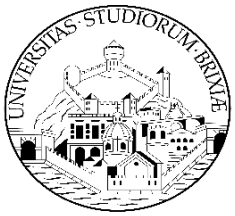


UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE
SETTORE SCIENTIFICO DISCIPLINARE ING-INF/03
TELECOMUNICAZIONI

Robust 3D Scanning and Real-Time Reconstruction Techniques in a Deep Learning Framework

Ph.D. candidate:
Marco Lombardi Matricola: **86095**

Academic Advisor:
Alberto Signoroni, Università degli Studi di Brescia

Company Advisor:
Andrea Riccardi, FARO Technologies, Brescia

Coordinator:
Prof. **Costantino De Angelis**

XXXIV cycle

Quanta cultura
ci circonda
che non sappiam
capire!
Di tanta cultura
abbisogna l'uomo,
tanta quanta
dura la vita.
Se non riceve
degli altri, il dire,
aziona le dita
e se la inventa.
Bisogno di Cultura, Ugo Mabellini

Acknowledgements

First of all, I would like to thank my PhD advisor Alberto Signoroni and my company advisor Andrea Riccardi for their incredible support. They always let me be free to choose my own path, teaching me much more than what I can express in this thesis and being inspirational for my future. I want to thank you for your wisdom and your kindness.

Then I would also like to thank Mattia and Matteo. You both were valuable as as much any real advisor can be and helped me a lot to go through these years. In particular, I thank Matteo for believing in me and giving me the opportunity to join OpenTechnologies in the first place, allowing me to understand the meaning of being part of a group. Indeed, this groups is very special to me. I cannot name you all, but you know how much I appreciate all of you and I thank you for all the help and I hope this is just a starting point for us.

Another thanks goes to my closest friends, especially Mattia and Francesco, with whom I shared the most happy moments. I own you a lot of laughs, as well as many beers!

Then, of course, I am also thankful to my mother, to my dad, to my brother and sister, and to all the rest of our wonderful family. You know I tend to keep it for myself, but you are a part of me and your support is fundamental.

Finally, I do not only want to thank the family I belongs, but also the family I've joined. Cristina, it's a fact that I would not be here if it wasn't for you. You believed me also when I did not, you gave me the confidence and ultimately you loved me and for that, I cannot do anything else than loving you back.

Compendio

La tipologia di contenuti digitali che fruiamo ogni giorno sta evolvendo e vedendo estensioni di dominio nelle quali diventa sempre più importante la componente tridimensionale, sia a livello consumer che in ambito industriale e professionale. All'interno di questa evoluzione tecnologica si riscontra quindi l'esigenza di disporre di strumenti per la transizione dal mondo reale a quello digitale che siano sempre più efficaci ed immediati. Inoltre, il continuo incremento delle capacità computazionali favorisce nuove soluzioni via via sempre più sofisticate e performanti, al contempo sempre più accessibili anche dal punto di vista economico. In questo contesto possono diffondersi (su diverse tipologie di mercato) dispositivi in grado di fornire flessibilità e accuratezza nel processo di ricostruzione 3D, mantenendo contenute le richieste di specifiche hardware.

Negli anni, la ricerca accademica ha prodotto una serie di risultati eccellenti basandosi sull'uso di quelli che comunemente vengono definiti scanner 3D ottici a basso costo, nati nel contesto delle piattaforme di gaming. Questi dispositivi sono caratterizzati da dimensioni compatte, camere di profondità con risoluzione relativamente bassa e campo di lavoro relativamente ampio. Per via di queste caratteristiche, questi strumenti trovano largo impiego in situazioni di utilizzo per ricostruzioni indoor o per applicazioni di rilevamento di oggetti, o di gestualità dove il livello di dettaglio della ricostruzione 3D non è necessariamente prioritario.

Un'evoluzione di queste tecnologie è rappresentata dagli scanner 3D portatili manovrabili a mano libera, basati su ricostruzione ottica, in grado di produrre dati a qualità più elevata rispetto alle controparti a basso costo, pur rimanendo in fasce di prezzo accessibili a livello professionale. In questo contesto è molto interessante disporre di tecniche di ricostruzione 3D real-time che assecondino e siano in grado di guidare l'azione dell'utente mediante feedback visivi immediati.

La possibilità di creare modelli complessi deputati allo svolgimento di attività a loro volta complesse è affascinante e apre le porte a diversi e nuovi scenari applicativi. Contestualmente ad un trend evolutivo in termini di hardware, l'interesse per le tematiche di ricostruzione 3D sta trovando nuove soluzioni nel settore di ricerca in forte ascesa legato alle tecniche di *deep learning*. È quindi di attuale interesse rivalutare problemi classici nell'ambito

della ricostruzione 3D da nuovi punti di vista, basati sui dati e su approcci di apprendimento guidati da questi.

L'importanza dei dati è quindi cruciale, oltre che in un contesto di valutazione sperimentale, per la necessità di fornire esempi e informazioni ai modelli che si vogliono progettare. Tuttavia, abbiamo rilevato alcune carenze legate alla tipologia dei dati impiegati nella ricerca accademica dove vi è un'attenzione prevalente legata ai dati provenenti da dispositivi a basso costo rispetto ad un più esteso panorama offerto dalle moderne tecnologie di scansione. Si tratta dunque di capire se e come le diverse caratteristiche dei dati e i diversi requisiti sulla qualità e sulle tempistiche di scansione si relazionino alla tipologia di dato generato ed alla scelta o al design delle migliori soluzioni di ricostruzione.

Durante il percorso di dottorato, ho avuto modo di lavorare con un prototipo pre-commerciale di scanner 3D portatile manovrabile, denominato Insight, sviluppato con l'obiettivo di fornire ricostruzioni con un maggiore livello di accuratezza rispetto alle controparti a basso costo, per essere utilizzato in contesti applicativi in cui il target è un singolo oggetto di scala medio-piccola di cui si vuole una fedele rappresentazione digitale. Esempi di questi contesti sono il controllo di qualità, il reverse engineering, la digitalizzazione per fini ludici (cinema e videogiochi), contesti commerciali (ad esempio cataloghi per lo shop online), culturali (preservazione di statue e oggetti storici) ed anche medicali (creazione di protesi ed ortesi).

In questa tesi ci concentriamo quindi su tecniche di ricostruzione 3D innovative, legate principalmente alla suddetta tipologia di dati, cercando di analizzare e di rispondere ai requisiti sfidanti legati a strumenti come quelli in uso durante il nostro lavoro, specialmente il requisito di ricostruzione real-time, confrontandoci con altre soluzioni disponibili in letteratura. In particolare, ci siamo preoccupati dapprima di collezionare e di rendere disponibile un nuovo dataset, DenseMatch, e di analizzare e confrontare approfonditamente, e per la prima volta, le recentissime soluzioni basate su deep learning, potenzialmente sfruttabili e fruibili nei contesti di interesse. Tale confronto avviene sfruttando sia un dataset classico che il nostro, per avere una comparazione che stabilisca quali metodi meglio generalizzano su diversi domini e quali sono i più promettenti per il nostro contesto.

Facciamo infine leva su tutti i risultati ottenuti per sviluppare un flusso di ricostruzione real-time adeguato allo scanner portatile che renda più affidabile e robusta la soluzione di ricostruzione nativa dello scanner Insight. Il nostro approccio supera nettamente la soluzione di riferimento in letteratura, denominata BundleFusion, soprattutto per la tipologia di dati e per le applicazioni di interesse. Vedremo come i migliori risultati si ottengano unendo il meglio degli approcci classici basati su feature geometriche con quelli che sfruttano i moderni modelli di apprendimento guidati dai dati.

Abstract

The type of digital content that we use every day is evolving and seeing domain extensions in which the three-dimensional component becomes increasingly important, both at the consumer level and in the industrial and professional fields. Within this technological evolution there is therefore the need to have tools for the transition from the real to the digital world that are increasingly effective and immediate. Furthermore, the continuous increase in computational capabilities favors new solutions that are increasingly sophisticated and performing, at the same time increasingly accessible also from an economic point of view. In this context, devices capable of providing flexibility and accuracy in the 3D reconstruction process can spread on different types of market, while keeping the requests for hardware specifications moderate.

Over the years, academic research has produced a number of excellent results based on the use of what are commonly referred to as low-cost optical 3D scanners, born in the context of gaming platforms. These devices are characterized by compact dimensions, depth chambers with relatively low resolution and relatively large working range. Due to these characteristics, these tools are widely used in situations of use for indoor reconstructions or for object and gesture detection applications, where the level of detail of the 3D reconstruction is not necessarily a priority.

An evolution of these technologies is represented by the hand-held portable 3D scanners, based on optical reconstruction, capable of producing higher quality data than their low-cost counterparts, while remaining in price ranges affordable at a professional level. In this context it is very interesting to have real-time 3D reconstruction techniques that support and are able to guide the user's action through immediate visual feedback.

The possibility of creating complex models for carrying out complex activities is fascinating and opens the doors to different and new application scenarios. Concurrently to an evolutionary trend in terms of hardware, the interest in 3D reconstruction issues is finding new solutions in the rapidly growing research area linked to *deep learning* techniques. It is therefore of current interest to re-evaluate classical problems in the field of 3D reconstruction from new points of view, based on data and on learning approaches guided by these.

The importance of data is therefore crucial, other than in an experimental evaluation context, for the need to provide examples and information to the models we want to design and develop. However, we found some shortcomings related to the type of data used in academic research where there is a prevalent attention linked to data coming from low-cost devices compared to a wider panorama offered by modern scanning technologies. It is therefore a question of understanding if and how the different characteristics of the data and the different requirements on the quality and timing of the scan relate to the type of data generated and to the choice or design of the best reconstruction solutions.

During my PhD, I was able to work with a pre-commercial prototype of a hand-held 3D scanner, called Insight, developed with the aim of providing reconstructions with a higher level of accuracy than its low-cost counterparts, to be used in application contexts where the target is a single small-medium scale object of which a faithful digital representation is desired. Examples of these contexts are quality control, reverse engineering, digitization for entertainment purposes (cinema and video games), commercial contexts (for example catalogs for online shops), cultural heritage (preservation of statues and historical objects) and also biomedical (e.g. anatomic scanning for the design of prostheses and orthoses).

In this thesis we therefore focus on innovative 3D reconstruction techniques, mainly related to the aforementioned type of data, trying to analyze and respond to the challenging requirements related to tools such as those in use during our work, especially the real-time reconstruction requirement, comparing ourselves with other solutions available in the literature. In particular, we first took care to collect and make available a new dataset, DenseMatch, and to analyze and compare in depth, and for the first time together, several very recent solutions based on deep learning, potentially exploitable and usable in the contexts of interest. This comparison takes place using both a classic dataset and ours, to have a comparison that establishes which methods best generalize on different domains and which ones are the most promising for our context.

Finally, we leverage all the results obtained to develop a real-time 3D reconstruction pipeline suitable for our handheld scanner that improves and makes the native reconstruction solution of the Insight scanner more reliable and robust. Our solution clearly outperforms the reference method in the literature, i.e. BundleFusion, especially for the type of data and for the applications of interest. We will see how optimal results are obtained by combining the best of classic approaches based on geometric features with those that exploit modern data-driven learning models.

Contents

Introduction

I Scenario

Deep Learning (DL) is a remarkably interesting topic which is influencing the research activities of a wide range of fields. This range includes 3D analysis, which is growing interest due to the increasing requirement of instruments for producing and consuming contents that go beyond the bi-dimensional barrier. For instance, applications such as augmented reality and virtual reality, along with the transition of sectors, as industrial and bio-medical, towards new technological paradigms, are demanding for new devices and solutions to create and to represent 3D models. Moreover, aimed by the successful results obtained by applying deep learning to Computer Vision, the community recently started developing solutions that extend the 2D-tailored approaches in order to tackle 3D related problems. Some of these problems are typically faced during the process of reconstructing a scene via 3D scanning.

In this work, 3D scanning is declined as the problem of reconstructing a scene in real-time fashion by means of an handheld optical scanner. This specific type of application is chosen due to the collaboration with Open-Technologies - FARO (Rezzato, Brescia, Italy), the company where I worked during the PhD in apprenticeship period. Indeed, at the company I had the opportunity to test a prototype of 3D scanner they are developing in order to target sector specific applications in which it is important to offer an accurate reconstruction with a fast and robust system. The prototype is still a work in progress and some critical aspects are still under review. In particular, we want to detect suitable solutions to improve the robustness of the system, which indeed is affected by several loss of track episodes, it lacks of smart solutions to adjust and refine the on-going reconstruction and does not offer an appealing feature as the recover of the tracking in the middle of the scan.

Then the final goal is to define a new working pipeline for the scanner under study. In order to achieve this, our research led us to investigate the world of deep learning to address canonical problems in 3D domain, namely 3D feature extraction for matching corresponding points which refer to multiple views and 3D registration to align them. Moreover, to proceed

with the research we also found the necessity to remedy for the lack of a dataset close enough to our working scenario. All these elements compose the final aims of the work, which we declare properly in the following section.

II Aims of the work

II.i Creation of benchmarks for high resolution handheld scanners

Recently, the widespread availability of low-cost depth sensors and the affordability of high-performance computing and graphics cards helped to achieve significant results in the field of real-time reconstruction. This kind of devices are appealing for the market because of the advantages they offer compared with static scanners. In fact, these scanners allow constructing a 3D scene by relying on a fast and flexible solution: the user is free to move with no fixed path to follow, while a constant feedback is returned live, so that it can show the growing model to help to understand the areas still missing from the scan.

However, it is important to notice how these devices usually focus more on providing fast and robust reconstruction systems at the expense of a certain level of accuracy. Such a trade-off is well suited for specific applications like indoor reconstruction. In contrast with this scenario, many other applications can be targeted with handheld scanners, but they usually require a higher accuracy and data fidelity. Examples of these applications are for instance reverse engineering for product design, for quality assurance, cultural heritage or even medical CAM/CAD-related as orthotics. Indeed the recent developments move towards an improved accuracy for low cost optical devices but the literature still lacks of examples for this kind of objectives. Therefore we aim to fill the void by proposing a reference dataset that can be useful for future researches.

Exploiting the prototype available for this research, we create a novel collection of scenes representing subjects which are typically targeted in several application context, such as quality inspection, modeling for commercial and ludic purposes, reverse engineering and body scanning. The collection comprises both the source data generated with the scanner, *i.e.* the stream of frames RGB and the associated DepthMaps, and the point clouds extracted from these frames, properly aligned by hand to provide a reliable ground truth.

We state that this dataset can be used to develop new strategies to address different tasks that deal with high resolution optical scanners input data. Moreover, it can be used by computer scientists as a benchmark to perform both quantitative and qualitative analysis and comparisons. Finally, it can be used to train or fine-tune AI models to cope with small-scale and dense 3D object acquisitions and reconstructions.

Indeed, in this work we will refer to it for our own assessment.

II.ii Deep Learning applied to 3D registration of dense data

Real-time optical scanners [1] are a powerful and flexible tool for 3D reconstruction. In essence, they need to rely on fast and robust techniques to align the partial views they acquire at a high rate during the scan. Many traditional methods addressed the challenges through the years and some solutions became the standard de facto to deal with live reconstruction-related problems. However, no solution is flawless and the research continues by exploring new paths every day. A promising example of such a new path, is represented by the novel introduction of Deep Learning to tackle 3D problems.

Early attempts tried to leverage standard 2D deep learning solutions by translating the problem to 2D domain via projection mapping [2, 3, 4, 5]. Later, a structured representation of 3D space [6, 7, 8] was adopted to extend the algorithms effortless. However, these solutions require a high level of quantization and partially lose local information which can be valuable instead. Conversely, PointNet [9, 10] is the first attempt to deal with unstructured data to feed a network able to process a dense point cloud. Despite the technological step, the method suffers from several limitations, such as the curse of dimensionality and poor understanding of local context. 3D convolutional neural networks (3D CNNs) are then an appealing alternative because of the advantages of convolutional layers which mitigate the issue arose with PointNet. However, 3DCNN can not be applied directly on point clouds due to the unstructured nature. Sparse convolution [11, 12] has then been introduced to cope with this problem. By means of sparse convolutional layers backends, promising works have been recently introduced in relation with tasks which are meaningful to us, such as feature extraction and false correspondence rejection to infer the alignment transformation by exploiting valid matches only.

3D local feature extraction 3D local feature extraction commonly requires the preliminary detection of interesting points to use as an anchor during the definition of a local reference frame. Such a reference frame is necessary to define the descriptor representing the feature invariant to changing views and eventually to varying scale and local points density. Usually, the descriptor refers to spatial and geometrical relationships internal to the neighborhood. Each descriptor favors a specific property, therefore it is more suitable for specific applications only. In general, there is no standard method able to generalize well and most of the solutions can be exploited only on a sparse set of data due to low compactness and time consuming estimation [13]. Recent 3DCNN-based model [14, 15], conversely, allows to

exploit the whole set of points to extract geometric features from the full dense point cloud.

3D registration 3D registration can be classified according to multiple criteria. Our main focus is on pairwise rigid registration because it allows the tracking of a moving camera in a scene while scanning the model itself. Such a problem is usually split in different steps: a corresponding set of features is detected across two views. The matches are refined by pruning the outliers and the sparse set of correspondences is used to retrieve the coarse registration matrix which aligns the two. Finally, a last stage exploits the dense set of points to refine the alignment.

Most prominent DL-based contributions for 3D view alignment appeared very recently and almost concurrently [16, 17, 18]. They all tackle the task as a classification problem and weight the correspondences set according to their accuracy. By means of this weighting vector they then regress the transformation.

Due to their contemporaneity, an in-depth comparison among these methods is lacking, therefore we aim to fill the void by analyzing the opportunities they offer, as well as the eventual limitations.

Moreover, all of these technologies have been proposed and tested on data from low-cost scanners and with medium-large scales (about 1 to 10 meters). Therefore, there is the need to understand if the same solutions are equally effective when dealing with data acquired with hand-held scanners targeted for quality reconstruction of single objects/subjects rather than entire scenes/environments, and operating on smaller scales and ranges (from cm to few meters).

II.iii A Deep Learning framework for real-time 3D scanning pipeline

The DL-based methods we deal with during our dissertation have all been introduced with the goal of replacing handcrafted solutions to address classical problems. Up to our knowledge, none of these technologies have yet been introduced and tested in a real-time reconstruction framework. Thus, there is the need to understand their possible role and practical contribution especially in solving open issues affecting this demanding kind of reconstruction.

Throughout our research, we found that BundleFusion [19] is the state-of-the-art for medium and large scale 3D reconstruction. Most of its contributions are appealing to us because it offers a robust solution which is also able to adjust and refine the reconstruction on-the-fly. However, our experiments have shown that the registration strategy adopted by BundleFusion tends to struggle with our own data. Our investigation led us to assume that relying exclusively on 2D features for coarse matching can be tricky in our

context. Indeed, conversely to the situation addressed by BundleFusion, we target the reconstruction of single objects for which the frames we take can present regions with repetitive pattern or lack of significant texture to rely on for 2D feature matching.

Therefore, encouraged by the results we achieved from the preliminary evaluation of DL-based methods, we try to exploit the method proposed in BundleFusion by replacing the critical elements with some promising data-driven solutions, in order to combine the best of the two worlds. The final result is a new pipeline we developed in a DL framework that can perform a robust real-time 3D reconstruction. The key features of this pipeline are:

- A robust 3D registration module that is based on data-driven geometric feature extraction and coarse registration methods which are properly tuned to deal with high resolution 3D data for object reconstruction. The module is used to deal with the issue of tracking loss and camera relocation.
- A hierarchical structure to split the problem of frames alignment both locally and globally, operating within the scope of real-time reconstruction.
- A pose refinement module, similar to what is proposed in BundleFusion, which leverages the hierarchy division for a feasible adjustment.

In order to evaluate our pipeline we perform an extensive analysis by testing the benchmark dataset we presented earlier and we compare against two standard solutions (KinectFusion [20] and BundleFusion [19]) either with our dataset and with other common datasets as well. Finally, we point out that such a pipeline relies solely on open source libraries, therefore we are able to provide an open version of the code to the community. We think that this is a first contribution in the direction of streamlining real-time 3D reconstructions with the help of data-driven technologies that can be of help to investigate more advanced solutions in the same direction and to extend their application scope.

III Document organization

This document is organized as follows. In Chapter 1 we introduce the reader to the topic of 3D scanning. The chapter is preparatory to understand the challenges we encounter when dealing with real-time 3D reconstruction and what opportunities an handheld optical scanner can offer, especially when paired with high resolution cameras to target high quality object reconstruction. At the end of the chapter we clarify the motivations behind this work and we summarize the different stages it is composed by, which are then reflected in the structure of the document. In Chapter 2 we focus on the

device we tested throughout the entire PhD program, explaining the key characteristics and the challenges we faced to improve the user experience. At the end of this chapter we introduce a dataset we created with this scanner which is then used to assess novel deep learning methods we introduce at the beginning of Chapter 3. All of these evaluations lead us to propose in Chapter 2 our own reconstruction pipeline which exploits deep learning models to deal with some critical components and offers a new framework to tackle real-time 3D reconstruction via optical 3D scanning.

Chapter 1

3D scanning and reconstruction: traditional approaches and benchmark datasets

With 3D scanning and reconstruction (or, more concisely, 3D scanning) we refer to the process of geometry information retrieval from a real world scene (as well as additional gathering, such as the surface color information). The goal is to create a numerical representation of the scene that can be processed into a digital framework for several applications. The scene can qualify to be indoor/outdoor or to represent a single/collection of objects. In this chapter we introduce the reader to topic of 3D scanning. To this, we set the ground on multiple levels: we first discuss 3D scanning from an hardware perspective by describing what kind of sensors are typically used. Then we take a look at what data collections are accessible in literature, in order to test novel approaches and benchmark different solutions. Finally, we review the state-of-the-art of multiple-view 3D registration and we introduce the best pipelines currently available to achieve real-time 3D reconstruction. It is worth to mention here that in this chapter we will discuss about the traditional approaches only, meaning we deepen the handcrafted methods which are based upon standard heuristics, handcrafted features, and most popular real-time reconstruction algorithms not driven by neural networks decisions. Indeed, the recent introduction of Deep Learning to 3D processing, comprising its possible adoption for real-time 3D reconstructions will be the objective in Chapter 3 later in the document.

Specifically, this chapter is organized as follows:

- An overview of the most common types of 3D scanners which are available in the market is given to the reader in the Section 1. In particular,

the last paragraph is entirely focused on handheld optical scanners. As we mentioned, such a type of scanner is meaningful to our research. Therefore, in order to introduce our own device (we do it in Chapter 2) we first want to present the most similar consumer solutions available nowadays. By listing those devices, we point out the strengths and the weaknesses of this class of products.

- Section 2 is dedicated to the data. First we make a list of the most common benchmark dataset we have in literature, describing the source data input, the main specifications and the target they address. Then we focus on two specific datasets, which will be useful for cross-referencing our work with other popular solutions targeted towards real-time reconstruction.

- Section 3 is focused on the methods instead. Specifically, we divide the section in two parts. The first topic is an introduction to 3D registration, which is fundamental to track a camera during the scan. The second part focuses on the pipelines that drive the aforementioned optical scanners to perform real-time 3D reconstruction. We present the most relevant solutions and we highlight the open challenges. Indeed these challenges also affect our own device and methods. Therefore in the next chapters we will seek to understand how recent data-driven approaches can favorably contribute to tackle them.

1 3D Scanning Devices

Depending on the role that each sensor plays during the acquisition process, we can divide the 3D scanners in two macro areas: based on *active sensors* or on *passive sensors*. As the nomenclature suggests, the distinction is made upon the fact that a sensor can either interact with the target surface or it can simply not. Moreover, additional distinctions can be made according to the technology which is adopted by each sensor. In the following section we make a distinction for the most representative types and we briefly cover all of them by providing meaningful information for each solution, such as pros and cons, price range and applications they are meant for.

