larsson, and M. Pollefeys, “Pixel-perfect structure-from-motion with featuremetric refinement,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5987–5997.

[229] C. Olsson and O. Enqvist, “Stable structure from motion for unordered image collections,” in *Scandinavian Conference on Image Analysis*, Springer, 2011, pp. 524–535.

[230] D. Martinec and T. Pajdla, “Robust rotation and translation estimation in multiview reconstruction,” in *CVPR*, 2007, pp. 1–8.

[231] A. Parra, S.-F. Chng, T.-J. Chin, A. Eriksson, and I. Reid, “Rotation coordinate descent for fast globally optimal rotation averaging,” in *CVPR*, 2021, pp. 4298–4307.

[232] K. Ni, D. Steedly, and F. Dellaert, “Out-of-core bundle adjustment for large-scale 3d reconstruction,” in *ICCV*, IEEE, 2007, pp. 1–8.

[233] K. Ni and F. Dellaert, “Hypersfm,” in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, IEEE, 2012, pp. 144–151.

[234] Y. Chen, S. Shen, Y. Chen, and G. Wang, “Graph-based parallel large scale structure from motion,” *Pattern Recognition*, p. 107 537, Jul. 2020.

[235] B. Klingner, D. Martin, and J. Roseborough, “Street view motion-from-structure-from-motion,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013.

[236] G. Sharp, S. Lee, and D. Wehe, “Toward multiview registration in frame space,” in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 4, 2001, 3542–3547 vol.4.

[237] S. Phillips and K. Daniilidis, “All graphs lead to rome: Learning geometric and cycle-consistent representations with graph convolutional networks,” *arXiv preprint arXiv:1901.02078*, 2019.

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

[239] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, “MVSNet: Depth inference for unstructured multi-view stereo,” in *ECCV*, 2018, pp. 767–783.

[240] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “PatchMatch: A randomized correspondence algorithm for structural image editing,” *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.

[241] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, “MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo,” *arXiv preprint arXiv:2103.15595*, 2021.

[242] L. F. Julià and P. Monasse, “A critical review of the trifocal tensor estimation,” in *Pacific-Rim Symposium on Image and Video Technology*, Springer, 2017, pp. 337–349.

[243] R. I. Hartley, J. Trumpf, Y. Dai, and H. Li, “Rotation averaging,” *Int. J. Comput. Vis.*, vol. 103, no. 3, pp. 267–305, 2013.

[244] K. Levenberg, “A method for the solution of certain non-linear problems in least squares,” *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[245] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[246] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *arXiv:2201.05989*, Jan. 2022.

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

[248] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[249] F. Liu, C. Shen, and G. Lin, “Deep convolutional neural fields for depth estimation from a single image,” in *CVPR*, Jun. 2015.

[250] K. Tateno, F. Tombari, I. Laina, and N. Navab, “Cnn-slam: Real-time dense monocular slam with learned depth prediction,” in *CVPR*, Jul. 2017.

[251] J.-R. Chang and Y.-S. Chen, “Pyramid stereo matching network,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[252] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, “Codeslam — learning a compact, optimisable representation for dense visual slam,” in *CVPR*, Jun. 2018.

[253] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, pp. 1–1, 2020.

[254] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Relative camera pose estimation using convolutional neural networks," in *International Conference on Advanced Concepts for Intelligent Vision Systems*, Springer, 2017, pp. 675–687.

[255] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[256] O. Poursaeed, G. Yang, A. Prakash, H. Jiang, Q. Fang, B. Hariharan, and S. Belongie, "Deep fundamental matrix estimation without correspondences," in *European Conference on Computer Vision Workshops (ECCVW)*, Munich, Germany, 2018.

[257] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *ECCV*, Springer, 2016, pp. 740–756.

[258] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "Sfm-net: Learning of structure and motion from video," *CoRR*, vol. abs/1704.07804, 2017. arXiv: 1704.07804.

[259] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "Demon: Depth and motion network for learning monocular stereo," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[260] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017.

[261] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *CVPR*, Jun. 2018.

[262] K. Lenc, "Representation of spatial transformations in deep neural networks," Ph.D. dissertation, University of Oxford, 2017.

[263] J. J. Koenderink and A. J. van Doorn, "Representation of local geometry in the visual system," *Biological cybernetics*, vol. 55, no. 6, pp. 367–375, 1987.

[264] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155–162, Jan. 1964.

[265] T.-Y. Liu *et al.*, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[266] J. Revaud, J. Almazan, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *ICCV*, Oct. 2019.

[267] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[268] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, IEEE, vol. 1, 2005, pp. 539–546.

[269] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *CVPR*, ser. CVPR '06, Washington, DC, USA: IEEE Computer Society, 2006, pp. 1735–1742, ISBN: 0-7695-2597-0.

[270] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[271] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CVPR*, 2017.