We deliberately decide to keep the review concise and functional to the objective of the thesis. Indeed, the introduction is preparatory for understanding the context around the scanner we will analyze in the following chapters. For more technical details we refer the reader to more specialized works. For instance, for the passive system we present in Sec. 1.1 we refer to [1, 21, 22, 23, 24, 25] while for active stereo system we refer to [1, 26, 27, 28, 29]

1.1 Passive sensors

Passive 3D imaging solutions rely on detecting reflected ambient radiation, mostly visible light but not only (also infra red is a possible option). Because of this, the sensor can usually be as simple as a digital camera, with the addition of particular filters when necessary. Granting us a bit of terminology relaxation, we overlap the concept of passive depth sensing by means of optical instruments with the wide discipline of *photogrammetry* [30, 31, 32, 33]. Indeed, depending on the type of solutions adopted to process the images, several independent research fields are involved. For the sake of readability and according with the level of deepening we maintain for this preparatory introduction, the main distinction we make is based upon the number of sensors (and views) required to reconstruct the geometry of a scene. Therefore, in the following paragraphs we review the macro topics of single-view and multi-view photogrammetry and we add an additional paragraph for stereo configurations special case.

1.1.1 Single-view photogrammetry

Estimating depth map or 3D information from a single view is a common problem in computer vision [30]. Instances of these methods include linear perspective [34], atmosphere scattering [35], patterned texture [36], symmetric patterns [37, 38] and statistical patterns [39, 40].

Figure 1.1: Example of shape from texture. From left to right: original image, segmented texture region, surface normals, depth map, reconstructed 3D shape. Image taken from [36]

From a more strict photogrammetry perspective, single-view 3D reconstruction is a kind of linear perspective method which uses straight lines in the object space to obtain image rotations (Euler angles), interior orientation parameters of the image (focal length, and principle points). Relying on this information, the distances can be calculated by means of various mathematical methods [41]. Overall, the method can be extremely cheap, and usually the data processing can be done by non-specialist users. However, the reachable accuracy is not suitable in many applications. Moreover, single image photogrammetry cannot be used for 3D modelling of an object. Indeed, in order to determine ground points and obtain a 3D model, both the exterior orientation elements and the scale factor of every bundle ray should be known (the scale factor is not necessarily stable for different regions of an

image). This problem is solved by using binocular photogrammetry.

1.1.2 Binocular photogrammetry

Binocular photogrammetry is also known as stereo-view photogrammetry. The term *stereoscopic* refers to the fact that the method this system is based on has the same principles driving human stereoscopic vision. In Appendices Sec. A we provide more context and we give some elements to understand the geometry behind such method, so we do not go much further with details here. The reason why we dedicate an appendix section to the stereo vision is because it is the basis also for another important type of scanner, specifically the structured light handheld one which is also covered in the following section dedicated to the active scanners. In brief, stereoscopic system employ two cameras slightly apart (the distance between the two is referred to as *baseline*). The cameras are tilted to look at the same scene. The relationship between two views of the same spot taken from different angles is expressed by means of the epipolar geometry. By analyzing such a relationship it can be determined how far the point is from the center of the camera. It is then possible to build a depth map which assign a `z` coordinate to each pixel in the image and use the depth information along with localization of the camera in the system to retrieve the 3D map of the scene. **Pros** The solution is potentially very cheap but it can offer a high accurate reconstruction under specific requirements of illuminations of the scene, materials in it and proper system calibration. **Cons** Indeed it is sensitive to reflections, light speckles, etc. For this reason, external projectors are usually adopted to deal with environmental light source. Moreover, the baseline of the camera is critical to determine the field of view of the scanner. A larger field of view tends to suffer more from occlusions and clutter.

1.1.3 Multi-view photogrammetry

These systems are usually composed by an array of cameras that are synchronized to acquire several pictures of the target object from multiple locations. Those images are then processed using optics and projective geometry to finally retrieve the 3D scene information. Moreover, when particular light conditions are preserved, only one camera is sufficient for the goal. In this case the camera device is moved around, taking multiple shots from different angles. Many variants of algorithms exist to exploit the multitude of images. Shape can be derived from varying focus [42], from silhouettes [43], from motion. In Fig. 1.2 an example of the 3D reconstruction of a person with a photogrammetric system is shown. The polygons floating in the scene represent the camera positions where each of the pictures were taken.

Applications Multi-view photogrammetry is the core technology for several applications such as manufacturing, quality control, police investigation

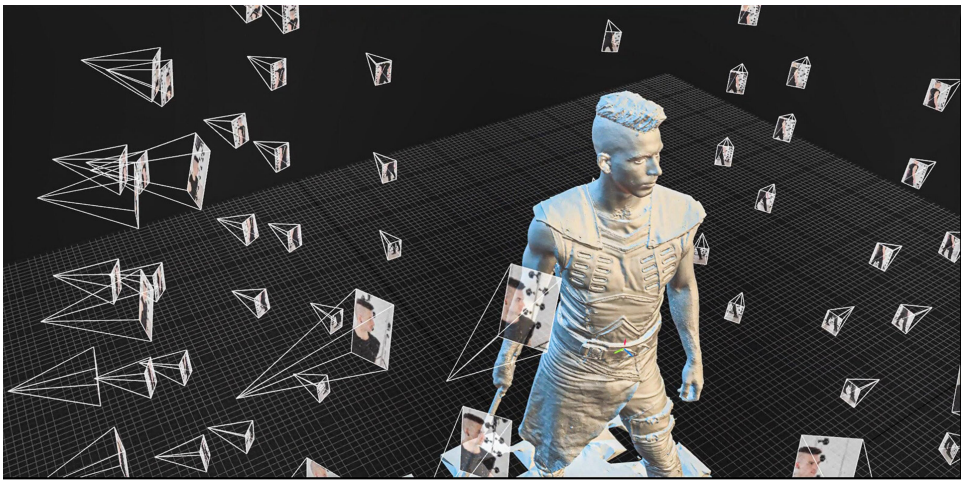


Figure 1.2: Example of 3D reconstruction of a human body with a photogrammetric system. Original image from `https://xangle3d.com/`

and cultural heritage. However it can also be used in different contexts, in fact it can be exploited for topographic mapping, architecture and geology. **Pros** The most sophisticated systems can reach sub-millimeters accuracy level. **Cost** Their cost spans a large interval which depends upon the system configuration. The simplest solution requires only a digital camera (even a smartphone is valid for very coarse reconstructions). On the contrary the most complex solution works with a cluster of high quality digital cameras along with a set of coded markers and reference scale bars placed around the scene in order to increase the robustness and camera tracking accuracy. **Cons** The reconstruction is performed offline and it can be time consuming, even with modern computers. Since it is offline, the user has no live feedback during the scan. Therefore it is common to deal with missing parts after one round of acquisitions. The scan can then be tedious, especially when only one camera is available, since it is usually necessary to repeat some shots. Finally, the light conditions in the scene should be set up properly to ensure quality results, which also increases the overall time-to-data experience.

1.2 Active sensors

1.2.1 Contact-based

As intuitive as it could be, the first example of active sensors is from the contact-based family, *i.e.* those scanners that retrieve the surface information of an object by physically touching it with a probe. These scanners are known as coordinate measuring machine (CMM). The measure is given according to the reference system of the beforehand calibrated probe which is constantly *aware* of its own placement in the world. Three variants of CMMs are

currently available on the market:

- A fixed system in which the probe is attached to a rigid arm and the inspected object is placed underneath it. This type of system is usually bulky but also highly accurate (in the order of few micrometers) and can completely automated via Direct Computer Control (DCC).
- A portable system with an articulated arm (Fig. 1.3) which is less bulky and it is ideal for probing into fissures and holes and other complex geometries.
- A mix of the previous solutions in which both the rigid and the articulated arms are deployed.

Application CMMs are mostly used for comparing parts against design intent, a common procedure in manufacturing process in different industrial sectors. **Pros** The main advantages are the high accuracy of the measure and the possibility to automate the scanning process. **Cons** On the other side, both the carriage system with rigid arm and the portable system with articulated arm have rigid stability constraints on the scanning environment (the system needs to be fixed during the acquisition). Moreover, the interaction with the object under inspection can be critical since the object could be damaged by the contact with the probe. Finally, the whole process of acquisition is typically slow if compared with other solutions. **Cost** The cost ranges from 30k euros for the entry levels up to easily more than 150k euros according to specifications.

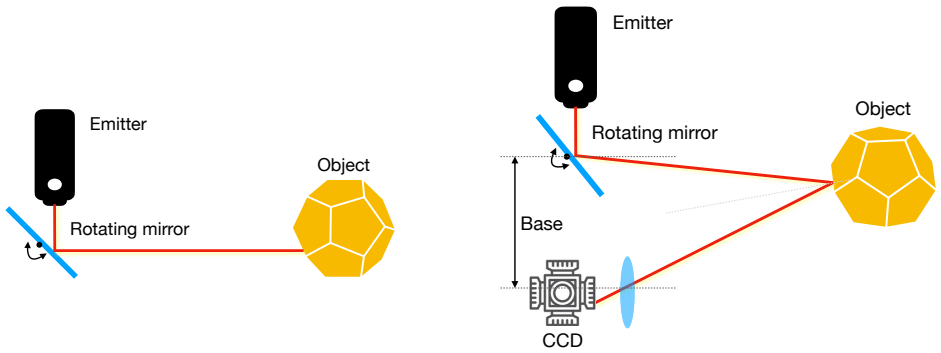
(a) (b)

Figure 1.3: Example of a Coordinate Measuring Machine. (a) Articulated arm with rigid bones in action. (b) Detail of the probe.

1.2.2 Laser-based

Laser scanners are versatile solutions that offer a large pool of options in terms of applications that can be addressed with. The core technology is

the source monochromatic laser beam that is obtained through a controlled electromagnetic radiation. Such a laser beam is then employed in different ways to sense the target surface. The two main configurations are briefly described in the following paragraphs.



(a) Range finder (b) Triangulation

Figure 1.4: Laser-based scanner configurations.

Laser range finder Usually known also as LIDAR [29, 27, 28], such a scanner works under one of two measurement principle: *time-of-flight* measurement or phase shift measurement. The former evaluates how long the emitted beam takes to bounce back after hitting the target object (Fig. 1.4(a)). The rationale is that the speed of light being a constant, it allows the distance to be evaluated as the product between the velocity of the beam and its round-trip time. In alternative, the phase shift can also be measured. Due to the wave nature of the emitted light, if the original wavelength is known, the a distance measure can be derived by measuring the difference between the phase of the emitted wave and the phase of the reflected one. Moreover, in order to scan multiple points on the target surface, the beam needs to be moved over the target scene or object. A common practice to achieve this is to use mirrors to deflect the beam. Indeed the mirrors are lightweight and allow for very accurate deflections when a sufficient quality of the optics is guaranteed. Moreover, they can be rotated at a high rate, allowing the scanner to acquire up to millions of points per second. Each of this points is collected creating a *point cloud* data structure. Multiple point clouds are finally aligned to create the complete 3D model. **Pros** Overall, the stability of the laser beam is one of the key properties for this type of scanners: a point can be acquired few centimeters away from the scanner, up to hundreds or even thousands of meters of distance. **Application** For this reason, the most common application fields are remote sensing and building information modeling (BIM [44]) (Fig. 1.5). **Cons** These solutions are usually expensive and mostly oriented to professional applications. More-

over, the accuracy and the point density can be sub-optimal with respect to other solutions: indeed, they can suffer the competition of passive stereo systems for medium and large scale scenes (*e.g.* indoor and outdoor scanning) reconstruction while structured light solutions can perform better on small scale reconstruction (*e.g.* single object scanning). **Cost** Due to the high standards required for optical and mechanical manufacturing, the entry level class starts from nearly 40k euros.

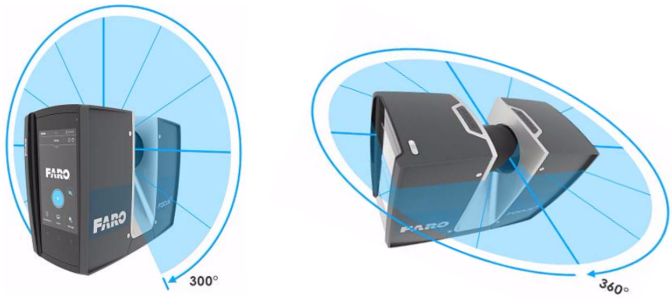


Figure 1.5: Example of a range finder laser scanner. A rotating mirror mechanism spans the laser beam on x and y axes to create a point cloud.

Triangulation based laser scanner Instead of elaborating how the beam traveled from emitter to sensor, triangulation based laser scanner rely on a camera sensor (*e.g.* a charge-coupled device, CCD) to evaluate how the beam hit the surface. The name is due to the fact that emitter-camera-dot configuration compose the three vertices of a triangle (Fig. 1.4(b)). The distance between laser emitter and camera sensor is given at calibration stage, while effective methods of Computer Vision [30] are involved to detect the laser dot in the image. In practice, in order to speed up the acquisition process a laser stripe is projected instead of a single dot. Then, every single acquisition provides a set of sparse points, which are referenced to the laser-camera system. In order to align each set to create the final model, it is necessary to keep track of the scanner during the acquisition process. Then, a common choice to provide a reference to the system is to place markers all over the scene, since they are easy to detect and to track throughout the scan session. As an alternative to markers, natural features and direct geometry based alignments can be exploited for tracking. However, the former requires to work with a rich texture (few symmetries, presence of corners, edges, etc.) while the second needs to increase the number of stripes to leverage on a slightly more dense geometry from a single shot. Nevertheless, those solutions tend to be less accurate than the marker-based one. In practice, the triangulation-based methods are largely adopted in building *handheld scanners*. **Pros** Nowadays handheld laser scanners can reach a metrology level accuracy when matched with targets (*i.e.* in the order of micrometers).

They can be also relatively lightweight (top class weight around 1kg) which is good enough for rapid scanning sessions. **Cons** In contrast with time-of-flight scanners, their range is extremely short, even below 1m: in fact, it is necessary to work within a short range to keep the beam narrow enough to avoid sparkles and reflections that could affect the accuracy of dot detection. **Application** Many applications are available with this system, as for instance shape inspection for quality assurance, reverse engineering, cultural heritage, etc.. **Cost** Their cost spans within a large range (from 10k up to 70k euros). Such a large margin depends upon multiple aspects, such as the accuracy, the amount of scanning modes available (*e.g.* working with markers, without markers, providing additional texture information), the number of points that are processed per second, the weight, the connectivity, etc..

1.2.3 ToF cameras

A common solution for depth sensing relies on time-of-flight (ToF) cameras [45, 46]. The core idea is similar to LIDAR, meaning it is based on a time-delay algorithm. However, conversely to LIDAR, every pixel is a measurement point with ToF cameras, therefore the reconstruction is dense, regardless of scene texture. One very popular ToF camera is Kinect2 (see Fig. 1.6). **Pros** The advantages for ToF cameras comprise the dense and

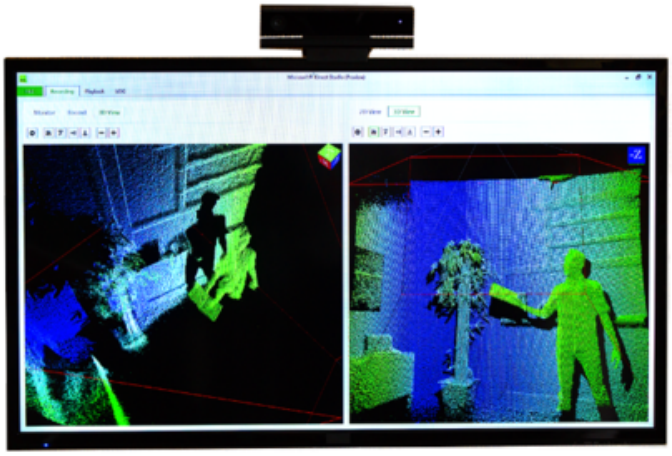


Figure 1.6: Kinect2 Time-of-Flight sensing camera (image taken from `https://developer.microsoft.com/it-it/windows/kinect/`).

instantaneous 3D reconstruction which is suited for real-time applications. They are cheap and robust due to the lack of moving parts. For these reasons ToF cameras are easy to use also by non-expert users. **Cons** The main limitations are the fact that are susceptible to multi-echo returns distorting data and a relatively low resolution. Moreover, noisy input requires long integration time to cope with motion blur and resolve range ambiguity. **Ap-**

plication They are generally used in indoor environments and address short range object scanning [47, 48] and gesture recognition [49, 50, 51].

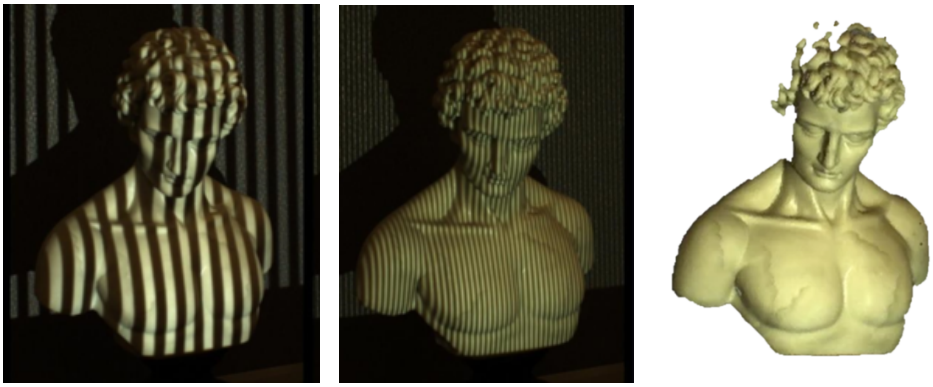
1.2.4 Structured light-based

Laser scanners are limited by the fact that the laser beam can scan only one point or one line of points at a time. Although fast movable mirrors are available, as well as line-arranged beams can be used to substitute the dot with a stripe, overall the set of points that is acquired tend to be more sparse compared to an ideal density for good 3D representation. The same is for contact-based scanners. All these methods are also inherently sensitive to the problem of distortion from motion, which is caused by non rigidity of the target throughout the single view acquisition. On the contrary, structured light 3D scanners perform a *dense* single view acquisition by scanning the entire field of view (FOV) at once. In order to illuminate the scene these scanners do not use laser but LCD or infra-red (IR) projector instead. In Fig. 1.7 we report an example of the scanner mounted on a tripod with additional turntable to make the object spin around itself at a controlled angular intervals. Nevertheless, structured light scanners can also be conceived in handheld configuration. In any case, the structure of the projected

Figure 1.7: Example of structured light scanner with optional turntable.

pattern is known *a priori* by the system which then estimates the geometry of the object surface by evaluating how such pattern is deformed (Fig. 1.8). In practice, to ensure the robustness of the method a pattern is projected either using a spatial or a temporal encoding [30]. An example of successful encoding pattern is Gray code [52].

Pros Structured light scanners are an accountable solution because of their speed (fast acquisitions especially when single projected patterns are



(a) (b) (c)

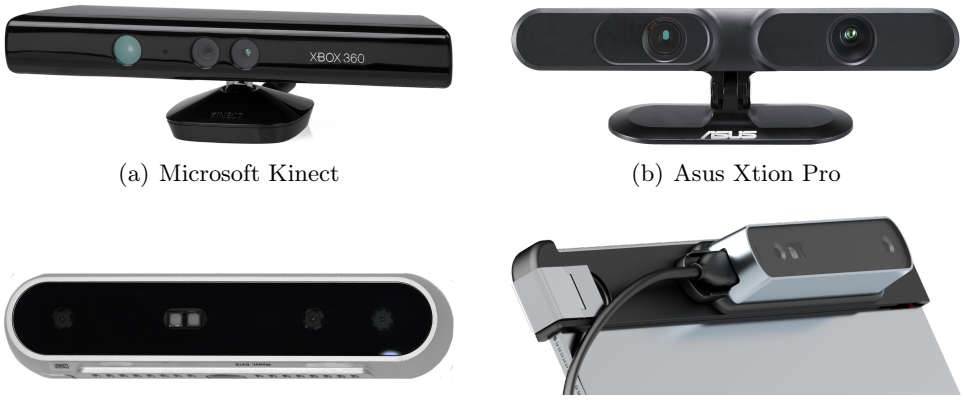
Figure 1.8: Example of structured 3D reconstruction with structured light 3D scanner. (a-b) Binary code projected (c) Final model.

used) and resolution (dense acquisitions, especially when high resolution cameras are used). The accuracy spans from tens of micrometers to millimeters depending on the optics configuration and the context application. **Applications** Speaking of applications, they can be used for industrial metrology, shape measurement for production control, body shape measurement but also simultaneous localization and mapping (SLAM) technologies, as we will see in the next section. **Cons** Similarly to other optical-based solutions, one of the most relevant problems is related to light reflection and deflection. The method suffers from scanning challenging surfaces such as metal and glass which are either highly specular or semi-transparent. **Cost** The cost range is again wide. However, optical scanners tends to be not too expensive with respect to other solutions. In particular, there are several scanners with a very low entry price, sharply below 1k euros. Indeed, in the next section we cover some of the most common scanners available in this cheap sector, since they are well known in the academic community and they are employed to create shared datasets for research purposes. In the next chapter we shift the attention on higher performance devices, which open to relevant but more demanding applications.

1.3 Handheld optical scanners available on the market

In the previous paragraph we introduced structured light scanners. We mentioned the fact that the hardware required to power such systems can be relatively low-cost. Indeed, in this section we overview some of the main devices available at consumer end. These scanners come in a very affordable price range, which makes them appealing for research activities. Moreover, the working pipeline that powers them, is very similar to the pipeline we first

developed for our own case study device, which we will deepen in Chapter 2. Pictures of the most popular devices are given in Fig. 1.9 while in the following paragraphs we give some historical context and we describe in detail the technical sheet of each of them. Finally, in Tab. 1.1 we extend the list to few more meaningful models and we summarize the main specifications to better organize such information for the reader.



(a) Microsoft Kinect (b) Asus Xtion Pro

(c) Intel RealSense d415 (d) Structure IO

Figure 1.9: Examples of low cost optical devices.

Microsoft Kinect It was presented in 2009 as an accessory of the Microsoft console gaming XBOX, making it the first mainstream motion sensing device for less than 200 euros. It has two cameras. The first camera is an RGB camera working at 30 fps with a resolution of 640×480, while the other camera has lower resolution of 320×240 and a near-infra red band pass filter which is paired with an projector. This makes the device, in its first configuration, a single pattern structured-light scanner (projector + IR camera) with the possibility to create colored point clouds (by the RGB camera). The working distance spans from 0.4m to 4.5m, with a field of view of 57 and 43 degrees on horizontal and vertical plane respectively. Those numbers indicate that the Kinect has a relatively large field of view. Microsoft released a second version of Kinect in 2013 even with a higher resolution for each the two cameras, but it also changed the core technology by shifting it to a time-of-flight scanner (see Section 1.2). While the Kinect group has dismissed the support for playing with the Microsoft console, the scanner has found new life in academia, especially after the introduction of seminal works such as KinectFusion [20, 53, 54].

Asus Xtion Pro Surfing the wave of marketing affordable sensors for motion tracking, other big tech companies besides Microsoft launched their

own devices. Asus for instance presented a scanner which is almost identical to the first version of Kinect: same specifications for the cameras, same infra red projector and similar field of view but narrower working range (up to 3.5m). All these components are engineered into a smaller enclosure ($18 \times 3.5 \times 5$cm against $28 \times 8 \times 8$cm dimension for Kinect) and sold for less than 150 euros.

Intel RealSense Another very interesting option is the D series (where D stands for Depth) from Intel RealSense products. The lineup has several configurations including rolling and global shutters, different baselines and additional Inertial Measurement Units (IMU). overall all the devices are incredibly lightweight and compact (D415 is $9.9 \times 2.0 \times 2.3$cm) which is also the main difference with the other competitors from Microsoft and Asus. Besides the enclosure, the depth-sensing setup is similar to the former two scanners: a RGB camera for texture information and a IR camera paired with a infra red projector are used. The D415 is the most accurate scanner in the current lineup, since its depth camera sensor has a resolution up to 1280×720 which makes it the preferable choice for high fidelity reconstruction, while other devices like D435i are preferable for other task such as object detection and fast tracking (having global shutter camera and IMU on board). Finally, Intel has made a lot of effort in the past few years for opening their system to the academic research: along with the scanner it offers computer vision libraries and SDKs which are also supported by Open3D [55], a trend popular 3D library maintained by Intel itself.

Structure Sensor The scanner from Occipital, Inc. is a slightly different kind of device: originally meant for being mounted on iPad because of its compact form factor ($10.9 \times 1.8 \times 2.4$cm) now it can be fully integrated with other operating systems too. The scanner itself is very similar to RealSense devices and it offers all the advanced features: a on-device processor, a high quality depth infra red camera (1280×960 running at 54 fps) with a large working range (from 30cm up to 5m), infra red class 1 projector (for eye safety), RGB camera (640×480 at 100 fps) and a 6-axes IMU.

Scanner Type	Technology	RGB Res	Depth Res	Z range	FOV (HxVxD)	Size [mm]
Kinect v1	active IR stereo	640×480	320×240	0.4 - 4.5m	$57° \times 43° \times ?$	$289 \times 73 \times 71$
Kinect v2	active IR ToF	1920×1080	512×424	0.5 - 4.5m	$70° \times 60° \times ?$	$249 \times 66 \times 67$
PrimeSense	active IR stereo	1280×960	640×480	0.4 - 3.0m	$58° \times 45° \times ?$	$180 \times 25 \times 35$
Xtion Pro	active IR stereo	640×480	320×240	0.4 - 3.5m	$58° \times 45° \times 70°$	$180 \times 35 \times 50$
RealSense R200	active IR stereo	1920×1080	640×480	0.3 - 3.5m	$59° \times 46° \times 70°$	$130 \times 20 \times 7$
RealSense D415	active IR stereo	1920×1080	1280×720	0.3 - 3.5m	$65° \times 40° \times 72°$	$99 \times 20 \times 23$
Structure IO	active IR stereo	1920×1080	640×480	0.4 - 3.5m	$58° \times 45° \times 70°$	$119 \times 28 \times 29$
Structure Core	active IR stereo	640×480	1280×960	0.3 - 5.0m	$59° \times 46° \times 70°$	$109 \times 18 \times 24$
MS Azure	active IR stereo	1024×1024	4096×3072	0.3 - ?m	$90° \times 70° \times ?$	$103 \times 39 \times 26$

Table 1.1: Main specifications for the most popular low cost motion-sensing devices. Not found specifications were replaced with "?".

2 Benchmarking 3D Datasets

In order to evaluate new solutions and to create reference benchmarks for them, it is necessary to rely on common baseline datasets. If we take a look at the current proposal in literature, it is clear that 3D domain is not as florid as 2D in terms of research data production. However, the low price of 3D cameras has increased exponentially the interest in using depth information for solving vision tasks. In this section we highlight the most important datasets for addressing 3D vision related challenges. In particular, our investigation goal is to find a dataset which is close to the data type produced by our own device (see Chapter 2) and is suitable to represent the scenario of an accurate object reconstruction application, which is the main target of our study (Chapter 3 and 4).

Most recent 3D reconstruction performance comparisons involve the following datasets, presented in decreasing order in terms of scene scale (or target size).:

- KITTI [56], comprises either laser and RGB-D scans and contains large volume point clouds that can not be used in the context of rigid registration because to the presence of moving elements. Indeed it targets several applications related to autonomous driving, such as topology mapping, object detection, and pedestrian tracking and it is widely adopted in the research community.

- 3DMatch [6] instead aims at providing examples of indoor scenes acquired via optical handheld scanners. Such a dataset is also a standard in the field of indoor reconstruction application and it is indeed the main benchmark dataset among most of the works we end up wanting to compare with. For this reason, it is one of the two datasets that are covered with more details below.

- ModelNet, a subset of ShapeNet [7], is a rich object oriented dataset which includes a large dictionary of objects from a miscellaneous set of data, especially CAD models. Indeed ShapeNet is a benchmark dataset for several applications including object classification, semantic segmentation, object detection, etc. but it is based primarily on synthetic data while we are more interested into evaluating real 3D scans for our study.

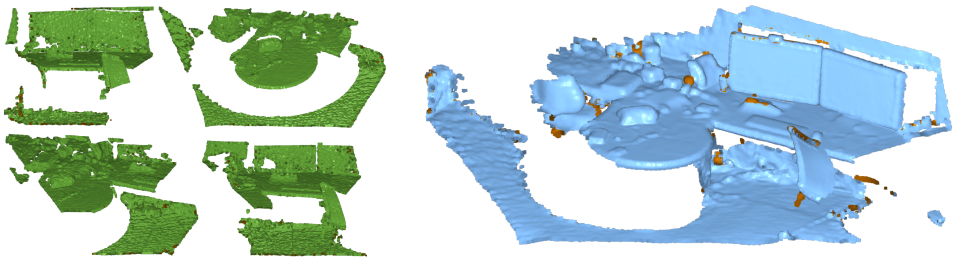
- The final dataset, which is in theory the closest one to our working scenario and the second we deepen in the following paragraph, is the Redwood dataset [57], which offers a collection of real object scanned with a low cost scanner as well as 3DMatch.

2.1 Indoor 3D scene reconstruction: 3DMatch dataset

Figure 1.10: Collection of scenes in 3DMatch dataset [6].

3DMatch [6] is a large collection of 3D point clouds used for indoor reconstruction analysis, object detection and shape retrieval. Actually the dataset is a combination of several RGB-D datasets produced by the research group of Princeton. Each of these dataset is created using one of the low cost device presented above: 7Scenes dataset [58] and RGB-D Scenes v2 [59] created with Kinect; Sun3D dataset [60] created with Asus Xtion Pro; Analysis by Syntesis [61] and BundleFusion [19] created with Structure.IO.

We discussed that the selected scanners output a sequence of noisy and low resolution RGB-D images. In order to weight the sensor noise, to have a more dense data and finally a reliable 3D model with sufficient geometry information, each point cloud (blue one in Fig. 1.11) is created via volumetric integration [62] by merging 50 consecutive frames (4 of them are shown in green in Fig. 1.11). All the details and the code to recreate the training and testing set are available in the project repository[1].



(a) (b)

Figure 1.11: 3DMatch dataset, `office3` scene: (a) example of single frames and (b) a point cloud in the scene, obtained by fusing 50 of such frames.

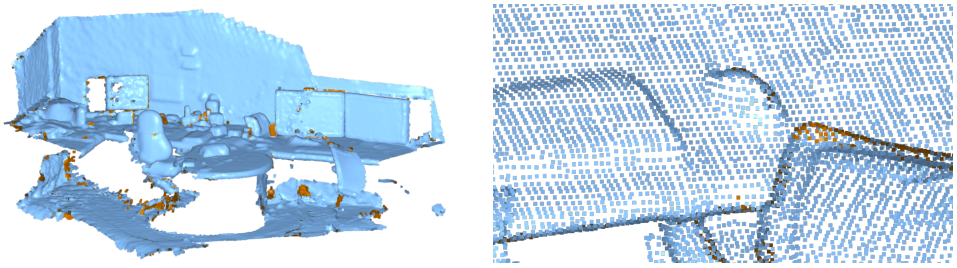
	#	min	max	avg
Scenes	64	-	-	-
RGB-D frames	130k	-	-	-
Point Clouds (PC)	2601	-	-	-
Points per PC	-	11k	640k	150k
BBox Volume per PC	-	0.35 m^3	134 m^3	20 m^3
Points Spacing	-	6.5 mm	8.2 mm	7.10 mm

Table 1.2: 3DMatch [6] main specifications. BBox stands for Bounding Box. The oriented BBoxes are estimated using Open3d [55] as well as the average point inter-distances that have been used to define the spacing.

In practice, we have picked 3DMatch dataset because it is one the most referenced dataset in the research community. Specifically, it is cross-referenced in most of the works we will end up evaluating in the following chapters. In

[1] `https://3dmatch.cs.princeton.edu/`

Tab. 1.2 we report the main specifications of the dataset. Such specifications have been estimated using Open3d [55], an open source library to deal with standard 3D data structures for multiple purposes. For each point cloud in 3DMatch we first counted the original number of points (no down-sample was performed), then we evaluated the bounding box that contains the whole data to estimate the volume of the scan. Finally, for each point we computed the nearest neighbor distance, we sorted all of them and we filtered out the 5% of the closest and furthest values. In particular, it is interesting to notice how on average each point cloud requires 150k points to represent a model with a volume as large as 20m^3. The 20 m^3 bounding box is coherent with the fact that each PC is obtained after fusing 50 consecutive frames, so it is reasonable that it scan a space as $3.5 \times 2 \times 3\text{m}$. Overall the average spatial resolution is around 7.5 mm. This is an acceptable value for such a large scale reconstruction, even considering the necessity of preserving meaningful geometry to address problems such as shape retrieval or object detection, which is accounted in the original work of 3DMatch. In Fig 1.12 a detail of the point cloud resolution is given for scene `office3` [19]. Although overall the common objects are recognizable in the global context, the quality of the reconstruction is low and not adequate to represent the geometry properly when they are considered individually.



(a) (b)

Figure 1.12: Example of the point spacing in a point cloud of 3DMatch [6]: (a) full scene `office3` (b) detail of the mouse and keyboard on the desk.

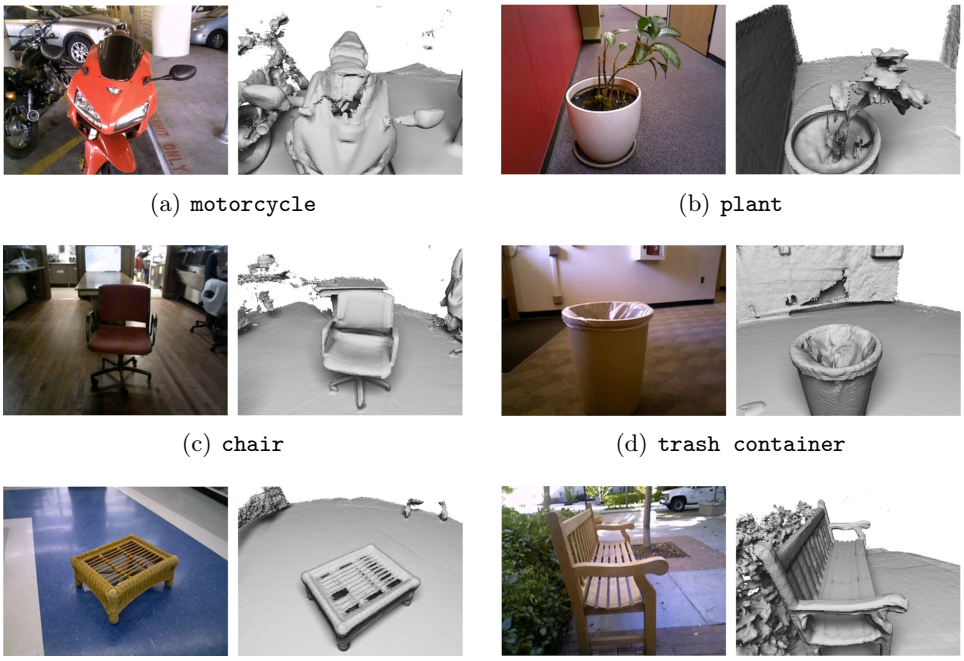
2.2 3D Object reconstruction: Redwood dataset

Figure 1.13: Example of object scenes collected in the Redwood dataset [57].

Although 3DMatch is a very useful dataset, it addresses a data type which is certainly different from what we target for our scanner. Instead of working with large scale indoor scenes, we are more interested into finding a dataset that targets object reconstruction. Choi *et al.* defines their Redwood dataset [57] as a "large collection of object scans". The dataset is large indeed: the authors recruited 70 operators to acquire 10933 scans, by using low cost PrimeSense Carmine optical scanner (see Tab. 1.1 for specifications). The scans were then categorized into 320 categories. Most of the scenes represent cars, furniture, sculptures and toys. Along with the RGB-D sequences, the authors offer also the 3D meshes representing the reconstructed 3D models, obtained via combination of KinectFusion [20] and dense visual slam [63] solutions. Actually, only a small fraction of the entire dataset is reconstructed. Indeed, most of the scenes are hard to deal with due to the lack of rigidity and the poor quality of the scan itself. Moreover, all of the meshes have been reconstructed with low space resolution and they still differ quite a lot from the kind of reconstruction quality we aim to perform with our scanner. In Fig. 1.14 we report some examples of the input video along with the final mesh for some of the most common scenes in the dataset. It is clear how also the camera field of view is the same of the ones used for building 3DMatch scene. Therefore, this type of dataset can be seen as particular case of 3DMatch where instead of the full indoor space, the focus is on a

smaller subset of the room but still with the same hardware and accuracy reconstruction level. In Chapter 2 we will see how the case study device differs from these scanners and how a novel dataset can be built by means of it.

However, we are still interested into leveraging some of the scans proposed by Redwood dataset. We dug into the dataset to find some useful scene so that we can exploit them along with our own data for future tests on the reconstruction pipeline that we will discuss later in the last Chapter 4. More comments on this will be given in that section.



(a) `motorcycle` (b) `plant`

(c) `chair` (d) `trash container`

(e) `table` (f) `bench`

Figure 1.14: Examples of input RGB frames - reconstructed mesh pairs for different categories in the Redwood dataset [57].

3 3D Reconstruction Methods

Many contributions have been given to this topic, which has been extensively covered over the last 30 years exploiting several methods, renovating technology and specific points of view to investigate the general problem. In essence, we want to build a 3D model by aligning a set of partial views. Our focus is on pairwise rigid alignment techniques (see Sec. 3.1 which can then support real-time multi-view reconstruction, starting from a collection of frames which resembles a video sequence in 3D domain, created via hand-

held 3D scanning. Such an approach represents only a small sub-part of a wider family of solutions aimed to address multi-view 3D reconstruction. Real-time applications favor a fast and redundant kind of acquisition at the expense of a low quality single view. Indeed, choosing for a higher quality of the single view affects the standard workflow during acquisition but the multi-view solutions can leverage such a higher quality. We refer the reader to some of our research group previous works on the latter subject [64, 65, 66], as well to other meaningful research papers covering the topic of multi-view and global registration. Moreover, we do not cover non-rigid reconstruction solutions either, because they are out of the scope of our research. Nevertheless, very interesting challenges and brilliant results can be addressed also in those situations [1, 67].

3.1 Rigid 3D registration techniques and workflow

Generally speaking, estimating a pairwise rigid 3D registration means finding the correspondences between the points in one set with respect to another and then to compute the rigid transformation $\boldsymbol{T} \in \mathbb{R}^{4\times4}$ that aligns the two point sets [68]. When no information is given beforehand regarding the relationships between the two sets, it is necessary to extract a set of features from both and then find the best matching point correspondences by exploiting the distances between the descriptors associated to the feature points. The type of transformation estimated by means of this investigation is usually called *coarse* because it relies on a sparse set of feature points only. After the coarse alignment is then possible to find the *refined* transformation based on a denser matching of the closest points in the two sets. In the following we review the workflow and the most popular methods to achieve a pairwise (i.e. between two 3D views) rigid registration by keeping in mind that the computational speed is almost as critical as the robustness and the accuracy when we want to fulfill the real-time requirement to run online reconstruction.

3.1.1 Keypoint detection

Feature-based methods rely on 3D keypoints to be extracted from a 3D surface to allow for effective description and matching and to be robust to the noise and point-of-view variation for achieving 3D reconstruction. For this reason, several 3D keypoints detectors have been proposed and discussed in surveys [69, 70].

The 3D detectors are generally divided in two categories: fixed-scale and scale-invariant.

Fixed-scale detectors They find distinctive keypoints at a constant preset scale by evaluating the distinctiveness of each point either in a point-

wise fashion or in a region-wise one. The points that maximize a quality function in a scale-dependent spatial neighborhood are then promoted as keypoints. The Local Surface Patches (LSP) [71] is an example of fixed scale point-wise detectors which uses the Shape Index [72] as the quality measurement for evaluation to represent the maximum or minimum principal curvature. A popular region-wise method is Intrinsic Shape Signatures (ISS) [73] which first computes the eigenvalue decomposition of the scatter matrix of the points in the support region, then uses the magnitude of the smallest eigenvalue and the ratio of two successive eigenvalues as distinctive quality measurement.

Scale-invariant detectors A quality measurement estimation at a fixed-scale is repeated for multiple scales for this type of detectors. The final measurement is then maximized both spatially at specific scale and across scales. Many of these methods works via 2D parametric representation of the lattice structure of the 3D mesh. For instance [74, 75] create a scale-space representation of the normal map of the mesh, then compute the eigenvalues of the Gram matrix of the support to estimate the cornerness of the point. Instead of cornerness, in [76] the quality measurement is represented by the mean and Gaussian curvatures which is then connected across similar regions. On the contrary MeshDoG [77] builds scale-spaces directly on the mesh by applying a Difference-of-Gaussians (DoG) filter to the mean curvature, the Gaussian curvature or the photometric appearance of a vertex in a region. Bonarrigo *et al.* [64] propose a 3D extension of SIFT [78]. Starting from the work proposed by Castellani *et al.* [79] and a revised proposal made by Lee *et al.* [80], the method applies multiple Gaussian kernels $G(r)$ where $r \in [1, M]$ is the scale used to produce one of the M filtered versions of the input range image. Then, the difference between consecutive filters $G(r)$ is evaluated to create $M - 1$ saliency maps that are then combined to detect interesting points at the original scale. As stated in [64], in order to improve the performances on range images and point clouds is necessary recompute the normals for all the points at each scale. Finally, 3D SURF [81] works on a voxelized version of the mesh to build the scale-spaces to extract the quality measurement, which is the Hessian of Gaussian second-order derivatives computed at each grid bin and each octave.

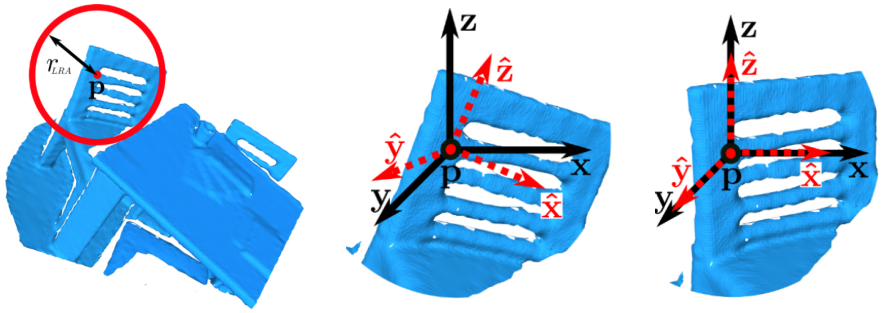
All these methods are useful to study the saliency of the points in a 3D model and to detect the most distinctive ones. However, for the sake of speed and lightweight computation, sometimes this step is omitted and a more rigid approach, like uniform sampling [82, 83], is adopted.

3.1.2 Feature description

Once located, feature points need to be qualified by some descriptors. In literature we can distinguish between two families of methods, namely *local*

and *global* descriptors.

3D global descriptors consider the entire geometry of an object, which is possibly segmented from a larger scene. Only one signature is then assigned to a cluster of points. In general these descriptors are employed in object recognition, model retrieval, pose estimation and shape analysis. Therefore, they are out of the scope of this review, which is more focused on the problem of local rigid registration. For the sake of completeness, we refer the reader to surveys that better cover this topic [84, 85, 86].

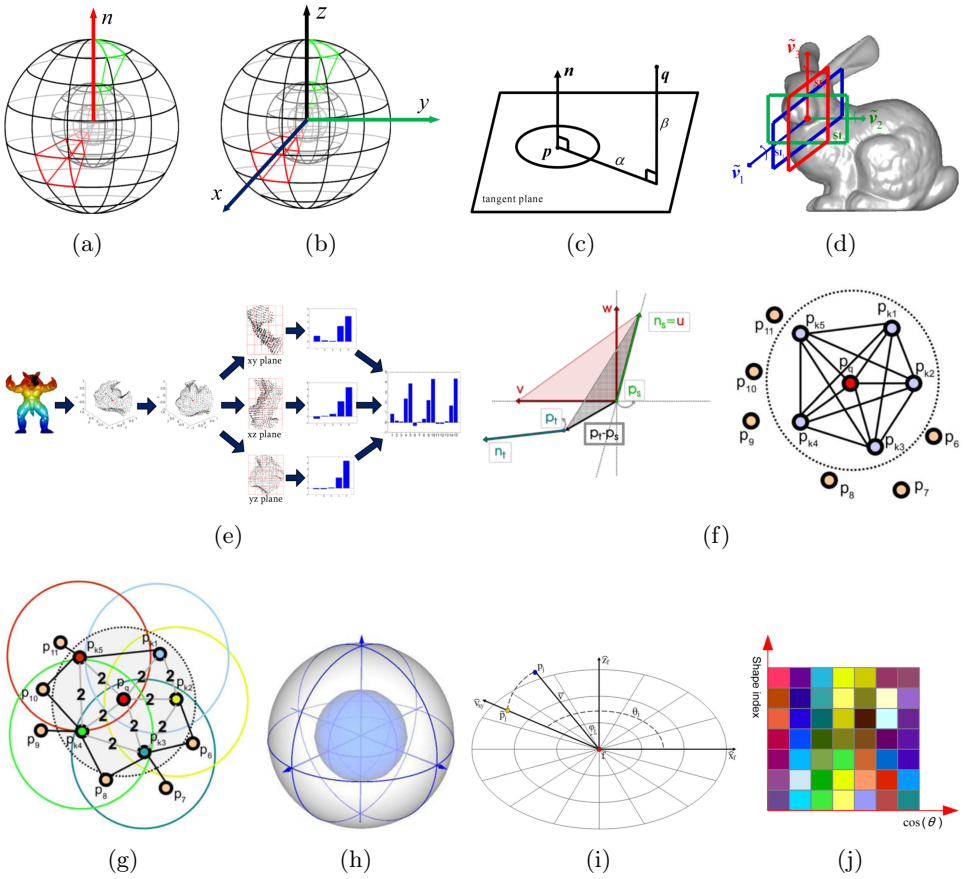


(a) (b) (c)

Figure 1.15: Local reference frame (LRF) estimation. (a) Spherical support is extracted from anchor point **p** (b) then the LRF is estimated and (c) the support is transformed to its canonical representation.

On the other side, local descriptors assign a distinct signature to each keypoint in the cluster under analysis [87, 88, 13]. Usually 3D geometry domain presents more challenges than its widely studied 2D color counterpart: indeed, additional challenges derive from data occlusion, clutter areas and not homogeneous point distribution [64]. Moreover, since they are local methods operating in the 3D domain, it is critical to provide robustness against affine transformations as rotation and scaling. On this regard, the developed approaches can either be based on intrinsically invariant features (similarly to what SIFT [78] does in 2D [64]) or they can exploit on the definition of a local reference frame (LRF) to transform each point into a canonical representation before evaluating its properties [89, 90] (Fig. 1.15). Moreover, highly expressive features like 3D SIFT [64], despite remaining only partially invariant to geometric transforms (they can be made equivalent up to a circular shift) can be pre-aligned (seeking their principal orientation) to obtain a a streamlined distance computation, similarly to what happens for LRF features [91].

Spatial relationship-based descriptors The support region around the anchor is then partitioned into several spatial bins, each of them containing a sub-part of the neighborhood points set. A simple measure as the amount of



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 1.16: Selection of 3D feature descriptors: (a) 3DSC [92], (b) USC [93], (c) Spin Image [94], (d) TriSI [95], (e) RoPS [96], (f) PFH [97], (g) FPFH [98], (h) SHOT [90], (i) 3D SIFT [64], (j) LSP [71].

points can be used to label each bin which is finally stored into an histogram. Indeed, most of the common methods for 3D feature description leverage on histogram structure to represent their signature. The main differences are related to the approach used to define the measured geometric properties or the LRF and the partition of the support region. Examples of this kind are:

- 3D Shape Context (3DSC) [92], which creates a 3D spherical grid around the anchor point **p**, aligned with respect to its normal **n**, then divide the grid into logarithmically spaced bins along radial dimension as shown in Fig. 1.16(a) and counts the weighted amount of points falling in each bin. A variant of this method is the Unique Shape Context (USC) [93] (Fig. 1.16(b)).