[272] H. Longuet Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, 1981.

[273] P. H. Christiansen, M. F. Kragh, Y. Brodskiy, and H. Karstoft, "Unsuperpoint: End-to-end unsupervised interest point detector and descriptor," *CoRR*, vol. abs/1907.04011, 2019. arXiv: 1907.04011.

[274] G. B. Hughes and M. Chraibi, "Calculating ellipse overlap areas," *Computing and visualization in science*, vol. 15, no. 5, pp. 291–301, 2012.

[275] H. Jegou, H. Harzallah, and C. Schmid, "A contextual dissimilarity measure for accurate and efficient image search," in *CVPR*, IEEE, 2007, pp. 1–8.

[276] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. Van Gool, "Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors," in *CVPR 2011*, IEEE, 2011, pp. 777–784.

[277] C. Urmson, J. Anhalt, J. A. (Bagnell, C. R. Baker, R. E. Bittner, J. M. Dolan, D. Duggins, D. Ferguson, T. Galatali, H. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y.-W. Seo, R. Simmons, S.

Singh, J. M. Snider, A. (Stentz, W. (L. Whittaker, and J. Ziglar, “Tartan racing: A multi-modal approach to the darpa urban challenge,” Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., Apr. 2007.

[278] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, “Junior: The stanford entry in the urban challenge,” *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, Sep. 2008.

[279] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. V. Covern, and M. Webster, “Odin: Team victortango’s entry in the darpa urban challenge,” *J. Field Robot.*, vol. 25, no. 8, pp. 467–492, Aug. 2008.

[280] S. International, “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” Tech. Rep. J3016, Jun. 2018.

[281] D. Pannen, M. Liebner, W. Hempel, and W. Burgard, “How to keep hd maps for automated driving up to date,” in *2020 International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2288–2294.

[282] M. Heo, J. Kim, and S. Kim, “Hd map change detection with cross-domain deep metric learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 10 218–10 224.

[283] A. Karpathy. (Jun. 15, 2020). “Workshop on scalability in autonomous driving: Keynote talk.” CVPR 2020, Tesla Motors, (visited on 11/15/2020).

[284] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “Nuscenes: A multimodal dataset for autonomous driving,” in *CVPR*, 2020, pp. 11 621–11 631.

[285] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, “One thousand and one hours: Self-driving motion prediction dataset,” *arXiv preprint arXiv:2006.14480*, 2020.

[286] Waymo, *Waymo open motion dataset*, https://waymo.com/open/data/motion/, 2021.

[287] G. Mattyus, S. Wang, S. Fidler, and R. Urtasun, “Hd maps: Fine-grained road segmentation by parsing ground and aerial images,” in *CVPR*, Jun. 2016.

[288] G. Mattyus, W. Luo, and R. Urtasun, “Deeproadmapper: Extracting road topology from aerial images,” in *ICCV*, Oct. 2017.

[289] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun, “Torontocity: Seeing the world with a million eyes,” in *ICCV*, Oct. 2017.

[290] N. Homayounfar, W.-C. Ma, S. K. Lakshmikanth, and R. Urtasun, “Hierarchical recurrent attention networks for structured online maps,” in *CVPR*, Jun. 2018.

[291] N. Garnett, R. Cohen, T. Pe’er, R. Lahav, and D. Levi, “3d-lanenet: End-to-end 3d multiple lane detection,” in *ICCV*, Oct. 2019.

[292] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, “Covernet: Multimodal behavior prediction using trajectory sets,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[293] D. Pannen, M. Liebner, and W. Burgard, “Hd map change detection with a boosted particle filter,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2561–2567.

[294] K. Jo, C. Kim, and M. Sunwoo, “Simultaneous localization and map change update for the high definition map-based autonomous driving car,” *Sensors*, vol. 18, no. 9, p. 3145, 2018.

[295] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, “Lifelong localization in changing environments,” *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, 2013.

[296] N. Shaik, T. Liebig, C. Kirsch, and H. Müller, “Dynamic map update of non-static facility logistics environment with a multi-robot system,” in *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, Springer, 2017, pp. 249–261.

[297] Chieh-Chih Wang and C. Thorpe, “Simultaneous localization and mapping with detection and tracking of moving objects,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 3, 2002, 2918–2924 vol.3.

[298] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, “Map building with mobile robots in dynamic environments,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, IEEE, vol. 2, 2003, pp. 1557–1563.

[299] T. Roddick and R. Cipolla, "Predicting semantic map representations from images using pyramid occupancy networks," in *CVPR*, Jun. 2020.

[300] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *ECCV*, 2020.

[301] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, "Cross-view semantic segmentation for sensing surroundings," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4867–4873, 2020.

[302] K. Mani, S. Daga, S. Garg, S. S. Narasimhan, M. Krishna, and K. M. Jatavallabhula, "Monolayout: Amodal scene layout from a single image," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020.