- Spin-Image (SI) [94] constructs the LRF similarly to 3DSC by using the normal **n** of the anchor point **p** as its main direction. Instead of

a spherical grid, a plane containing the point is defined perpendicular to this normal. Then the two legs of the triangle that is given by the anchor, by a sample point in the neighborhood and by its projection onto the plane (Fig. Fig. 1.16(c)) are used as 2D parameters stored into an array. The descriptor is generated by accumulating the points in the support region into each bin of the 2D array. Variants of SI are Tri-Spin-Image (TriSI) [95, 99] (Fig. 1.16(d)) and Multi-resolution Spin-Image (MrSI) [100].

- Rotational Projection Statistics (RoPS) [96, 101] is generated by concatenating the statistics of distribution matrices created after multiple rotations around the three axis and projections onto the three coordinates planes (one distribution for each plane), applied to all the points into the LRF (Fig. 1.16(e)).

Geometric relationships-based descriptors This kind of descriptors focuses instead on attributes like point normals distribution and curvatures. As before, the support region is partitioned into bins that are then used to compute a histogram for representing the final descriptor. Among these methods there are:

- Point Feature Histogram (PFH) [97] builds a multi-dimensional histogram investigating point pair features with pairs belonging to a Darboux frame (Fig. 1.16(f)). Each pair gives exactly 4 features (based on points distance and normals relationships) and final PFH is generated by accumulating points in the 4-dimensional histogram bins. Therefore the dimension of the feature is d^4 where d is the number of bins for each dimension. In order to speed up the description process, Fast Point Feature Histogram (FPFH) [98] calculates the relationships within a restricted neighborhood and stores them into three separated histograms that are then concatenated (Fig. 1.16(g)). This is a Simplified version of the original PFH (SPFH). FPFH also removes the first feature (*i.e.* the distance between points) to improve the robustness to non homogeneous point distribution situations. The final descriptor is then generated as the weighted sum of the SPFHs of the points in the support region and the SPFH of the anchor point. Its dimension is $3d$ where d is again the dimension of bins along each dimension.

- Signature of Histogram of Orientations (SHOT) [90] again starts from constructing an LRF, in this case by the eigenvalue decomposition of the input point p. It then segments the support region into several volumes along the 3 axes and generates a local histogram for each of them (Fig. 1.16(h)). The information stored in each bin is angle between the normal of p and the normals of the neighbors. This method is particularly robust to clutter.

- Extensions of 3D SIFT are proposed for descriptors as well. Bonarrigo *et al.* [64] propose to associate to each feature point extracted at scale r a descriptor which encodes information extrapolated from both the normal vectors and saliency data of the neighbor points confined in a polar grid which spans the tangent plane of the point and is composed of angular and radial sectors (Fig. 1.16(i)). Other extension are proposed in [102] with a 2.5D version of SIFT that works with depth images and in SI-SIFT [103] which uses SIFT to extract descriptors from the shape index values.

- Other methods are the aforementioned Local Surface Patch (LSP) [71, 104] (Fig. 1.16(j)), which evaluates the shape index [72] of each point and it is robust to occlusion; THRIFT [105, 106], which stores similar content to SHOT but in a 1D histogram, it usually performs worse than others but it is also robust to clutter; Persistent Point Feature Histogram [107], another variant of PFH. Finally, in order to address non-rigid deformations, GFrames [108] computes the intrinsic gradient of a scalar field within the LRF, while in [109] the wavelets are used for the localized space-frequency analysis.

An extensive comparison is made in [87] for all the methods presented in Fig. 1.16 except 3D SIFT. The results of such an evaluation show that the family of descriptors based on PFH have the best performances, both considering descriptiveness and robustness to keypoint localization. In particular, FPFH seems to be well suited for 3D modeling and it is by far the most compact descriptor. This means that the computational overhead which is required for matching is marginal with respect to the competitors. Moreover, it is the fastest descriptor to compute, up to 10k points while it tends to have scalability problems with larger sets. 3D SIFT, instead, is tested by Pingi *et al.*[110] and Petrelli *et al.* [111]. Both the works evaluate the computational cost, the robustness and determined the suitability of the method for the alignment of high-quality camera range images coming from metrologic static scanners. What emerges from these comparisons is that there is no absolute winner among the above kind of features for 3D data, and that the choice should be driven by the data characteristics and the application requirements. Overall, FPFH seems to be a good candidate for working in a object reconstruction framework with real-time constraints, due to the lighter computational cost and the higher compactness. We will cover more of this in the following chapters.

3.1.3 Feature matching

The goal of feature matching is to create a list of correspondences out of two or multiple sets of features. In practice, the euclidean distance between descriptors is the most common metric used for estimating the best match.

However, other metrics can be used, according to the type of feature signature, such as the Hamming distance. The problem can be generalized as:

$$x_j^i = \underset{x_j \in X_1}{\arg\min}(\|x_i - x_j\|_\ell)$$

where X_1 X_2 are two sets of features and $x_j \in X_1$ and $x_i \in X_2$, so for each feature in the first set we need to find the one closest to each feature in the second set. This is known as the Nearest-Neighbor (NN) Matching Problem on a high-dimensional data. In order to find the NN, a first trivial solution is to perform a linear search. However this is $O(n)$ problem which can be very expensive in terms of computational complexity when n is big. More efficient solutions are based on tree structures [112, 113] which are constructed by recursive binary splits that reduce the complexity to $O(\log(n))$ on average.

After we found the closest feature $x_j^i \in X_1$ for each $x_i \in X_2$, we still need to reject wrong matches. The first assumption we can made is that the matching should be unique. Therefore we can perform a *mutual* search and put in correspondence only those features that best match with each other. Another test we can perform is to consider *ratios* of relative distances (Fig. 1.17). In this case the rejection rule is to discard a match if the ratio of the distances between closest match x_j^i and second closest match x_k^i for x_i is greater than a chosen threshold. In a such a way, we try to remove ambiguous matches that are prone to produce false alignments. Overall, these tests

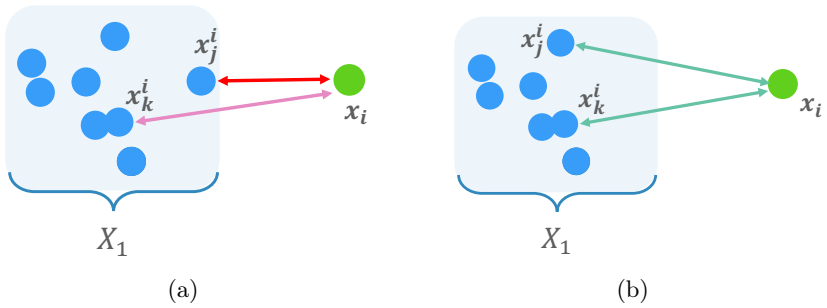


Figure 1.17: Ratio test for feature matching. (a) Clear closest match for x_j^i. (b) Matches are ambiguous. The correspondence is rejected.

does not guarantee the complete outliers removal. In order to get rid of remaining mismatches, a common practice is to use geometry constraints. In the following paragraph we see which solutions can be adopted to tackle the task.

3.1.4 Coarse transformation matrix estimation

In theory, if we assume that the outcome of the feature matching has no false positives, then the aligning matrix should be found straightforward.

However, in practice, the correspondence set is full of outliers that can affect the estimation. Robust techniques are then necessary to address the issue. This is analogous to a model fitting problem and the most popular technique in this field is RANdom SAmple Consenus algorithm (RANSAC) [114]. In practice, a small sub-set of the input data is selected randomly to estimate a coarse transformation matrix via Horn's method [115]. A threshold is given as input to the algorithm to define the minimum distance to consider two points as aligned. Once the putative matrix is applied to align the source set to the target, the number of overlapping points is evaluated. Then the process is repeated in iterative fashion starting from a different sub-set. After k_{max} times, the algorithm stops and the best transformation is chosen as the matrix that obtained the higher value of overlaps.

A visual example of a model fitting problem addressed with RANSAC is shown in Fig. 1.18. The first two triples of points have a bad fit, even if some points are actually inliers. The last triple instead produces the best fit. Critical parameters to chose for this method are the number of maximum iterations, the point sampling strategy and the inlier threshold, while a degeneracy check [116] is still usually required.

Through the years, many variants of RANSAC have been proposed in order to reduce the computational complexity [117, 118, 119, 120] or to improve the effectiveness [121]. Alternatively, other iterative methods investigate the use of branch-and-bound [122] or semi-definite programming [123]. Nevertheless, in practice is still difficult to ensure one of the above methods to be fully compliant to real-time constraints therefore they are commonly employed in offline applications only.

3.1.5 Transformation matrix refinement

Once a coarse registration is found or whether two sets of points are supposed to be close to each other (*e.g.* because they come from adjacent frames of a high frame-rate scanning as it happens with handheld scanners), it is possible to refine the alignment by considering the whole set instead of sparse correspondences only. Iterative Closest Point (ICP) [124, 125] is the *de facto* standard technique in this context. The name is self-explanatory: ICP aims to minimize an objective distance between the two sets by estimating a new correspondence set at each iteration. The most trivial objective is the point-to-point distance, however it has been demonstrated that a point-to-plane distance is preferable because it is faster to converge [125]. In Fig. 1.19 we report a simple example of how ICP can converge to a valid alignment when two sets are well posed initially.

Through the years, many variants of ICP have been proposed [126] and several papers addressed the convergence problem [125, 127] as well as its robustness to local minima [122, 128]. Fast Global Registration (FGR) [129] is another method which similarly to ICP uses a second-order optimization.

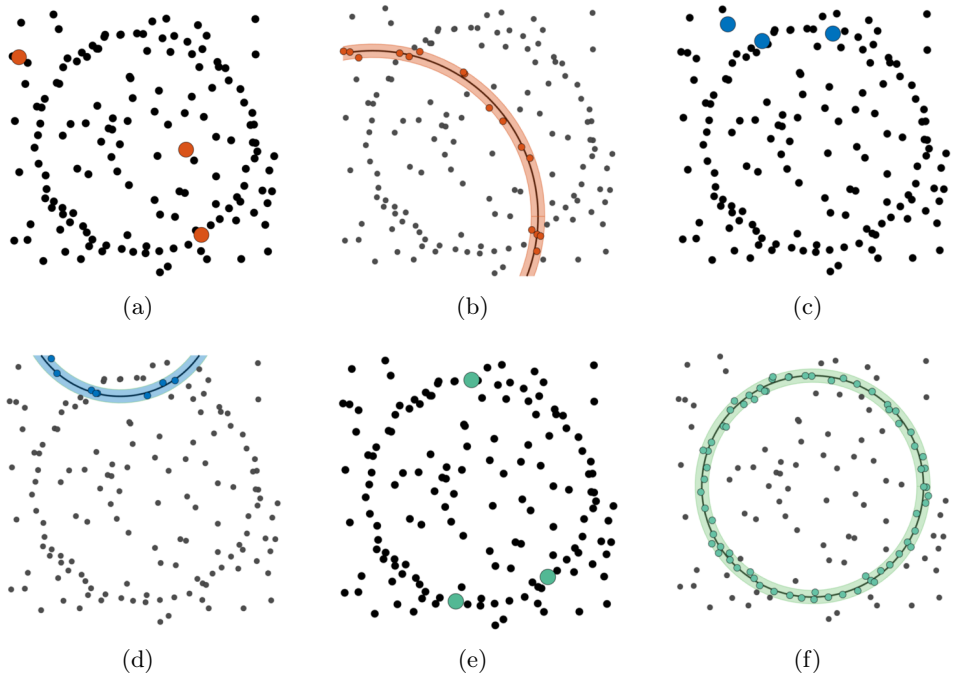


Figure 1.18: Example of model (a circle) fitting using RANSAC [114]. (a, c) Triplets of putative inliers are selected in the distribution. (b, d) The corresponding square is fitted using these triplets but bad results are produced. (e) The best triplet is selected and (f) the resulting best fit is shown in green. Image is courtesy of Luca Magri from the course of Geometric Computer Vision (`https://iecs.unitn.it/node/873`).

It is slower than ICP in good conditions but it is more robust to noise and local minima related problems. Finally, a less used but valid alternative is an iterative procedure to minimize correlation scores, proposed by Makadia *et al* [130].

3.2 Real-time 3D reconstruction pipelines

3D reconstruction [1] was for a long time just an offline process because of the computational complexity of the tasks asked to be solved. However, thanks to the increasing power of processors and by means of the computational gap gained with the advent of parallel programming on graphic cards, in the first decade of 2000s research about live 3D reconstruction systems consistently started. In Sec. 1.3 we presented the devices that lead the revolution of the 3D scanning on affordable systems. In particular, Kinect was the first solution to become mainstream. By leveraging on this consumer technology, academic research started focusing on the problem of online 3D

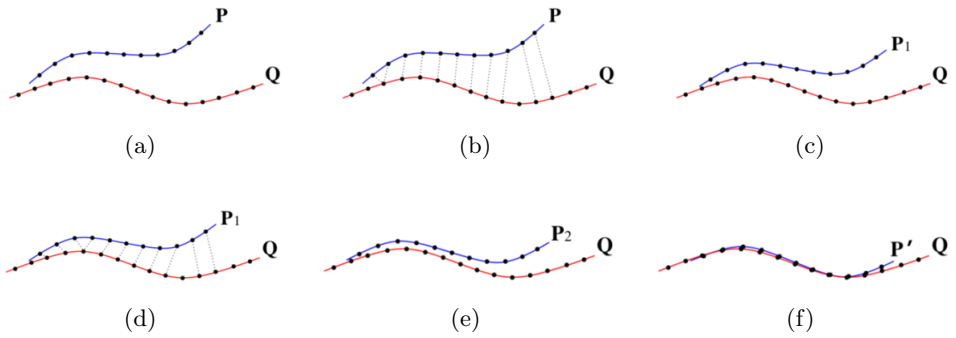


Figure 1.19: Iterative Closest Point (point-to-point) algorithm. (a) Initial situation. Set of points **P** must be aligned to **Q**. (b) For each point in **P** a corresponding point in **Q** is found. (c) First alignment matrix based on previous set of correspondences. **P** is transformed to $\mathbf{P}_1$. (d) Repeat search of closest point between $\mathbf{P}_1$ and **Q**. (e) New transformation applied to $\mathbf{P}_1$, which becomes $\mathbf{P}_2$. (f) Final result after ICP converged to minimal.

reconstruction. The power of the solution is that it allows the user to have a constant feedback of the growing 3D model while he scans. In order to achieve it, the systems needs to simultaneously update the model and localize the sensing device with respect to it. This is a well established problem which goes under the umbrella of Simultaneous Localization and Mapping (SLAM) applications [131, 132, 133, 134, 135, 136, 137]. Such a topic covers a large number of sub-fields, from autonomous driving, unmanned aerial vehicles, robotics, etc.

We start our review by deepening the first outstanding solution proposed based on Kinect data, namely KinectFusion (KF) [20, 53, 54] and then we move on with BundleFusion (BF) [19], which is current state-of-the-art for robust real-time reconstruction method when a handheld optical scanners is in play. We review the core methods and we seek to understand the strength and the weaknesses of both.

3.2.1 KinectFusion

KinectFusion (KF) [20, 53, 54] (Fig. 1.20) is addressed as one of the milestones for real-time 3D reconstruction. The key ideas in this seminal work are: 1) to leverage on a *volumetric integration* method [62] to average the contribution of all the incoming frames that are acquired at a high rate with stereo camera sensor; 2) to apply a *frame-to-model* registration to boost the robustness of the method and to reduce the drift problem that affects sequential registration-based frameworks. A scheme for the entire pipeline process is presented in Fig. 1.21.

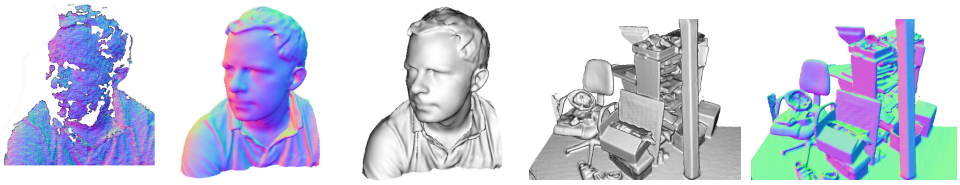


Figure 1.20: Example output from KinectFusion system. An incomplete, noisy data is acquired live with a Kinect (first image). The normal maps (color images) and the Phong-shaded renderings [138] (gray scale images) are obtained from the dense reconstruction system. Image taken from [20].

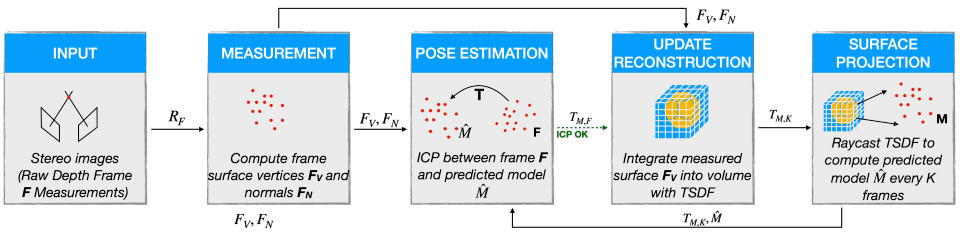


Figure 1.21: 3D reconstruction pipeline for KinectFusion [20]. The input is a pair of images acquired in stereo configuration. A set of points and normals is extracted from the RGB-D image which is created via stereo triangulation. The working frame is then aligned with the growing model using ICP [125]. Via TSDF integration [62] the aligned data is integrated into the volume representing the reconstructed scene, which is then updated. In order to define a model to align with and to provide a live feedback of the reconstruction, the volume is periodically ray cast.

A raw depth map is estimated from the stereo triangulation of two cameras, then a `Measurement` stage is in charge of estimating the vertices and the normals of the surface extracted from the input depth map, which defines the current frame **F**. The output model is obtained in the `Update Reconstruction` step by means of an implicit representation of the surface, via volumetric integration of the frames (as suggested by Curless and Levoy [62]). Then, in order to fuse each frame properly with respect to the volume, the camera coordinate system needs to be referred to the world one. In other words, it is necessary to keep track of the moving camera in the environment. To do that, a first trivial solution would be to align consecutive frames. However, such a solution suffers from the accumulation of the errors inherent in each pairwise registration, especially when source raw data is noisy and quite sparse. Therefore, KF suggests to use a *frame-to-model* approach instead. In the `Pose Estimation` step ICP is run to align the incoming frame to a reference one $\hat{\mathbf{M}}$ deriving from the ray casting of the growing volume. $\hat{\mathbf{M}}$ is also known as *key frame* or *anchor frame*. In order to stay as close as possible

to the current location of the camera, such a key-frame is then updated on a regular basis in `Surface Projection`. The authors of KF opted for updating the model every $K = 10$ frame according to an empirical evaluation. Finally, ray casting is also exploited to give a constant feedback to the user about the geometry of the scene under construction and the scanner position relative to it.

KinectFusion issues The method is affected by three main problems. The first is about **memory management**, which is inherently affected by the voxel grid data structure. Such a structure is inefficient since each voxel can either represent a surface point, or an empty space. In such a configuration, the complexity of the system that is required to access and to update a voxel value is linearly dependent on the number of voxels. Even when a smarter structure like octree [113] is in place, the complexity is reduced down to $O(\log(n))$ only. In their work [139], Niessner *et al.* propose to use voxel hashing to hash occupied voxels only, which is capable of reducing the complexity to $O(1)$, and it makes the pipeline robust to scaling with larger volumes. Indeed, all the methods successive to KF propose a certain hashing algorithm to deal with the problem of memory management.

The second problem is the **camera tracking loss**. As we mentioned in Sec. 3.1.5, ICP requires the two set of points to be close to converge to optimal solution. If 1) there is no sufficient overlapping surface or 2) the geometry is ambiguous (*e.g.* planar regions), then the algorithm will fail and so the camera will not be tracked correctly. In the original implementation of KF, no solution is given when the tracking is lost. Indeed, in these situations usually the user is simply warned of the failure and asked to move back, in order to get closer to the latest known position and try to recover the track. In worst cases, the user fails to re-align properly and must stop the acquisition. To address such problem, Glocker *et al.* propose an image-based camera relocalization solution [140] in which the low resolution images produced by the Kinect are encoded using randomized ferns [141] and binary feature tests. First a dissimilarity check is performed on each frame to choose whether a frame should be added to a key frame table or not. Then, whenever the track of a new frame fails, a block-wise hamming distance evaluation occurs to find the best alignment in the table and it retrieves the relative coarse camera pose. If the estimation is sufficiently close to the optimal, then the ICP should converge again to a valid results. Overall, the solution is promising and it produces good results whenever the encoding can rely on a meaningful 2D texture. However it still lacks of a robust solution for those cases with poor texture information.

The third problem is about **false positive alignments**. Indeed, ICP can converge to local minima in some challenging situations. The result of this issue is that the data is integrated into the volume with a wrong pose

and the final model is compromised. Although some specific heuristics can be developed to make the system self aware of the wrong evaluation, they are usually hard to generalize. A more robust solution should then be put in place to compensate for these errors, and it should be necessary to work on-the-fly during model creation.

3.2.2 BundleFusion

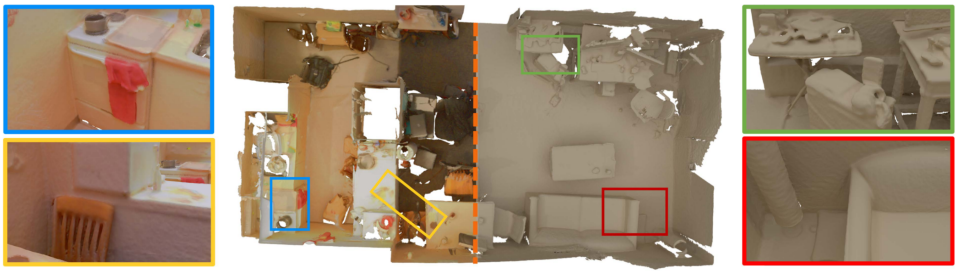


Figure 1.22: Example output from BundleFusion system [19]. Reconstruction of `copyroom`. Normal map and RGB texture in color images with details of the model, and Phong-shaded rendering [138] in gray scale.

The work presented by Dai *et al.*, called BundleFusion (BF) [19], addresses all the problems we discussed above with KinectFusion. Since its focus is primarily on 3D scanning of large-scale scene, a voxel hashing technique [139] is adopted for managing the volume. Then there are three main novelties in the work: 1) use of a **hierarchical data structure**, 2) definition a **coarse-to-fine** registration method based on sparse and dense information, 3) development of an **optimization pipeline** to adjust the initial guess of a frame pose and to refine the model on-the-fly. A scheme of the working pipeline is provided in Fig 1.23.

The hierarchical structure organizes the sequence of frames into buckets, or *chunks*. At local level, each chunk contains a fixed amount of frames ($N = 10$) where the first of the list is the anchor. Moreover, the full sequence can be seen at global level as a cluster of chunks. Then, in order to be registered, a new frame goes through a two-steps registration stage. First, a set of 2D features based on SIFT [78]) is extracted from the image. A fast research against the table containing all the previously extracted key frames features is performed in order to find the best putative match. All the process is powered by a GPU card and multiple researches are performed in parallel, allowing for real-time performances. The main limitation is about the size of the table, which is kept lightweight by heavily sub-sampling each input image and it works with a maximum of 150 features per frame. Moreover, a dense matching evaluation is additionally performed by assessing dense photometric and geometric constraints. Overall, the alignment problem is

addressed as a variational non-linear least squares minimization problem. The objective of the minimization can be seen as a weighted composition of sparse and dense contributions in the unknown parameters χ which is the collection of unknown camera poses:

$$E_{align}(\chi) = w_{sparse}E_{sparse}(\chi) + w_{dense}E_{dense}(\chi)$$

Such a minimization problem is then exploited using a parallel Gauss-Newton solver (GN), which is fast to converge and GPU compatible. Finally, the GN method can be used independently to solve a local alignment problem or a global one. The local alignment is an *intra-chunk* alignment: $\chi = \{f_i\}, \forall f_i \in C$ where C is a chunk and $\{f_i\}$ is the set of frames poses in it. On the contrary, the global alignment works at chunk level, *i.e.* it is an *inter-chunk* alignment: $\chi = \{k_i\}, \forall k_i \in S$ where S is the full sequence and $\{k_i\}$ is the set of key frames representing their chunks.

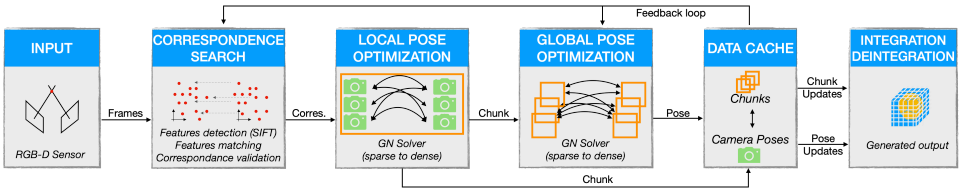


Figure 1.23: 3D reconstruction pipeline for BundleFusion [19]. A pair of images in stereo configuration is used to retrieve the RGB-D data frame of the current view. A first sparse correspondence search is made for the working frame with respect to the key frames already acquired. The pose is first optimized locally using a Gauss-Newton solver to minimize the sparse matches distances and exploiting dense photometric and geometric consistencies. The second optimization is global, and evaluates only the chunks containing a bunch of frame each. A data cache is used to store the key frames for a fast global correspondence search. The model is grown by volumetric integration and eventually refined via re-integration after pose optimization.

Since the frames are constantly re-evaluated in a multi-view alignment configuration, the original poses can be optimized during the scan. The final contribution of BF is to propose to revert the volumetric integration [62], which is actually possible because of the symmetry property of the TSDF. Therefore a *re-integration* stage is set: when the optimized pose p_{opt} of a frame differs from its current pose p_{curr} by a significant margin, the frame is removed and re-integrated back with p_{opt}.

BundleFusion issues The method is significantly more robust than KinectFusion but it is also inherently more complex and it requires a lot of computational power to run. Specifically in the original paper the system is powered

by two high-end graphic cards simultaneously. Moreover, the sparse registration method relies on 2D matches using SIFT descriptor. Such a descriptor is very solid and invariant to scale. However, using a 2D descriptor can be tricky in cases where the framed scene is homogeneous and the surface has a repetitive or featureless pattern. This is not the case for the typical type of input data in BundleFusion, where a large field of view camera acquires scenes with several distinctive regions in it. For instance, in the reconstruction process of a room a single frame can contain partial view of the floor, a chair, a table, objects on the table, etc. On the contrary, it can become an issue when the target is a partial view of a single object, so that the frame represents a partially smooth surface with poor or no additional texture information. In this case, the first step of the registration pipeline could be deceived several times, creating a bad model in the end or failing the reconstruction at all.

4 Conclusion and Motivations

Currently, a large selection of 3D scanners is available on the market, sorted by different geometric and metric characteristics, as well as price ranges, so that we can address a multitude of tasks in various application contexts. So far, Computer Vision and Computer Graphics communities have focused their effort on several problems in 3D domain by leveraging on affordable devices to produce a considerable amount of work and favoring a technological progress in different areas. For instance, regarding 3D reconstruction and 3D registration, the most common choice is to rely on low cost handheld passive stereo systems which in general provide low resolution models. These devices were originally introduced as appealing solutions for gaming and gesture motion tracking and they are suitable for non accurate reconstruction in indoor and outdoor environments. However, other applications, such as reverse engineering, industry quality control, cultural heritage and single object modeling require a higher level of accuracy which is usually achieved by means of photogrammetric systems. Nevertheless, nowadays the continuous technological progress offers the possibility to investigate higher-end handheld solutions to tackle the task of 3D reconstruction seeking for quality results closer to photogrammetric counterpart, but offering also the advantages of a real-time operating pipeline. Although the trend is promising for this kind of devices, which indeed are taking their spot in the consumer market, the academic research still lacks of a proper deepening and a reference dataset to assess the algorithms of interest.

In this chapter we saw how the state-of-the-art proposes a rich set of solutions to deal with 3D reconstruction. In particular, feature-based registration systems offer a robust approach to deal with some of the most common issues affecting classical scanning workflow, namely loss of the track, camera

relocation and bad alignments detection and correction. Even if the research comprises many valid examples, we can not consider it full-stacked. Indeed, most of the choices for feature design and pipeline development answer to specific problems and requirements and explore several trade-offs to achieve their goal (as e.g. in [111]).

Moreover, 3D domain gained interest in deep learning research community recently. Some modern data-driven solutions have been proposed to address specific tasks of the reconstruction chain, such as the feature extraction [14, 15] and robust correspondence matching to regress 3D registration matrix for pairwise [16, 17, 142] and multiple-view alignment [18]. Such a novel branch of study is fascinating and promising and it seeks to revise the way we deal with classical problems related to 3D reconstruction. In addition, it opens to new exciting challenges specifically related to the case of real-time robust reconstruction, exploiting high quality data derived from an handheld optical scanner.

In order to better understand the context, in Chapter 2 we review the application scenario and we introduce all the technical details defining the instrument under study. Finally, we make up for the lack of reference data for further evaluations by introducing a novel dataset we created *ad hoc* with the aforementioned device.

Then, in Chapter 3 we review the world of 3D deep learning by focusing on the latest techniques that have been proposed to address the challenging 3D registration of point clouds and we compare them with a cross-domain assessment in which we consider both classical data and our own dataset form benchmark purpose.

In the final Chapter 4 we present a novel pipeline which processes high resolution data by leveraging on a combination of handcrafted and data-driven solutions in order to achieve a high quality, robust and real-time compliant 3D reconstruction of single object models.

Chapter 2

The InSight scanner and the DenseMatch dataset

The InSight is a real-time handheld optical scanner which can be classified as an active stereo-based scanner. It is a fully working pre-market hardware prototype with its own dedicated acquisition software which has been developed by OpenTechnologies - FARO (Rezzato, Brescia, Italy) the company where I worked during the PhD in apprenticeship period. Although not currently sold by the company it is well representative of a family of semi-professional and professional products on the market with price ranging from few keuros to few tens of keuros. The active component of the InSight scanner is the LED projector which emits infra red light to add texture to the object for ensuring depth sensing robustness enhancement. In order to review the features of the scanner, we now take a look at its specifications sheet. However, before diving into the technical description of the device, we would like to focus on its appearance. A render of the device is given in Fig. 2.1. Still being a prototype, the form factor is clearly less compact than what competitors sell on the market nowadays. Moreover, being a project started in 2014, some of the hardware solutions adopted here could sound a bit out of sync with respect to the current alternatives. Nevertheless, we actually do not stress the hardware perspective in this work, since our focus is more inclined to an algorithmic point of view. In practice, the most important aspect in the technical sheet is the optics configuration, because of its distinctiveness with respect to the competitors.

That being said, we finally review how the scanner works. We will start from the hardware specifications and then we will move onto dissecting how algorithms perform in order to reconstruct a 3D model of the real world.

When I first started the PhD, my first goal was to become familiar with the scanner, to study the properties, the qualities and the issues it came with. Since the beginning, the goal was to improve the usability of the product. In order to do this, I focused on the flaws of the pipeline, I looked

Figure 2.1: Render of the InSight scanner. Red dot: IR projector. Blue dots: IR cameras in stereo configuration. Green dot: color camera.

for the most critical aspects and I tried to solve them by exploiting the most recent solutions I learned through my studies and my research. In Sec. 2 we review the first attempts I made in my first year. However, the improvements were not conclusive. Because of this, we then started exploring a new path of research and we then focused on modern data-driven applications to address the problems related to robustness and computational efficiency. Such an investigation will be the core and the objective of the discussion in the reminder of the work. The last section of this chapter is then preparatory for assessing the recent solutions based on neural networks we found promising. Indeed, in Sec. 3 we introduce a novel dataset we created by means of the InSight, DenseMatch [J2]. We will refer to this dataset for testing and benchmark purposes moving on with the research.

1 How the InSight scanner works

1.1 Hardware specifications

The key components in the scanner are the optics, the infra red pattern projector and a custom electronic board. We review them separately in the following paragraphs.

1.1.1 Optics

InSight has three cameras: two are used in stereo for depth sensing while the third is used for color information. The camera model is the same for all, specifically a 1.3Mpx Basler Ace[1] that runs up to 60fps (Fig. 2.2). The

[1] https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-60gmnir/

C-mount camera focal length is equal to 16mm and the time exposure is fixed at 20ms per frame. A 850nm centered band-pass filter is then mounted on top of both the depth sensors to pair with the wavelength emitted by the projector.

According to the technical sheet, the field of view of the scanner has a nominal value equal to $150 < x < 500$mm. However, in practice, the scanner works using a frame size close to 200×250mm and a working distance ranging from a minimum distance of 500mm up to 750mm. Overall, the working field is narrow, which is meant to achieve high accuracy on closeups. We point out that the specifications of the field of view and the high resolution of the cameras are critical to understand how the final image we create with the InSight differs from the standard solutions available with low-cost scanners as the Kinect or the Structure, which instead have a wider frame and lower spatial and depth resolutions.

Figure 2.2: Basler Ace camera mounted on InSight for depth sensing.

1.1.2 Projector

The projector is a OSTAR LED emitter made by OSRAM[2] (Fig. 2.3). It has a nominal optical output power of 3.5 and a wavelength working range from 850nm to 940nm, which belongs to the near infra red region. Collimating lens is used to direct the radiation. Moreover, in front of the projector is placed a glass which has a pattern printed via reflective coating. Such a pattern has a pseudo-random distribution of points which is used to add a synthetic texture onto the target. The rationale is to help building the depth map by disambiguation of repetitive structures using a noisy and unique pixel distribution (Fig. 2.4). Since the pattern is pseudo-random and no exploitation of the point distribution is made, InSight can be considered an *unstructured-light* optical scanner. Such an approach is proved to be a valid alternative to standard structured light solutions since it can be effective against interreflection issues [143].

[2] `https://www.osram.com/os/applications/ir-illumination/surveillance.jsp`

Figure 2.3: Infra red emitter made by OSRAM.

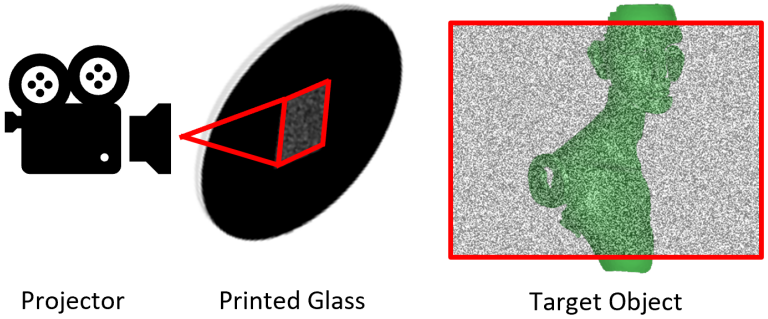


Figure 2.4: Pseudo random pattern projection for unstructured light scanning.

1.1.3 Electronic board

The custom electronic board is used for several tasks: projector power control, cameras-projector synchronization, temperature monitoring and fan control (which are in place for cooling the system). In Fig. 2.5 we show a picture of the board. It is important to highlight here that the computation is made entirely off-board, while on board we have two connectors for interfacing with an host PC. In particular, a Gigabit Ethernet switch is used to connect all the cameras to the host by means of a single cable, while an USB2 connector is used for the control board instead. For the sake of completeness, we report that more recent alternatives can rely only on one USB3 protocol to drive all the information together. However, Ethernet is known for being an extremely robust protocol which also allows for longer cables, therefore it is a valid option still nowadays.

Finally, on the back of the enclosure we can find two physical buttons (used for starting and stopping the scan) and an RGB LED that provides scan status information (blue light indicates that the scanner is in pause, green light is on during the scan and red light warns the user that the tracking is momentary lost).

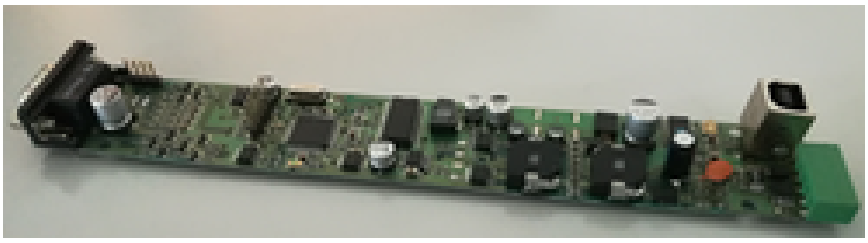


Figure 2.5: Electronic control board mounted onto the InSight.

1.2 3D Scanning Pipeline and Algorithms

The workflow the InSight uses for scanning is very close to the one presented in KinectFusion project [20, 53, 54] we already discussed in Sec. 3.2.1. In Fig. 2.6 we show a schematic representation of the main stages that compose the pipeline. In the following paragraph we explain how to obtain a 3D model with our stereo scanner. However, for readability sake we do not address again the elements of stereo vision in this section. Indeed, the reader can refer to the Appendices Sec. A for further details on the matter.

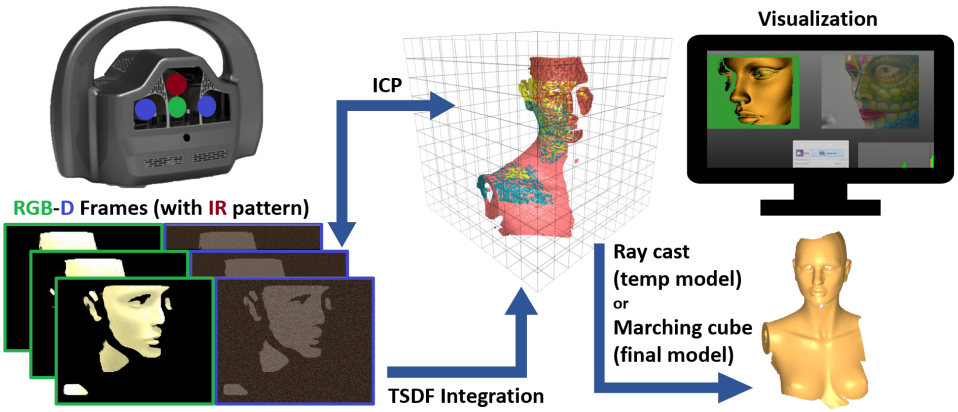
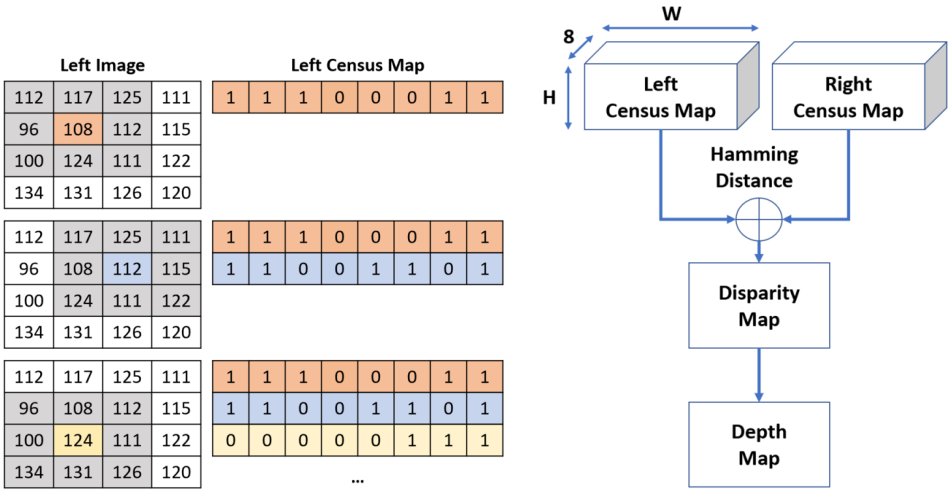


Figure 2.6: InSight reconstruction pipeline main components.

Each iteration of the pipeline starts from the acquisition of an image from both the two depth sensors. A disparity map is created by means of a multi-scale sparse Census transform [144] (Fig. 2.7). At each scale, a Census string is associated to every pixel. The Hamming distance between the strings is the chosen matching cost used to find putative matches between the images. The disparity computed at the higher level of the pyramid is then propagated to the lower levels (we use 3 levels in total). Finally, a parabolic interpolation refinement is applied to find the sub-pixel 2D matching coordinates in the images. Once we have the disparity map, we can finally create the depth map. We store all the information into a custom data structure, called `range image` (RI), which has the same domain of the input image - the pixels - but

a different co-domain. In fact each pixel is assigned with 3D coordinates and auxiliary information such as color, normal and confidence.



(a) Census creation (b) From census to depth map

Figure 2.7: Census transformation for depth map estimation. (a) Census creation: each pixel in the stereo images is compared against a fixed kernel neighborhood. For every pixel a binary signature is defined, in which a bit represent whether the neighbor has lower (0) or higher (1) value than the anchor pixel. (b) The two census Map are then evaluated using a Hamming distance to estimate a Disparity Map. Using basic stereo triangulation mathematics the final Depth Map is derived from the Disparity Map.

Similar to KinectFusion, we adopt a frame-to-model solution to track the camera. Every 10 frames we update our model by ray casting the volume we are building (we adopt a voxel hashing policy as in [139] to efficiently manage the data) and we exploit ICP point-to-plane [124, 125] to align our working RI to the anchor RI. To speed up the alignment process, we apply a uniform sub-sampling of the working RI by selecting only one column out of 10 in the image. If the alignment is successful, we then integrate our RI into the volume by means of a TSDF [62]. Once the scan is complete, we can instantly recover the 3D model in the form of mesh by using a marching cubes algorithm [145] onto the volume.

All the algorithms are implemented in C++ and CUDA [146, 147] to leverage on the GP-GPU capabilities provided by Nvidia graphic cards. The system can run up to 15 fps using low budget configuration with GeForce 1050, Intel i5 7400 processor and 16 Gb of RAM.

2 Direct pipeline adjustments to improve 3D registration

In Sec. 3.2.1 we discussed what issues affect the method of KinectFusion. In particular, we mentioned the frequent camera tracking loss and the lack of a robust solution for the relocalization. Indeed such issues affect also the InSight reconstruction pipeline, which adopts the same frame-to-model solution for tracking. Therefore we want to try to improve the robustness of the system by rethinking some of the classical approaches aimed for robust a tracking. In particular, we start by addressing two critical aspects in the KinectFusion pipeline, namely the decision policy for updating the key frame which is used as an anchor during the tracking, and the input sampling strategy adopted to prepare the alignment between the current frame and such anchor. The essence and results of these investigations was published in [C1]. The second attempt instead, replaces entirely the current pipeline with a more sophisticated one, *i.e.* BundleFusion, which we already introduced in the previous chapter in Sec. 3.2.2. We test this approach on the InSight generated data using the open source version of BundleFusion and we discuss the emerging issues, addressing the fact that some of the key strategies adopted in BundleFusion are not well suited for the typical application domain of the InSight scanner and similar ones.

2.1 Replacing key frame selection method with an adaptive approach

During the scan process, we must update the key frame many times in order to follow the camera movement. Until the sensed area matches with the reference model the two range images have a sufficiently large overlapping region and so the ICP can successfully converge. Therefore it is crucial to determine the optimal moment to update the model.

The first trivial solution is to choose a uniform pattern to update the key frame every N iterations. In this case, when $N = 1$ we move back to the degenerate solution of a frame-to-frame model, which we already discussed to be non ideal because of the large drift it inherently produces. In practice, KinectFusion [20] suggests $N = 10$ as a valid trade-off for a scanner that runs at 20fps on average.

We give a schematic representation of the approach in Fig. 2.8. The solution is trivial and effortless, however it is sub optimal and actually prone to fail quite frequently. Indeed, the lack of robustness for such a *static* method is due to the absence of any kind of evaluation regarding how fast the user actually moves the camera (i.e. the handheld scanner). Indeed, choosing $N = 10$ is a good value *on average* but it does not considers neither situations when the scanner stands almost still nor, on the opposite, when the user moves the scanner faster than appropriate. In the former case, the

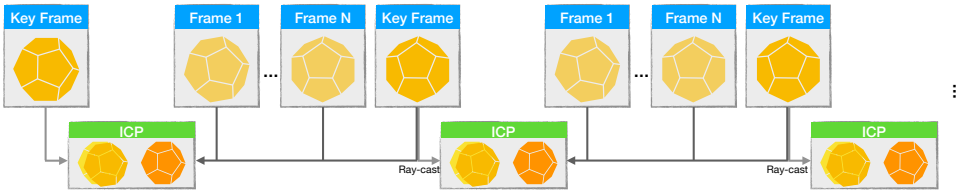


Figure 2.8: Uniform key frame update policy which updates the model every N frames. We suggest to look at the picture in the electronic version.

update should be unnecessary, while in the second case the scanner could rapidly end up framing an area with no sufficient overlap with respect to the model in use. Finally, in some cases it can happens that the update occurs at critical moment such that the ray casting traverses only a small subset of the occupied voxels in the volume. Since no dynamic policy is in place, the method is agnostic to the frame-model relationship and no update will be considered for the next N iterations. Unfortunately, it is highly probable that the track will be lost in the meantime.

In order to be responsive, we need to define a parameter to describe the difference between the range image and its key frame. A common strategy is to first estimate the difference between the key frame pose and current camera pose and then update the model whenever such delta exceeds a fixed threshold [148]. Another solution is to set a threshold for the maximum deviation of the alignment error [149]. Although such approaches are more robust than the uniform update, they still relies upon a threshold selected *a priori* which can not be optimal with respect to runtime applications.

Promising results are then presented by Kerl *et al.* in their work related to dense visual odometry (DVO) [63, 150]. We report in the following the main steps that lead to the adopted solution. However, in order to avoid an overhead of formalization we intentionally omit some lateral steps throughout the computation. We then invite the reader to refer to [150] for the complete overview of the method.