[303] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "Cdnet 2014: An expanded change detection benchmark dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 387–394.

[304] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, "Street-view change detection with deconvolutional networks," *Autonomous Robots*, vol. 42, no. 7, pp. 1301–1322, 2018.

[305] O. Zendel, K. Honauer, M. Murschitz, D. Steininger, and G. F. Dominguez, "Wild-dash - creating hazard-aware benchmarks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Sep. 2018.

[306] U.S. Department of Transportation, *Strong economy has americans driving more than ever before*, Press Release, https://www.fhwa.dot.gov/pressroom/fhwa1905.cfm, Mar. 2019.

[307] E. H. Adelson, J. R. Bergen, *et al.*, *The plenoptic function and the elements of early vision*, vol. 2.

[308] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *CVPR*, Jun. 2018.

[309] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *CVPR*, Jun. 2015.

[310] J. Lambert, O. Sener, and S. Savarese, "Deep learning under privileged information using heteroscedastic dropout," in *CVPR*, Jun. 2018.

[311] Z. Yang, Y. Chai, D. Anguelov, Y. Zhou, P. Sun, D. Erhan, S. Rafferty, and H. Kretzschmar, “Surfelgan: Synthesizing realistic sensor data for autonomous driving,” in *CVPR*, Jun. 2020.

[312] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from single depth images,” in *CVPR*, Ieee, 2011, pp. 1297–1304.

[313] A. Geiger, “Monocular road mosaicing for urban environments,” in *2009 IEEE Intelligent Vehicles Symposium*, IEEE, 2009, pp. 140–145.

[314] H. Zhang, M. Yang, C. Wang, X. Weng, and L. Ye, “Lane-level orthophoto map generation using multiple onboard cameras,” in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 2014, pp. 855–860.

[315] L. Rapo, “Generating road orthoimagery using a smartphone,” M.S. thesis, Lappeenranta University of Technology, 2018.

[316] L. Porzi, S. Rota Bulò, A. Colovic, and P. Kontschieder, “Seamless scene segmentation,” in *CVPR*, Jun. 2019.

[317] J. Philion, A. Kar, and S. Fidler, “Learning to evaluate perception models using planner-centric metrics,” in *CVPR*, Jun. 2020.

[318] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML ’08, Helsinki, Finland, 2008, pp. 1096–1103.

[319] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *ICCV*, Dec. 2015.

[320] R. Zhang, P. Isola, and A. A. Efros, “Split-brain autoencoders: Unsupervised learning by cross-channel prediction,” in *CVPR*, Jul. 2017.

[321] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *ICCV*, Oct. 2017.

[322] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[323] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *CVPR*, Jul. 2017.

[324] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *Journal of graphics tools*, vol. 2, no. 1, pp. 21–28, 1997.

[325] J. Lambert, Z. Liu, O. Sener, J. Hays, and V. Koltun, "Mseg: A composite dataset for multi-domain semantic segmentation," in *CVPR*, Jun. 2020.

[326] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[327] Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," in *NeurIPS*, 2021.

[328] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *CVPR*, Jun. 2016.

[329] M. El Banani, L. Gao, and J. Johnson, "Unsupervisedrr: Unsupervised point cloud registration via differentiable rendering," in *CVPR*, Jun. 2021, pp. 7129–7139.

[330] Z. Teed and J. Deng, "Tangent space backpropagation for 3d transformation groups," in *CVPR*, Jun. 2021, pp. 10 338–10 347.

[331] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, vol. 30, 2017.

[332] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, *Block-nerf: Scalable large scene neural view synthesis*, 2022. arXiv: 2202.05263 `[cs.CV]`.

[333] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[334] Y. Ma, S. Soatto, J. Košecká, and S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer, 2004, vol. 26.

[335] A. Chatterjee and V. M. Govindu, "Efficient and robust large-scale rotation averaging," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013.

[336] F. Dellaert, *Lie groups for beginners*, https://github.com/borglab/gtsam/blob/develop/doc/LieGroups.pdf, 2021.

[337] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[338] R. Szeliski, *Computer vision: algorithms and applications, 2nd Edition*. Springer Science & Business Media, 2010.

[339] S. Boyd, *Low rank approximation and extremal gain problems*, http://ee263.stanford.edu/archive/low_rank_approx.pdf.

[340] J. P. Snyder, “Map projections: A working manual. u.s. geological survey professional paper,” p. 61, 1987.

VITA

John Lambert is a researcher in the fields of computer vision, robotics, and machine learning. He is currently a PhD student in the Computer Science program at the Georgia Institute of Technology in Atlanta, GA, USA. His PhD thesis is supervised by Dr James Hays and Dr Frank Dellaert. John holds both a masters degree and bachelors degree in Computer Science from Stanford University, both with concentrations in Artificial Intelligence.