DVO aims to estimate the camera motion g^* between two range images, parametrized by a six-dimensional vector $\boldsymbol{\xi} \in \mathbb{R}^6$ that contains the three components of the linear velocity of the motion (v_1, v_2, v_3), and the three components of its angular velocity $(\omega_1, \omega_2, \omega_3)$. The estimate is done by minimizing the photometric and the geometric residuals, respectively r_I and r_Z, defined as:

$$r_I = I_2\big(\tau(\mathbf{x}\mathbf{T})\big) - I_1(\mathbf{x})$$
$$r_Z = Z_2\big(\tau(\mathbf{x}\mathbf{T})\big) - \big[\mathbf{T}\pi^{-1}\big(\mathbf{x}, Z_1(\mathbf{x})\big)\big]_Z$$

Here $I_i(\mathbf{x})$ and $Z_i(\mathbf{x})$ represent the intensity and the depth values at pixel $\mathbf{x}$ for a generic frame i. $\mathbf{T}$ is the rigid body transformation, π^{-1} represents the inverse of the projection function $\pi(\mathbf{p})$ which projects a 3D point $\mathbf{p}$ from

homogeneous coordinates space onto the pixel $\mathbf{x}$, τ is the *warping function* that maps the location of a pixel from one image to the other and $[\cdot]_Z$ returns the Z component of a point. Then, by assuming both the geometric and the photometric consistencies, the residuals differ from zero with a distribution that follows the probabilistic sensor model $p(\boldsymbol{r}_{I,Z}|\boldsymbol{\xi})$. Moreover the noise is assumed to be independent and identically distributed (*i.i.d.*) for all n pixels. Under this not so mild assumptions, we can leverage the Bayes' rule to determine the camera motion $\boldsymbol{\xi}^*$ by maximizing such a probability given the pixel-wise error $p(\boldsymbol{\xi}|\boldsymbol{r}_{I,Z})$:

$$\boldsymbol{\xi}^* = \arg\max_{\boldsymbol{\xi}} \frac{p(\boldsymbol{r}_{I,Z}|\boldsymbol{\xi})p(\boldsymbol{\xi})}{p(\boldsymbol{r}_{I,Z})} \tag{2.1}$$

By assuming the noise to be *i.i.d.*, dropping the $\boldsymbol{\xi}$-non dependent elements and converting the maximization to a minimization of the negative log-likelihood, Eq. 2.1 can then be written as:

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \Big(- \sum_i^n \log\big(p(\boldsymbol{r}_{(I,Z)_i}|\boldsymbol{\xi})\big) - \log\big(p(\boldsymbol{\xi})\big) \Big) \tag{2.2}$$

From such equation, by dropping the motion prior $\log\big(p(\boldsymbol{\xi})\big)$ and setting the derivative of the log likelihood to zero we end up with the equation:

$$\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{\xi}} \boldsymbol{w}(\boldsymbol{r}_i)\boldsymbol{r}_i = 0$$

where $\boldsymbol{r}_i$ is an easier to read notation for $\boldsymbol{r}_{(I,Z)_i}$ and $\boldsymbol{w}(\boldsymbol{r}_i)$ ($\boldsymbol{w}_i$ in the following) is the *weight function* defined as $\boldsymbol{w}(\boldsymbol{r}_i) = \partial \log(p(\boldsymbol{r}_i)/\partial \boldsymbol{r}_i \cdot 1/\boldsymbol{r}_i$.

In particular, according to Kenneth *et al.*, such a distribution is better approximated by a t-distribution [151]. Specifically a bivariate t-distribution, since the random variable $\boldsymbol{r}_{I,Z}$ itself is bivariate. A bivariate t-distribution with 0 mean, scale matrix $\boldsymbol{\Sigma}$ and ν degrees of freedom can then be represented as *i.e.* $p_t(\mathbf{0}, \boldsymbol{\Sigma}, \nu)$. In this context, the Eq. 2.2 now can be re-written as:

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \Big(- \sum_i^n \boldsymbol{w}_i \boldsymbol{r}_i^T \boldsymbol{\Sigma}^{-1} \boldsymbol{r}_i \Big)$$

which is a non-linear problem in the motion parameter $\boldsymbol{\xi}$. Kerl *et al.* [63, 150] use a first order Taylor expansion for the linearization around the current estimate, which turns the non-linear least square problem into a normal equation $\mathbf{A}\Delta\boldsymbol{\xi} = \mathbf{b}$ in which:

$$\mathbf{A} = \sum_i^n \boldsymbol{w}_i \mathbf{J}_i^T \boldsymbol{\Sigma}^{-1} \mathbf{J}_i \tag{2.3}$$

$$\mathbf{b} = -\sum_i^n \boldsymbol{w}_i \mathbf{J}_i^T \boldsymbol{\Sigma}^{-1} \mathbf{r}_i \tag{2.4}$$

Here $\mathbf{J}$ is a 2×6 Jacobian matrix containing the derivatives of $\mathbf{r}_i$ with respect to $\boldsymbol{\xi}$. The normal equations are iteratively solved for increments $\Delta\boldsymbol{\xi}$. Moreover, $\mathbf{A}$ can be seen as an Hessian matrix, or, equivalently, as the negative of the *observed Fisher information matrix*.

The final assumption is that the estimated parameters $\boldsymbol{\xi}$ are normally distributed as $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\xi}^*, \boldsymbol{\Sigma}_{\boldsymbol{\xi}})$ ($\boldsymbol{\xi}^*$ is the mean, while $\boldsymbol{\Sigma}_{\boldsymbol{\xi}}$ is the covariance). According to how the Fisher information matrix $\boldsymbol{A}$ is defined in Eq. 2.3, we have the lower bound to the variance of $\boldsymbol{\xi}$, *i.e.* $\boldsymbol{\Sigma}_{\boldsymbol{\xi}} = \mathbf{A}^{-1}$. Moreover, the differential entropy of a generic multivariate normal distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with n dimensions is defined as $H(\mathbf{x}) = 0.5 \cdot \big(n\big(1 + \ln(2\pi)\big) + \ln(|\Sigma|)\big)$. Dropping the constant terms, it results that the entropy is proportional to the natural logarithm of the determinant of the covariance matrix, *i.e.* $H(\mathbf{x}) \propto \ln(|\boldsymbol{\Sigma}|)$, so the information about the uncertainty encoded in the covariance matrix is converted into a scalar value. In practice, we first compute the Hessian matrix $\mathbf{A}$ by solving the normal equations with an iterative increments of $\Delta\boldsymbol{\xi}$ using a standard expectation maximization algorithm for the t-distribution, as suggested in [152]. Once we have the Hessian matrix, we can then compute $H(\boldsymbol{\xi})$.

Finally, the goal is to understand how the accuracy of the pose estimation degrades during the scan. Therefore we can leverage on the differential entropy $H(\boldsymbol{\xi})$ to have a hint of this quality measure. Indeed, the determinant of the covariance matrix will decrease for less accurate pose estimations and so the natural logarithm will tend towards $-\infty$. We can then define the entropy ratio α as:

$$\alpha = \frac{H(\boldsymbol{\xi}_{k,k+j})}{H(\boldsymbol{\xi}_{k,k+1})} \tag{2.5}$$

The numerator of Eq. 2.5 is the differential entropy of motion $\boldsymbol{\xi}_{k,k+j}$ occurred between the key frame and the current frame j. At denominator instead we have the differential entropy of the estimate of motion $\boldsymbol{\xi}_{k,k+1}$ occurred between key frame k and the successive frame $k+1$. Since the distance between the two frames is the smallest possible, the estimate is certainly the accurate, therefore it is set as the reference value. In principle, by worsening the accuracy of the pose estimation, $H(\boldsymbol{\xi}_{k,k+j})$ increases as well as the ratio α. It is now possible to set a threshold for this metric which is relative to the quality of ongoing camera pose estimation.

In Fig. 2.9 we report the new workflow which is now adaptive and offers a dynamic policy for updating the key frame in the InSight frame-to-model registration stage.

2.1.1 Experimental results

We now want to evaluate how the replacing adaptive method performs against the original uniform selection-based solution. In order to have a fair

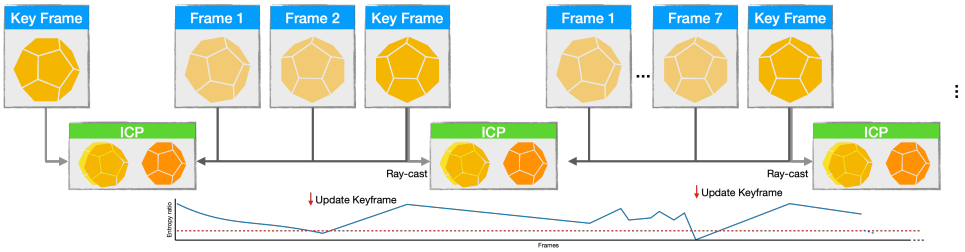
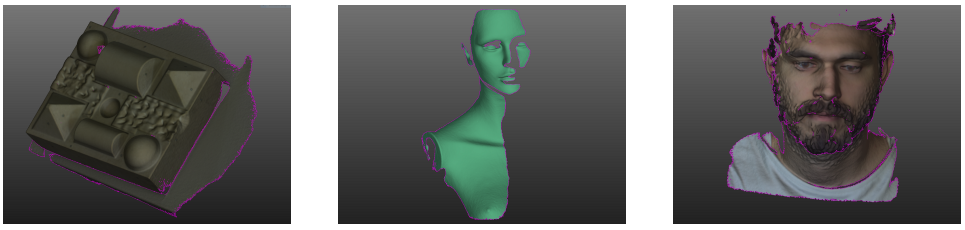


Figure 2.9: Dynamic key frame update policy based on entropy ratio evaluation [63, 150]. We suggest to look at the picture in the electronic version.

comparison we need to work using a common benchmark dataset. Therefore we use the InSight to acquire 6 scenes we want to test. These scenes are taken from real application scenarios where the InSight could be helpful. Specifically we scan a piece of marble with solids on top (a technical object for quality inspection), a plastic dummy (for object modeling), a couple of faces (for body scanning) and two hands (for orthotics). We report the 3D model of each scene in Fig. 2.10



(a) 85 frames (b) 161 frames (c) 239 frames

(d) 208 frames (e) 102 frames (f) 145 frames

Figure 2.10: InSight scenes acquired to test adaptive key frame solution. (a) `marble` (b) `dummy` (c) `face 1` (d) `face 2` (e) `hand 1` (f) `hand 2`.

In order to run the methods on the same benchmark dataset we need to test them offline. Therefore we have to save the frames sequences of each scene: such an operation comes with a sensible overhead to our end because it needs to move the data from the GPU to the host PC and then save it on disk. For this reason the final frame rate is halved to 7 fps. Moreover, due to

the reduced frame rate we need to account for a wider motion of the camera between two consecutive frames. Therefore we decide to set to $N = 5$ the number of frames we align to the same KF in the uniform sampling. Such a parameter is chosen after evaluating a couple of reasonable values for N starting by the initial consideration that $N = 10$ is effective when the frame rate is doubled. The same evaluation is made for the entropy ratio that ends up being set to $\alpha = 1.1$, meaning that the current differential entropy can not be higher than $\sim 10\%$ of the reference one. We use three metrics to evaluate the methods performance:

- The total number of successfully integrated frames, which tells us which method is more robust and reduces or eliminates the critical situations where the ICP is prone to fail.
- The total number of key frame updates required.
- The average root mean square error during ICP registration (considering the successful alignment only). In principle, the optimal key frame should have the lowest registration error possible. We then investigate which method sets the best condition for the ICP alignment.

Model name	# Frames Acquired	# Frames Integrated		# KF Updates		ICP RMSE [mm]	
		Constant	Adaptive	Constant	Adaptive	Constant	Adaptive
`Marble`	85	81	**85**	16	16	0.67	**0.37**
`Dummy`	161	134	**145**	26	9	1.05	**0.88**
`Face 1`	239	204	204	40	10	0.98	**0.72**
`Face 2`	208	119	119	23	7	**0.27**	0.29
`Hand 1`	102	93	93	18	9	0.66	**0.38**
`Hand 2`	145	145	145	28	6	0.21	0.21

Table 2.1: Testing two key frame update policies on 6 scenes acquired with InSight. `Constant` means the old method using a uniform pace update in which we ray cast the new model every $N = 5$ frames. `Adaptive` refers to the DVO-based key frame update policy [63] leveraging a entropy ratio α definition.

In Tab. 2.1 we report the results of our experiment. Overall the adaptive method is the winner: in two cases out of six it is able to align a higher number of frames while in the reminder of the scenes it performs on par with the uniform alternative but using from 2 to 4 times less updates. Overall, the most significant gain is expressed in terms of residual error during ICP alignment which decreased in five cases out of six. However, the improvement with respect to the original configuration seems to be non conclusive. Indeed, we still have cases in which the ICP was not conditioned well enough to succeed and the tracking was lost irrevocably. The novel updating strategy needs to be supported by an effective registration method but the current solution seems not robust enough. In the next section we investigate an alternative sampling strategy to try to improve the quality of the registration.

2.2 Using a tailored-random sampling strategy to prepare data for alignment

Since the new key frame update alone is not enough to avoid drastic lost of the camera track, we try now to investigate another critical step in our pipeline to see if we can benefit from its remodeling. Specifically, we are not happy with the current strategy adopted for sub-sampling the input range image before trying to align it to the key frame.

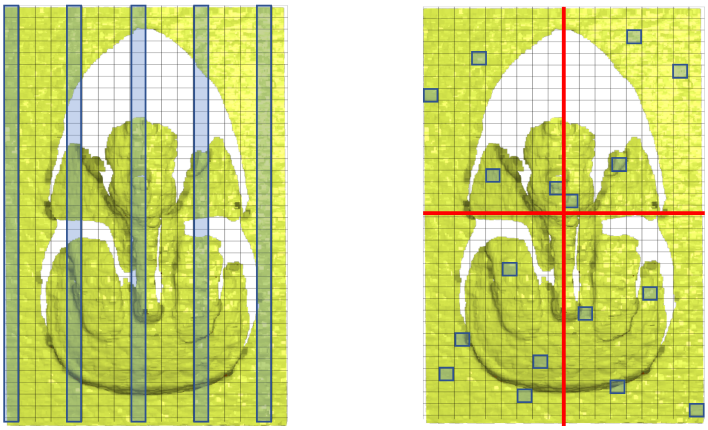
In practice, the resolution of our system is 1280×1024 pixels and we typically produce around 500k-700K valid points per RI. If we use a reducing factor equal to 5 along the columns of the RI, we usually end up working with a set of 20K-30K valid points. We point out that we need to keep the sub-sampling factor low due to the risk we can incur by using such a uniform pattern for filtering. Indeed, given the possibly high geometric complexity of the scene, we could fall into many sectors mainly containing invalid points (*i.e.* empty space) and consequently fail the tracking. Leveraging on a fixed pattern is a poor choice when we need a good level of generalization and we do not have *a priori* information about the shape we are going to acquire. Moreover, also the data redundancy can be critical: when the underlying shape has a smooth surface or a repetitive geometry, the uniform sampling could extract a lot of non distinctive points that could cause drifts during the alignment.

Ideally, a feature based approach should lead to the extraction of salient key points (see Sec. 3.1.1 for additional details on 3D detectors). However, this step involves a computational load which could be not sustainable in our real-time working scenario, therefore we need an alternative that is as fast and lightweight as the uniform sampling was. The solution is trivial but effective: we choose to extract the points according to a random distribution. A custom additional requirement takes the advantage of the properties of the range images that we have: the random sampling is performed only on the set of valid points after we split the 2D domain in 4 sectors. In Figure 2.11 we show a qualitative example of both methods applied to a range image acquired with the InSight (we represent the structured RI using a 2D grid on top of the 3D data).

The rationale behind this choice is that having points not belonging to a predefined regular grid we expect to cover a wider area with less redundancy in it, so that we should be more effective when solving ICP. Moreover, because of the reduced redundancy and the point validity constraint, we expect to be able to work with less points overall.

2.2.1 Experimental results

We repeat the same experiments we did for evaluating the KF update method. We set the maximum number of points to sample up to 150 for sector, *i.e.* up



(a) Uniform sampling (b) Random sampling

Figure 2.11: Different approaches for sub-sampling input range image before running ICP. In light blue we highlight the points selected by sampling the input RI.

to 600 points in total. This is an order of magnitude smaller on average with respect to the number of points we worked with using the uniform method. In Tab. 2.2 we report the results of the evaluation in comparison with what we obtained the previous run. In both cases we consider the better adaptive key frame selection strategy.

Model name	# Frames Acquired	# Frames Integrated		# KF Updates		ICP RMSE [mm]	
		Uniform	Random	Uniform	Random	Uniform	Random
`Marble`	85	85	85	16	16	0.37	**0.36**
`Dummy`	161	145	**161**	9	23	0.88	**0.34**
`Face 1`	239	204	**218**	10	10	0.72	**0.49**
`Face 2`	208	119	**205**	7	26	**0.29**	0.31
`Hand 1`	102	93	**98**	9	8	0.38	**0.36**
`Hand 2`	145	145	145	6	6	0.21	0.21

Table 2.2: Testing two ICP sampling strategies on 6 scenes acquired with InSight. `Uniform` refers to the original method we adopted for sampling, meaning picking one column out of 5 in the working range image. On the contrary, `Random` refers to method under evaluation were we split the RI in 4 sectors and we took up to 150 random samples per sector from the pool of valid points.

The table shows how the random sampling outperforms the previous uniform grid-based approach. In particular, we almost integrated all the frames in the testing scenes, overcoming the issues we had in the previous run when we drastically lost the tracking (especially for `dummy` and `face 1` scenes). Moreover, the residual error on ICP registration further decreased.

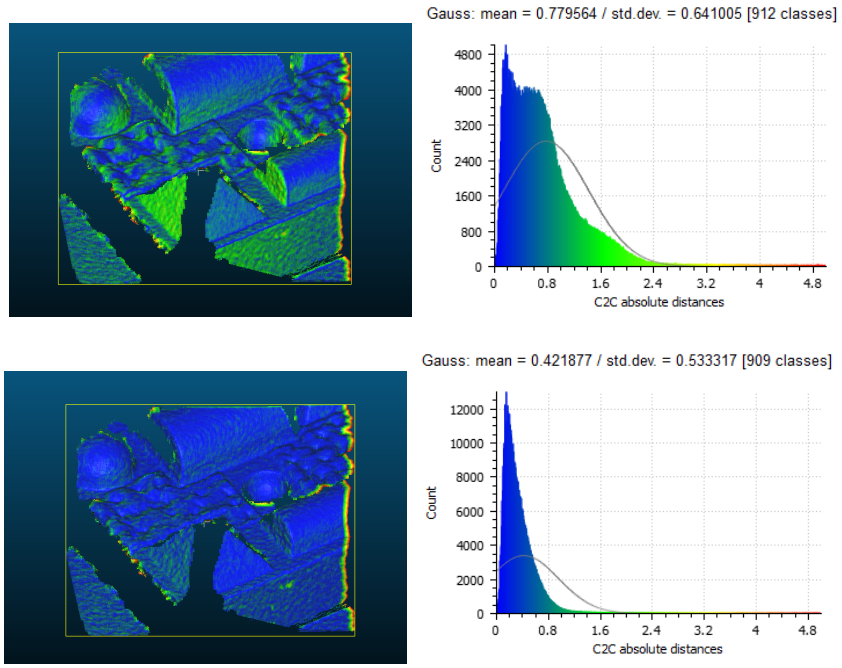


Figure 2.12: Point-to-point distances distribution after aligning frame 60 in `Marble` scene. (top) Alignment result obtained with a *constant* key frame update strategy and a *uniform* sampling of the input frame. (bottom) Alignment result obtained with an *adaptive* key frame update strategy and a *random* sampling of the input frame.

In Fig. 2.12 we also report a couple of examples for the outcome of ICP alignment on two frames expressed in terms of point distance distribution. In this case, the combination of the two new methods allowed ICP to converge to a better minimum than what it did with the original solution in place.

In conclusion, the changes we applied seem to be effective to improve the overall robustness of the tracking system. However, the pipeline still suffers from the remaining problems KinectFusion is also affected by. In particular, up to now we just focused on strengthening the only pairwise registration step into the frame-to-model alignment mindset. Indeed we have not addressed the issues related to the increasing drift, which is not completely solved with frame-to-model registration, as well as the lack of strategies both for an effective recovery from tracking loss (or from intentional acquisition resuming from another vantage point), and for repairing the model after a misaligned frame integration. All of these challenges are tackled instead in the work of Dai *et al.*, BundleFusion. In the following section we exploit such a complex solution to try to find a solution for these critical aspects.

2.3 Exploiting BundleFusion reconstruction pipeline

Up to our knowledge, BundleFusion [19] is the most sophisticated and robust solution available in the academic research to address online 3D reconstruction. We already addressed the key novelties of the method in Sec. 3.2.2, so we review them here very briefly avoiding to be redundant:

- A robust strategy for aligning frames based registration that leverages 2D SIFT features [78] for sparse matching and geometric and photometric consistencies for dense refinement.
- A hierarchical structure to decompose the problem complexity by using a local-to-global optimization strategy
- A re-integration strategy to enable on-the-fly model update and to recover from bad alignments situations.

The pipeline scales well and it is suited for dealing with large dataset reconstruction. A benchmark dataset is provided in the same reference paper [19]. BundleFusion dataset contains 8 scenes (3 apartment rooms, 1 copy-room and 4 offices) scanned with Structure IO sensor (see Sec. 1.3 for more details) and reconstructed using the pipeline above.

Because of the robustness and the promising model optimization feature the method presents, we then seek to evaluate the system performances when paired with the InSight data and its object reconstruction-oriented target application.

2.3.1 Source code

The code is available online [3]. Although the repository is not maintained anymore (last commit was pushed in 2018 and the code was originally developed using Visual Studio 2013), we were able to install it successfully on our PC running VS 2019 on a Intel i7-7820HQ with 64Gb or RAM. Most of the algorithms are developed using CUDA and the pipeline requires 2 GPUs to run the alignment and the integration processes in parallel. Our PC mounts a single NVidia GPU Quadro P5000 with 16Gb of dedicated memory, so it was possible to emulate the second GPU by splitting the memory in two distinct virtual instances of 8Gb each. We finally tested the standard configuration by reconstructing the `office3` scene from BundleFusion dataset itself. Our reconstructed model was indeed compatible with the result presented in the original paper when we set the voxel size equal to 1cm. However, we point out that we encountered some problems with smaller voxel $v = 4$mm. In such a case, the pipeline crashed during the process even by ensuring larger memory pre-allocation and reasonable estimation of the final number of voxels. The same problem is reported by other users on the repository, but

[3] `https://github.com/niessner/BundleFusion`

remained unanswered. Nevertheless, is fine enough for testing large indoor dataset, while we can still work with a smaller voxel size when dealing with InSight data, because our data have less frames and smaller volume in the end. We can then move on with our own evaluation.

In practice, in order to post-process a sequence of frames acquired beforehand with any kind of optical scanner, the pipeline accepts a specific data structure called *sens* (indeed it has `.sens` extension). The authors provide a reader[4] for the .sens data but no converter is given, up to our knowledge. However, we were able to develop a converter in Python by reverse engineering the original reader. After we extracted RGBs and Depth maps from our list of range images we converted them into a BF-compatible list of `.sens` files. Moreover, we wanted to test how robust BF is with respect to some critical track losses. Therefore we altered the original frames flow by switching two blocks of 10 images in the middle of the sequence. We refer to this scenes as *shuffled*. In front of the two blocks we also put a few void frames to simulate the scenario in which we voluntarily interrupt the scan, we point the camera into a slightly different area and we restart to scan again. The modification produces a drastic change of view during the reconstruction which can only be addressed by a robust registration technique. Although a misalignment could happen in this phase, we expect that the global optimization strategy can handle it and fix it on-the-fly as demonstrated in the original presentation during indoor reconstruction demos. Moreover, we also kept some of the original InSight data that we already tested in the previous section in order to evaluate the BF performance against the result we obtained with that configuration.

2.3.2 Experimental results

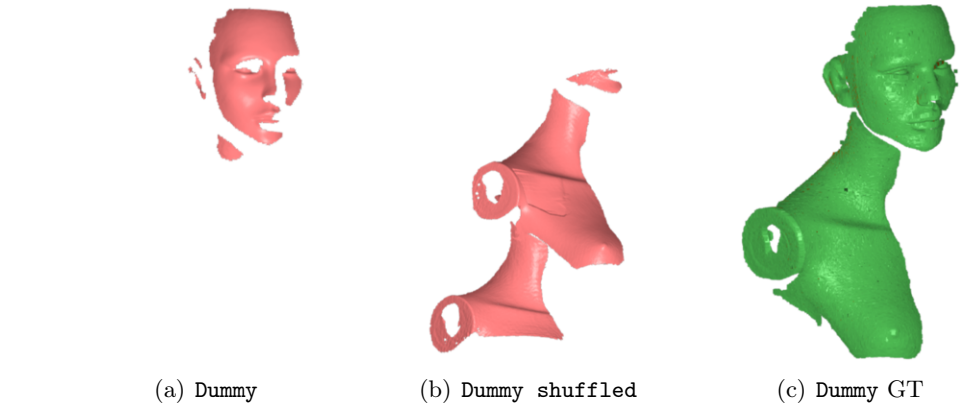
Before starting the evaluation, we properly tuned the settings via a configuration file. In Tab. 2.3 we report the main changes with respect to the standard configuration we used for reconstructing `office3`. First of all it has to be noticed that for our data we kept the original input image size for all the steps throughout the pipeline execution, except for the dense registration in which we down-sampled it by a factor 4. On the contrary, in the original implementation the authors decided to reduce the image by a factor 2 before integration and by a factor 8 before dense registration. We decided to avoid drastic down-sampling factors because we preferred to work with higher accuracy at the expense of some performance. Another difference is related to volume resolution: the original voxel size was set to 1 cm for 3DMatch, while we use 1.5 mm with the InSight. We can work with such a small voxel because the final volume will be much smaller as well and so we have no problem with the memory consumption (while with the original

[4] `https://github.com/ScanNet/ScanNet/tree/master/SensReader`

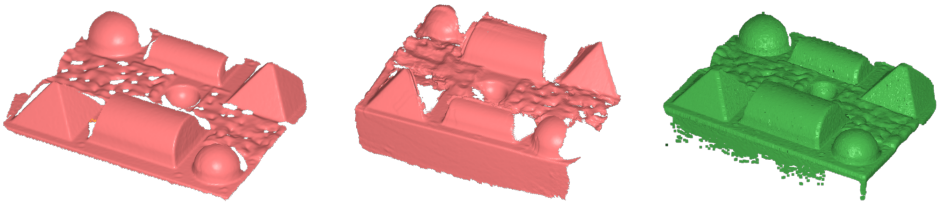
BundleFusion parameter	Parameter scope	Value per dataset	
		3DMatch	**InSight**
`image_res`	Input	640×480	1280×1024
`image_res`	Integration	320×240	1280×1024
`max_depth`	Integration	3.0 m	0.7 m
`sdf_voxel_size`	Integration	10 mm	1.5 mm
`image_res`	Matching	640×480	1280×1024
`min_area_corresp`	Matching	0.032 m^2	0.01m^2
`image_res`	Registration	80×60	320×256
`dense_dist_thresh`	Registration	150 mm	25 mm
`num_solves_final`	Optimization	30	50

Table 2.3: Set of BundleFusion parameters: comparing the values we used to work with 3DMatch [6] against the values we used to work with the data produced by means of our scanner.

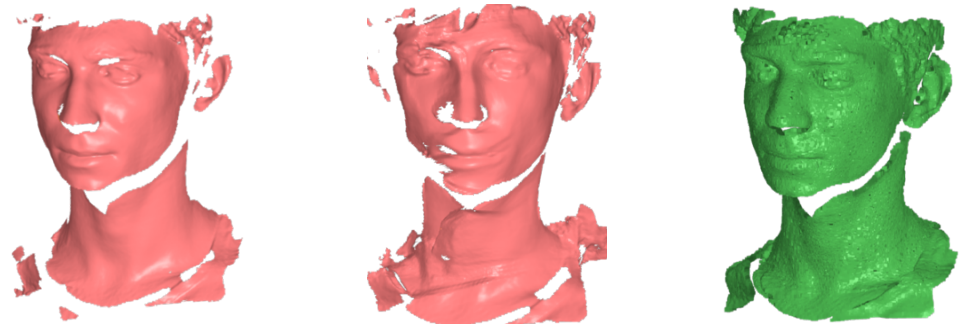
3DMatch scene `office3` the system crashed using 4 mm). Moreover, since the field of view is also reduced with the InSight, we then change the minimum area the keypoints have to span. Such a criteria is evaluated during sparse registration to avoid degenerate cases in which all the features are centered around a single spot. Because of the higher accuracy we seek, we also reduce some critical parameters for dense registration. In particular, we change the minimum distance threshold for dense evaluation of valid correspondences alignment from 15 mm to 2.5 mm. Finally, in order to address the fact that an artificial tracking lost was introduced in few scenes, we also increase the number of iterations we run the global solver after the end of the sequence for a higher probability of having a good global optimization in the end.



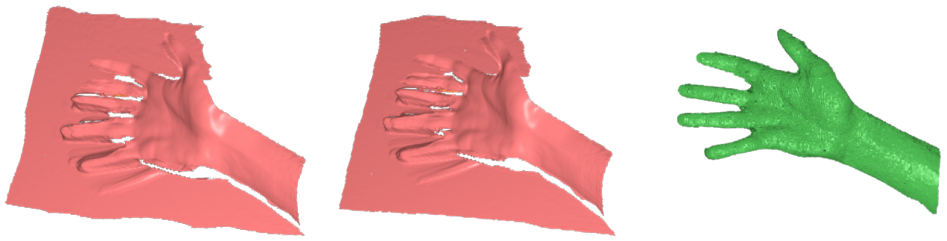
(a) `Dummy` (b) `Dummy shuffled` (c) `Dummy` GT



(d) `Marble` (e) `Marble shuffled` (f) `Marble` GT



(g) `Face` (h) `Face shuffled` (i) `Face` GT



(j) `Hand` (k) `Hand shuffled` (l) `Hand` GT

Figure 2.13: BundleFusion [19] results with InSight scenes (*shuffled*: variation of the original frames sequence, GT: ground truth).

In Fig. 2.13 we report the visual results we obtained reconstructing the InSight data with BundleFusion, compared against their ground truth models. Surprisingly, BF under-performed with respect to the initial expectation. Specifically, most of the scene we tested end with one of these three types of errors. In the first case, the track was lost after a while and never recovered back, so that the final model shows no misalignment but it remains partial, as for the `dummy` scene. On the contrary, `dummy shuffled` scene presents two distinct sub-volumes, indicating that after the artificial track loss the reconstruction moved on recovering the track and restarted building on a misaligned chunk with respect to the reference system. Additionally, the final optimization did not help to solve the issue. Finally, in the last case the pose optimization failed to adjust several frames that were wrongly integrated at first, reasonably due to a drift in the pairwise registration during chunk creation. Moreover, it is interesting to notice that we failed to reconstruct not only the *shuffled* cases but also the *standard* scenes having a regular flow for the frames in the sequence.

Our assumption is that the main issues occurred during the sparse-to-dense registration step. We point out that the errors happened regardless the configurations we adopted for the reconstruction. Indeed, we performed a grid search on the most critical settings and the output we report is the best we got. Nevertheless, these parameters sounds reasonable to us, therefore we think the problems arose for specific issues of the method with our data type. Specifically, we think that the leverage of a 2D feature-based sparse matching on scenes like the ones the InSight produces can be tricky. In Fig. 2.14 we show how the matching stage behaves on two very different data. In the first row we report the case of the `Office3` scene in 3DMatch dataset. The camera is framing a table with several objects on it. The initial set of raw matches is quite big. Such a set of putative correspondences needs then to pass through 3 filtering stages. We quickly review what kind of filters are adopted in BF:

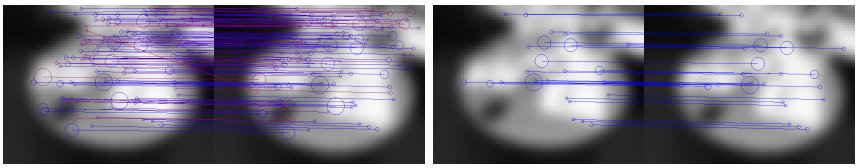
- A *brute force filter*, which works by the reiterated application of the Kabsch algorithm to estimate the matrix to align the two views. At each iteration, the estimated matrix is applied to the frame and the distance between all the putative correspondences frame-model are evaluated. The ones above a preset threshold are removed and the Kabsch algorithm is run again. The process goes on until no outliers remain in the list. If less than 4 correspondences remain, the frame is skipped.

- A *surface area filter* that evaluates that a minimum area is spanned by the keypoints which survived to the previous filter. If not, the frame is skipped.

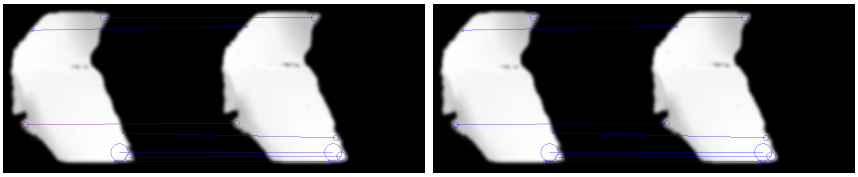
- Finally, a *dense verification filter* controls that geometric (depth and normals) and photometric (color) consistencies are preserved on a dense

evaluation (*i.e.* pixel-wise). In order to guarantee a real-time performance the input image is heavily down-sampled at this stage.

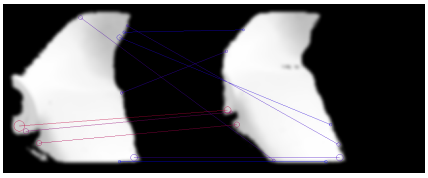
In Fig. 2.14(b) we see that after the skimming of the raw matches between the two images of the table, a sufficient amount of matches survived. On the contrary, in the last row of the same figure we have an example of the `Dummy` scene which shows much less raw matches at the beginning and no final result is reported because the brute force filter removed all of them. This is considered a failure in the pipeline and the frame is then skipped. This kind of failure occurred several times with our data throughout all the test we made. Even when a valid set of matches is found, as for the case shown the middle row of Fig. 2.14, the amount of matches is very poor.



(a) `Office3` raw (b) `Office3` final



(c) `Dummy` raw (d) `Dummy` final



(e) `Dummy` raw (failed)

Figure 2.14: Example of sparse matching using SIFT [78] in BundleFusion [19]. (a-b) Raw matches and final correspondences after filtering for two fragments from `Office3` [6]. (c-d) Result on two fragments from `Dummy` [J2]. This case appears more challenging and very few matches are found indeed. (e) In this case the raw correspondences are mostly outliers and the filtering stage reject the pair. We suggest to look at the picture in the electronic version.

2.4 Conclusions

In this section we discussed about the main flaws of the current solution we employ to run the InSight 3D reconstruction pipeline. In particular, we focused on the issues related to the camera tracking frequent losses, the lack of a smart method to relocate the camera in a fast and robust manner after such loss and the possibility to have an optimization module to adjust the already integrated frames on-the-fly when a bad estimation occurred in the first place, as well as to refine all the poses to solve the problem of the accumulated drift.

In order to face these challenges, we dived into the classical solutions we can find in literature and we tried to take advantage from that to boost the quality of our reconstruction. Specifically, to improve the tracking robustness we modified two critical policies in the current registration pipeline. The first one is the strategy we adopt to decide when the model needs to be updated. We moved from a trivial approach where the update occurs at a constant pace with a dynamic solution based on the evaluation of the ratio of the differential entropies of the camera motion estimation. The method is more effective in understanding when the registration is suffering from a rapid change of the geometry of the current frame with respect to the model. The second policy we changed is the method we use to down-sample the working range image to speed up the ICP algorithm. Instead of a uniform sampling we adopted a custom random sampling which allows spanning all over the RI with less redundancy. By leveraging on a much smaller set of points we were still able to integrate a significantly higher number of frames with respect to the original solution, by also reducing the residual error of the alignment. Then, in order to address the problem of camera relocation and online model refinement, we exploited the potential of BundleFusion. The method is widely considered the state-of-the-art for real-time reconstruction of large scenes and theoretically offers all the features we are interested into for our own device. However, the test we made highlighted the fact that leveraging solely on 2D feature extraction to address the sparse and coarse registration is critical with our data which are less informative when small regions with very poor texture are framed during the reconstruction of a single object. Although BundleFusion is very close to what we would like to achieve in the end, we still need to investigate an alternative solution to tackle the problems we encountered.

Motivated by the lack of an incremental solution which can solve in a conclusive manner our problems with the InSight, we need to look elsewhere to find new methods and new approaches to tackle the issues. Due to the success of recent data driven solutions addressing several computer vision-related tasks, we then decided to explore the deep learning research field. However, before diving into the literature we wanted to have a reliable dataset to test the networks that we eventually would find promising. To do

that, we refined the initial set of scenes we acquired for the previous tests we made and released a rich collection of models specifically acquired with the InSight. In the following section we present this collection.

3 DenseMatch: a novel benchmark dataset for dense scanner acquisitions

In the previous section we saw that we are far from having a perfect solution to solve all the issues we have with data generated by scanners such the InSight. However, before moving on with other tests we decided to improve the quality of our benchmark. In the first chapter we discussed about the type of datasets available online which address the problem of 3D reconstruction and we pointed out that we did not find anything similar to our own kind of data. Moreover, having a dataset tailored to a specific target application is nowadays more important than ever because of the large impact that data-driven research has on our field. In the next chapter we will deepen the topic of deep learning applied to 3D related problems. In such a context, it is crucial to have a real reference dataset to test any kind of model to evaluate how well it behaves in our working scenario and eventually to be able to fit a specific architecture to our own problems.

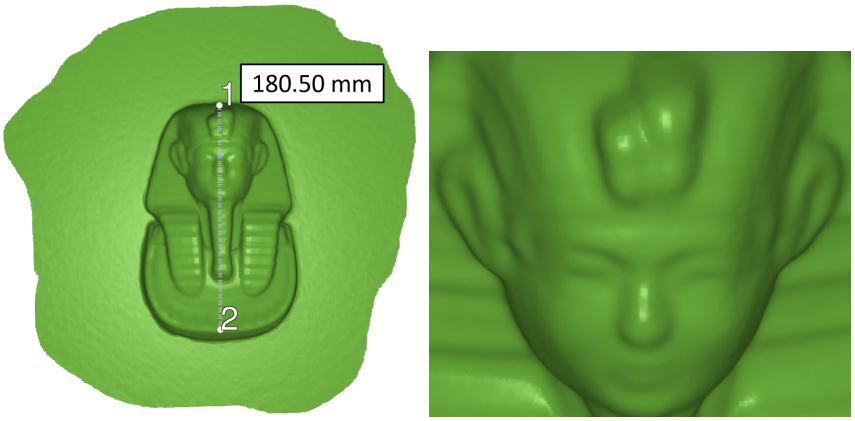
Starting from these considerations, we decided to propose a collection of small-scale scenes acquired with the InSight, in order to provide good fresh accurate dense data to the community which could benefit for investigating dense 3D object reconstruction problems. We call this dataset `DenseMatch`. Finally, a publication [J2] was made out of this project.

In the following paragraphs we try to answer to few specific questions which are helpful to understand everything about DenseMatch. Basically the questions we want to answer are *why is it useful?*, *what is it?* and *how did we create it?* and *how it can be used?*.

Why do we need this dataset? We summarize the contribution of our dataset in four bullet points.

- It is useful to develop models and alignment strategies to automatically reconstruct 3D scenes from data acquired with high resolution optical scanners.
- The high resolution optics mounted on the scanner provide a very dense 3D data, which is not comparable with other data acquired with low cost optical scanners we can find in literature so far. From this density property we derived the name of *DenseMatch*. A visual example of it is reported in Fig. 2.15.
- It can be used by computer scientists to perform quantitative analysis of 3D reconstruction and alignment methods.

- By using these data it is possible to train or fine-tune AI models to cope with small-scale and dense 3D object acquisitions and reconstructions.



(a) `Pharaon` dimension (b) `Pharaon` closeup

Figure 2.15: Example of a scene in DenseMatch. The size of the object is relatively small (it represent a statue 18 cm long). The closeup shows the level of detail we can reach on the face of such a small statue.

What is this dataset? The dataset comprises 19 scenes which are made by acquiring 10 different subjects using our scanner. Our collection of scans represents a realistic target application for the scanner we are developing, namely the acquisition of human bodies and faces, reproduction of artworks, and design objects. We depict these scenes in Fig. 2.16. Moreover, each scene is made of a sequence of RGB-D images that can be used to extract a list of point clouds with more than 500k points on average. In total we created 3140 of such point clouds.

	#	**min**	**max**	**avg**
Scenes	19	-	-	-
Point Clouds (PC)	3140	-	-	-
Points per PC	-	60k	1M	500k
BBox Volume per PC	-	10^{-4} m^3	10^{-2} m^3	3×10^{-3} m^3
Points Spacing	-	0.18 mm	0.35 mm	0.25 mm

Table 2.4: DenseMatch main specifications. BBox stands for Bounding Box. The oriented BBoxes are estimated using Open3d [55] as well as the average points inter-distances that have been used to define the spacing.

A report with all of these and additional information is also provided in Tab. 2.4. From the table we notice that the raw data streamed during the scan is remarkably dense. Indeed we already mentioned the fact that each

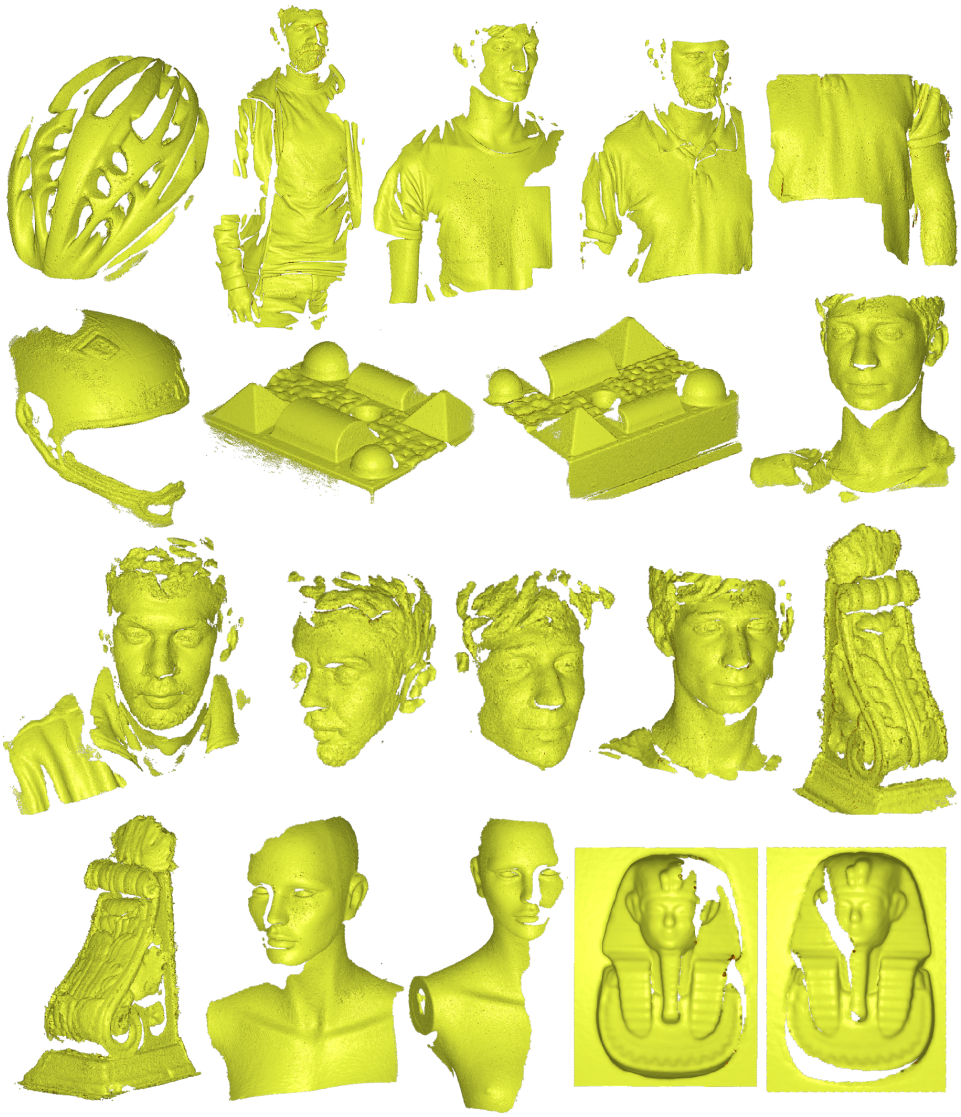


Figure 2.16: DenseMatch dataset scenes.

depth map produces 500k points on average. Additionally, such a relevant amount of points is bounded into a small volume: the size of the bounding boxes on average gives a hint about how large each scan actually is and this value is in agreement with the small field of view of the tracking system, which is equal to 250×200 mm. The number of points per frame, together with the spacing of the points, provides also a hint of the point cloud density.

How the dataset was created? To create the dataset, we used the pre-market prototype InSight we already introduced in the previous section of

this chapter. Our setup comprised a machine running Windows 10 with 16GB of RAM and an Nvidia GTX 1080 Ti. We acquired multiple scenes by scanning different subjects. During the scanning we stored on disk the flow of frames the system was grabbing, both the valid ones (meaning the frames we tracked successfully and also integrated in the model under construction) and the non valid as well.

Then, in post-processing we proceeded as follows: first, we extracted a point cloud for each frame. We then performed a direct alignment for all the point clouds composing the scene, by following the implementation of [91], which is a feature-based pipeline that leverages ICP [124, 125] for 3D alignment. Moreover, aiming at creating a reference without loss, we fixed by hand the failed alignments that occurred during the process by running again the pipeline with user-selected control points for the critical point clouds. Finally, we refined the result in a global fashion by leveraging on an optimization-on-a-manifold framework [66]. Eventually, the final set of point clouds is properly aligned, thus constituting the ground truth for different kinds of tests on 3D object registration or for training/fine-tuning models to cope with small-scale 3D objects and dense acquisitions.

How the dataset can be used? The dataset can be accessed from an open source repository[5]. In practice, the repository contains two types of file:

- List of *npz* binary format files (18.4Gb in totals) created using Python Numpy open source package [153]. Each file refers to a scene and it contains all the point clouds (in standard *ply* format[6]) and additional information (frames names and poses)..
- *tar.gz* is an archive (1.8Gb large) compressed using gzip algorithm, so that it can be unzipped with any standard decompression tool. Each folder contains the RGB-D images and the camera calibration parameters that were used to reconstruct the 3D data. These images are the original output produced by the 3D scanner that was used to generate the data. Specifically, the color images have *.jpg* extension, while the depth images are 16 bit *.png* files.

In the dataset repository we also provided a link to an open source code[7] for a custom reader we implemented in Python. The reader is a command line application which allows to specify different settings. For instance the user can choose to read a point cloud directly or to re-create it from scratch using the RGB-D images of a specific scene. Specifically, each RGB-D is wrapped into a `Frame` structure that contains color and depth images, camera pose

[5] `https://doi.org/10.7910/DVN/CU4UXG`
[6] `http://paulbourke.net/dataformats/ply/`
[7] `https://doi.org/10.5281/zenodo.5534851`

and point cloud members. Each frame can also be saved into a `Sequence` class which contains the camera parameters and additional metadata. Such a structure can be easily integrated with any research code the user intends to develop. Finally, the repository contains also a simple example to show how to load a sequence and visualize it. Everything is based on few popular Python packages and the code should run effortless on Windows, Linux and MacOS.

We already used DenseMatch in two studies [C3], [J1] where we focused on DL-based 3D alignment and real-time 3D reconstruction (as we will see in Chapter 3). In these works, the point clouds were initially misaligned by means of a supervised random perturbation of the ground truth poses. In particular, the random rotations span the whole range of angles from 0 to 2π onto the 3 axes. Additionally, a random translation is produced, spanning the the 3 axes again using a range of ± 1 m. The misaligned data serves as input to 3D reconstruction pipelines addressing the problem of 3D registration where we perform a cross-domain assessment of deep learning-based solutions. We point out that the set of transformations applied in the above-referenced works is also available as metadata in the provided dataset.

4 Conclusion

In this chapter we presented the InSight, a prototype of an handheld optical 3D scanner which is under development at FARO, the company where I worked during my PhD. This device is meant for specific applications which require a high resolution acquisition, but still maintaining the flexibility of an handheld solution. We first reviewed the technical aspects of the scanner and we discussed about the issues that affect the current pipeline which follows closely the implementation proposed by the authors of KinectFusion [20]. In order to deal with these issues, we then proposed a couple of approaches to revise two elements which are critical in the current registration framework. Specifically, we tried to improve the robustness of the tracking by replacing the current policy for key frame updating in our frame-to-model strategy by exploiting an adaptive solution based on the evaluation of the differential entropy for the motion estimation of the camera. Moreover, we also revised the down-sampling method we adopt to extract a subset of points from the current range image to align with the model using ICP algorithm. These modifications produced a boost in terms of robustness and allowed to recover a significant amount of frames we initially lost with the original workflow. However, it is clear that these adjustments are non conclusive to provide a pipeline which is robust enough to deal with tracking loss. Some problems remained unsolved, therefore we tried to investigate a completely different pipeline with respect to our KinectFusion-based one. We then deepened BundleFusion [19], which is the current state-of-the-art

for real-time indoor reconstruction. Although the method sounds promising on paper, we experienced a lot of issues trying to cope this pipeline with our own data.

In the end, we concluded that it could be helpful to search elsewhere to find suitable solutions for us. Indeed, we planned to explore the world of deep learning, encouraged by the successful results obtained with data-driven technologies and noticing their trends in moving towards the 3D domain.

Finally, before diving into studying the topic, we decided to propose a new dataset, DenseMatch. This dataset targets the type of reconstruction of our interest, which was missing from the literature but represents a branch of applications of growing interest for the community. This dataset will be leveraged in the following chapters, while all the examinations we made so far remains valid for further discussion.

Chapter 3

Deep learning applied to point cloud 3D registration

Although we already extensively discussed about 3D registration of views in Chapter 1 Sec. 3.1, we just reviewed the classical approaches so far, while we have not considered data-driven alternatives yet. However, the recent trend sees a collective effort in exploring the power of deep learning (DL) to make high-capacity models able to understand articulated tasks such as object detection [154, 155, 156], semantic segmentation [157, 158, 159], shape retrieval [160, 161], shape completion [162, 163, 164], as well as point cloud rigid and non rigid registration [16, 17, 18, 165, 166].

Two key aspects make these data-driven solutions extremely appealing in our context:

- The deep understanding of complex problems allows to provide **robust** methods that can deal with the most critical challenges we can typically find when dealing with 3D domain tasks. Effective models can in theory generalize well hard problems and so they can offer more robust results than their handcrafted counterparts.

- Exploiting the parallel computation capability to activate millions of neurons simultaneously, some of these networks can process deep models at a very high speed, making them not only robust but extremely **fast** and possibly better than others in case of real-time constraints.

Due to these reasons, we thought it was worth it to deepen the topic to understand whether we can leverage on these technologies. In particular, we aim to revise some of the methods we earlier adopted to develop the reconstruction pipeline of the scanner we introduced in the previous chapter.

Therefore, we organize the chapter as follows: in the first section we introduce the background of 3D deep learning (DL) by focusing on the approaches required to deal with point clouds, which are a natural choice for 3D acquisition and reconstruction and are the data type we can extract from

the range images during the scan with the InSight (see Chapter 2 Sec. 1.2). We point out that for the sake of readability, some of the elements of 3D DL are omitted here. In order to fully understand the main concepts considered in this chapter, the reader should be familiar with the very basic concepts of DL. Nevertheless, we cover most of them in the Appendices Sec. B, so we suggest to take a look at it before continuing with this chapter if needed.

Then, in the second and third section we review how DL deals with 3D registration. We follow a pattern similar to the one we proposed for standard solutions in Chapter 1 Sec. 3.1, starting from feature detection up to final transformation matrix estimation, and we deepen some of the most promising methods in the process.

Finally, in the last section we present a comparative assessment of some of main DL-based alignment solutions proposed in literature, considering a real-time reconstruction framework. The assessment is cross-domain, meaning that we evaluate how these methods perform with both "standard" benchmark data (we leverage on the already mentioned 3DMatch to do that) and InSight data (by considering the novel DenseMatch dataset). We do not limit ourselves to a simple comparison, but we run performance tests and extended comparisons, with different system configurations including model refinements, and we found solid evidence that the generalizability performance of deep learning systems for 3D alignment is critically linked to data features. The results of this in-depth assessment ended up in two publications ([C3] and [J1]).

All the considerations we make here are preparatory for the last chapter in which we will discuss our final real-time pipeline aiming at the improvement of the overall robustness and reconstruction performance of handheld 3D object scanning.

1 Background

Just recently, DL started being used to address tasks in 3D domain having point clouds as input for deep models. The work of Qi *et al.*, PointNet [9] is referenced as the first attempt to fit a deep model to deliver suitable applications with point clouds. Indeed, point clouds are critical because of their inherent unstructured (unordered structure) nature. In practice, a neural network should be able to produce consistent results with invariance to the order of the elements that compose its input. This means that the network needs to be permutation invariant. The problem is depicted in Fig. 3.1. Moreover, similarly to what happens with handcrafted solutions, the network should also be invariant to any sort of affine transformation. To cope with this requirement, a trivial solution is to apply several transformations to the original dataset during training stage to allow the network to face all the possible scenarios. However, it is critical to exploit all of the possible

cases and so a more robust approach should be preferable.

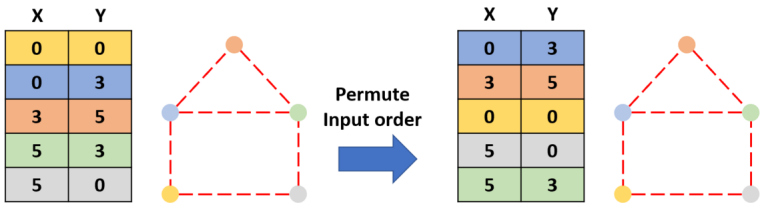


Figure 3.1: Visual representation of the permutation problem for neural network applications with unstructured input data. The network must be invariant to the order of the coordinates, because in the end the two list of coordinates represent the same model (here in 2D, but it applies to higher dimensions as well).

Convolutional neural networks (CNNs) [167, 168] are a powerful alternative which are able to empirically achieve strong rotation invariance and usually require less data augmentation [12]. In the reminder of the section we review the two main strategies currently adopted in DL frameworks to deal with point clouds-related problems.

1.1 PointNet: learning to manage point clouds using fully connected networks

PointNet is meant to work with point clouds since it addresses the problem of both order and rigid transformation invariance effectively. In Fig. 3.2 we show the original architecture. Specifically, the permutation invariance in PointNet is achieved since the model is able to represent a family of symmetric functions (*i.e.* producing the same result irrespectively of the order of factors). This is guaranteed by leveraging on a simple symmetric function (maxpooling) which acts on the output of a two-stage dimensionality augmentation operated by multilayer perceptrons (MLP) where, at each stage, weights are shared among MLPs for every input element. The max pooling is used to extract global context from the dense set of points. Moreover, a transformation network (T-Net), which is basically a mini-PointNet, is used to align the input to its canonical space in order to be invariant to rigid transformations, similarly to what it is done for handcrafted solutions where the eigen-values are used to determine the principal components of local reference frame [89, 90].

PointNet is easy to generalize to any kind of problem in which a high dimensional unstructured input needs to be feed to a classification or segmentation network. Indeed, since its proposal, it has been declined in several flavors to tackle different tasks sharing one common denominator: the necessity to make a deep network to learn the information content the point cloud carries within its spatial and geometrical relationships and manage

it according to the application. In origin, PointNet has been proposed for tackling object classification and semantic segmentation of point clouds. The classification block receives a tensor of dimension $N \times 3$ as input and then outputs a k-dimensional vector which represents a likelihood. Indeed, each element of the array refers to a predefined class and the value stored in it is the probability of the point cloud to belong to such class. The segmentation block instead takes as input the output of the second transformation network inside the classification block and it concatenates such tensor with the global feature used to represent the entire set. The output of this block is a $N \times M$ tensor which assigns to each point a M-dimensional vector score that can be used for clustering.

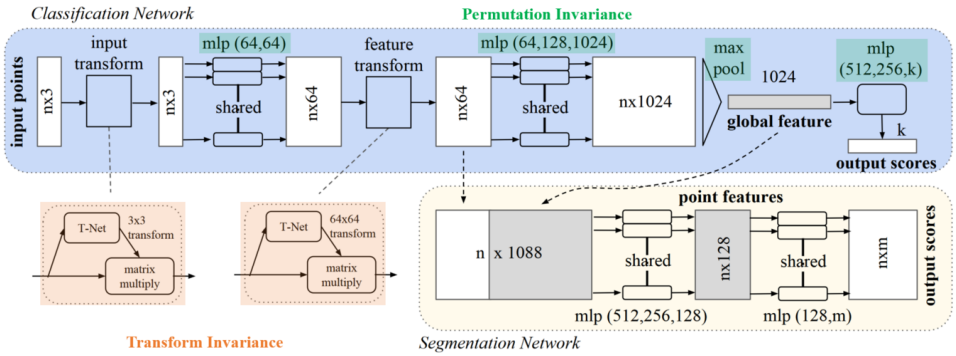


Figure 3.2: PointNet architecture. Image taken from [9].

The main limitation of PointNet is known to be the lack of a local context, which is lost when fully connected layers (FC) and max pooling are used to extract global information. A partial mitigation to this problem is given in PointNet++ [10], where the authors of PointNet address the local information preservation by sampling and grouping cluster of points and recursively applies PointNet on these support regions (Fig. 3.3), combining their output with the global information. Therefore, PointNet++ is a hierarchical feature learning method.

Although the network gained a huge success, it is not flawless. Indeed, since it adopts full connections, it cannot scale effortlessly. In order to deal with restrained sized input, a down-sampling policy or even a partitioning of the input usually takes place. However, this is a critical because it limits the spatial context. Moreover, inconsistent sampling density is common when dealing with partial point clouds, due to perspective effect, radial density variation and motion. Density variation affects hierarchy, which is sensitive to point uniformity. Finally, such a patch-based processing is inefficient because the network activations are not reused across adjacent patches within the intermediate layers.

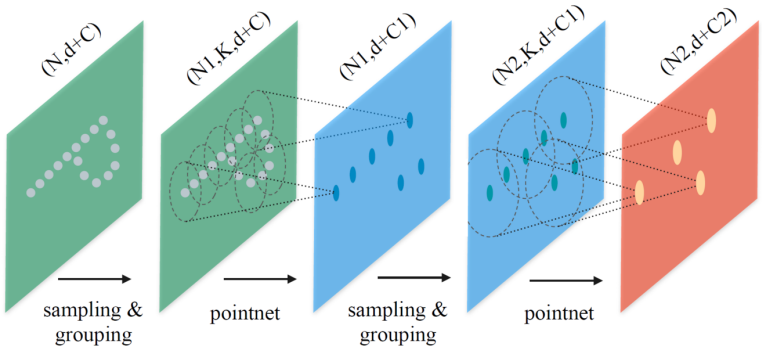


Figure 3.3: PointNet++ grouping and sampling. Image taken from [10].

1.2 Convolutional neural networks applied to point clouds

The introduction of convolutional neural networks [168] is considered a game changer for DL. In essence, convolutions allow building networks that are able of capturing broad context and are faster and more memory efficient than fully connected networks. Because of these reasons, they sound optimal to deal with 3D problems. Unfortunately, the main issue is that they need to work with a structured input, which point cloud is not. Therefore, in order to bring the advantages of CNNs in 3D, the initial attempts coped with alternative data types, such as 2D maps of 3D domain [2, 3, 4, 5] and volumetric representations of 3D space, using voxels-based regular grids [6, 7, 8]. However, the former tends to loose spatial relationship throughout the mapping while the second works with a dense grid which is also highly inefficient in terms of memory footprint.

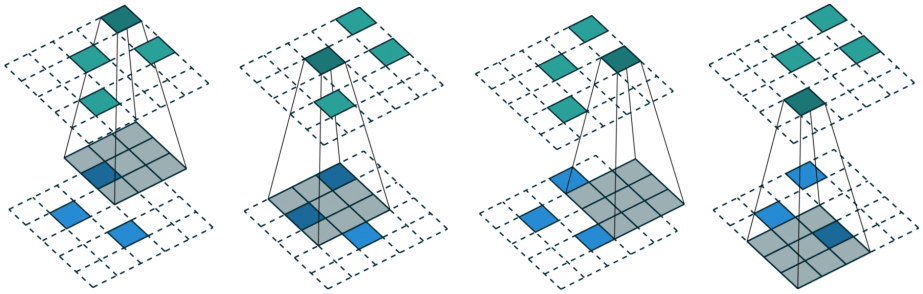


Figure 3.4: Example of sparse convolution with arbitrary in/out coordinates dimension.

More recently the use of sparse convolutional layers has been proposed to deal with irregular structures such as point clouds and meshes [169, 12, 11, 170]. These solutions exploit the local information of a point by means of sparse kernels defined via sparse tensors which are then applied during the

convolution process. We visualize a sparse convolution on a sparse tensor in Fig. 3.4. For the sake of an easy representation we keep the domain to 2D, but it can be easily extended to higher dimensions. Overall, the process is similar to the dense counterpart (see Appendix B), whereas in the sparse case the output is evaluated only on a few specified points which are controlled in the generalized convolution. Specifically, the conventional discrete dense convolution in D-dimension can be defined as:

$$\mathbf{x}_{\mathbf{u}}^{out} = \sum_{\mathbf{i} \in \mathcal{V}^D(K)} \mathbf{W}_{\mathbf{i}} \mathbf{x}_{\mathbf{u}+\mathbf{i}}^{in} \quad \text{for} \quad \mathbf{u} \in \mathbb{Z}^D \tag{3.1}$$

where $\mathbf{x}_{\mathbf{u}}^{in/out} \in \mathbb{R}^{N^{in/out}}$ is a $N^{in/out}$-dimensional feature vector in a D-dimensional space at D-dimensional coordinate $\mathbf{u} \in \mathbb{R}^D$ and $\mathbf{W} \in \mathbb{R}^{K^D \times N^{out} \times N^{in}}$ are the convolution kernel weights. In dense case $\mathcal{V}^D(K)$ is the list of offsets in D-dimensional hypercube centered at the origin (*e.g.* $\mathcal{V}^1(3) = \{-1, 0, 1\}$. We can then relax Eq. 3.1 to a general form:

$$\mathbf{x}_{\mathbf{u}}^{out} = \sum_{\mathbf{i} \in \mathcal{N}^D(\mathbf{u}, \mathcal{C}^{in})} \mathbf{W}_{\mathbf{i}} \mathbf{x}_{\mathbf{u}+\mathbf{i}}^{in} \quad \text{for} \quad \mathbf{u} \in \mathcal{C}^{out} \tag{3.2}$$

where $\mathcal{N}^D(\mathbf{u}, \mathcal{C}^{in})$ is the set of offsets from the current center $\mathbf{u}$ that exists in the predefined input coordinates $\mathcal{C}^{in}$ of sparse tensors. It is interesting to notice that when $\mathcal{C}^{in} = \mathcal{C}^{out} = \mathbb{Z}^D$ and $\mathcal{N}^D = \mathcal{V}^D(K)$ then we have the conventional dense convolution again. Finally, in order to efficiently compute the sparse convolution, it is necessary to determine how each non-zero element in an input sparse tensor is mapped to the output sparse tensor. Since the output is mapped throughout a kernel the mapping process is called kernel map [12].

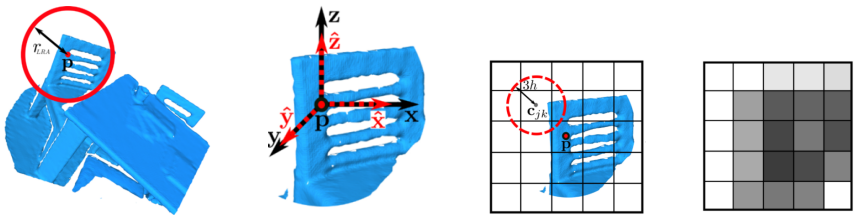
On top of this architecture novel methods have then been proposed in recent publications as alternatives to PointNet-based solutions to address several tasks: in the following sections we review the ones related to 3D point clouds registration, namely keypoints detection, feature description and transformation matrix estimation.

2 Learning to extract features from point clouds

Keypoint detection is a well established research topic in 2D and many successful DL-based architectures have proved to be valid and capable of outperforming handcrafted rivals [171, 172]. In 3D domain, conversely, the focus has been more on description than detection so far. The Unsupervised Stable Interest Point detector (USIP) [173] is one of the first detectors to appear in literature. It provides an unsupervised feature proposal network to detect highly repeatable and accurately localized keypoints. Its proposal network is a PointNet-like architecture. Similar to USIP, 3DFeatNet [174] is a network made of fully connected layers that performs a joint keypoint detection

and feature description, even if it focus largely on the latter. More recently D3Feat [14] also offers a simultaneous detection and description but instead of relying on MLPs and max pooling it exploits KPConv [11], a solution for flexible and deformable convolution of point clouds, as backbone for sparse convolutions.

Instead, we can find many solutions in literature to tackle the problem of 3D feature description [175]. A first bunch try to embed the spatial relationships of local neighborhoods by training a network to extract high dimensional descriptors from support regions. Different approaches can be used to encode the local information. For instance, ShapeNets [7], 3DMatch [6] and 3DSmoothNet [176] propose to use a uniform grid-based support to leverage standard 3D convolutions. In Fig. 3.5 we show how the input parametrization for 3DSmoothNet works. The output of such parametrization is a voxel grid which is passed to the actual network, shown in Fig. 3.6.



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

Figure 3.5: Input parametrization for SmoothNet. (a) A spherical support is defined around an interest point **p**. (b) The neighborhood is used to estimate the local reference frame (inspired by TOLDI method [89]) and the support is transformed to its canonical representation. (c) The support is voxelized using a Gaussian Smoothing Kernel. (d) A smoothed density value is used to normalize the voxel grid which is passed as input to the network. Image from [176].

However, such methods hardly cope with real-time requirements because of the overhead that building the local voxel grid implies, as well as the estimation of a local reference frame for each support. A different set of solutions avoid the inefficient volumetric structure and work with point cloud directly. In order to deal with such an unstructured data, we have seen already how the most frequent approach is to leverage a PointNet-like architecture. We mentioned earlier 3DFeatNet [174] which uses its method to detect keypoints and to extract features in outdoor scenes. Khoury *et al.* propose CGF, a very compact descriptor [177] extracted by mapping 3D oriented histograms to a low-dimensional feature space using MLPs.

PPFNet [178] is another solution that leverages PointNet but it encodes precomputed features instead. The method requires a preliminary step to

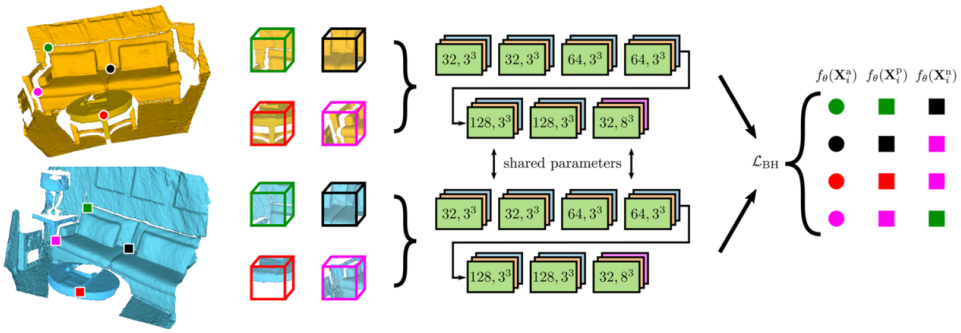


Figure 3.6: 3DSmoothNet [176] architecture. The inputs are two mini-batch of 3D SDV voxel grids. The network has a siamese architecture. Each sub-architecture consists of 3D dense convolutional (green rectangle), batch-normalization (orange), ReLU activation function (blue) and $\ell 2$-normalization (magenta) layers. $\mathcal{L}_{\text{BH}}$ is the batch hard loss working with the anchor $f_\theta(\mathbf{X}^{\text{a}})$ the positive $f_\theta(\mathbf{X}^{\text{p}})$ and negative $f_\theta(\mathbf{X}^{\text{n}})$ examples. Image taken from paper.

split the input point cloud in several local support regions containing a set of coordinates $\mathbf{C} \in \mathbb{R}^{\text{N}\times 3}$ (including an anchor). After grouping, a set of point pair features (PPF) [97, 179] is extracted from each region which is now represented by $\mathbf{F} \in \mathbb{R}^{(\text{N}-1)\times 4}$. The choice of PPF is due to the fact that such a descriptor is invariant to rotation transformation, therefore it can skip the LRF estimation (see Chapter 1 Sec. 3.1.2 for further details). Then, instead of encoding these features into a histogram-based representation, as proposed by PFH and FPFH [97], PPFNet uses a PointNet-based encoder to extract the final 128-dimensional descriptor. A local-to-global approach is adopted during the encoding, by mixing concatenations and pooling of layers as shown in Fig. 3.7.

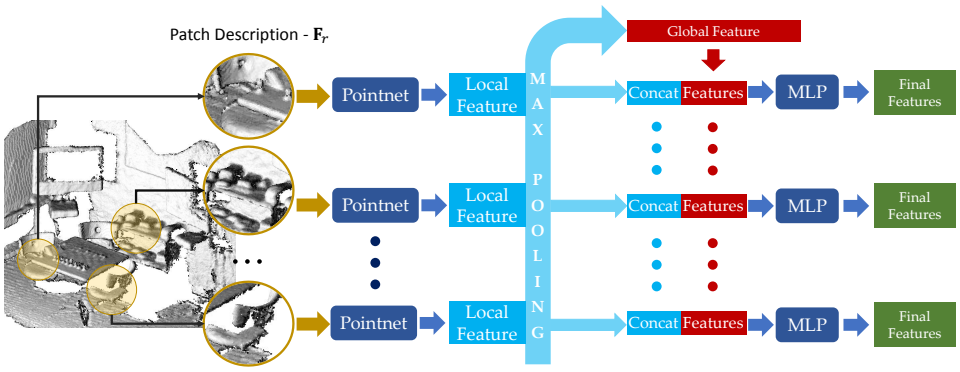


Figure 3.7: PPFNet [178] architecture. Image taken from paper.

A variant of PPFNet, is PPF-FoldNet [180] an unsupervised learning

method for 3D local descriptors based on the auto-encoding of point pair features by means of FoldingNet [181], a folding-based auto-encoder which encapsulates point pair features in an end-to-end trainable network. CapsuleNet [182] propose an auto-encoder designed to process sparse 3D point clouds while preserving spatial arrangements of the input data. The encoder is a capsule network using MLPs along with a dynamic routing scheme and the peculiar 2D latent space to approach several point cloud-related tasks, such as object classification, object reconstruction and part segmentation.

Instead of using fully connected layers, D3Feat [14] employs fully convolutional ones, in the attempt to add the knowledge of the spatial locality and to combine the detection of points of interest with their description. On the contrary, Choy [15] leverages on a reduced computational complexity via the sparse convolutional layers [12] and suggests using sparse tensors to tackle the problem on the data as a whole. Other promising works have been recently presented. These exploit a complex cylindrical convolution for an advanced invariance to rotation [183], or an encoder-decoder attentive model [184]. These more sophisticated methods however come with a higher computational footprint.

In the following we review two of the most promising solution in terms of robustness and low computational overhead which are suitable for real-time applications.

2.1 Fully Convolutional Geometric Features (FCGF)

The introduction of Minkowski Engine[1] (ME) [12] opened the door to the possibility of implementing fully convolutional neural networks (FCNN) to address classical problems of 3D vision to apply convolutions to unordered point clouds. One of the first methods to benefit from such a technological leap was the Fully Convolutional Geometric Feature (FCGF) descriptor [15]. In Fig. 3.8 we depict its architecture, which is quite straightforward: a U-Net structure takes as input a point cloud having size $N \times 3$, which is passed throughout a chain of 3D sparse convolutional layers aimed to encode the low dimension input into a higher dimensional space. Such a high dimensional description is then decoded by means of transposed convolutional layers [185] and the final output is a tensor of size $N \times M$, where M is the dimension of the descriptor associated to each point. ResNet blocks are in place, interleaved with standard CNNs layers to increase the depth of the network, while skip connections are used to pass the first layers information content to the decoding stage. In total we have 19 layers. The highest dimensional space the network reaches is 256D while in the end the descriptor has length 32, which is a good value to ensuring compactness.

Two learning metric are proposed to train the network, namely the con-

[1] `https://github.com/NVIDIA/MinkowskiEngine`

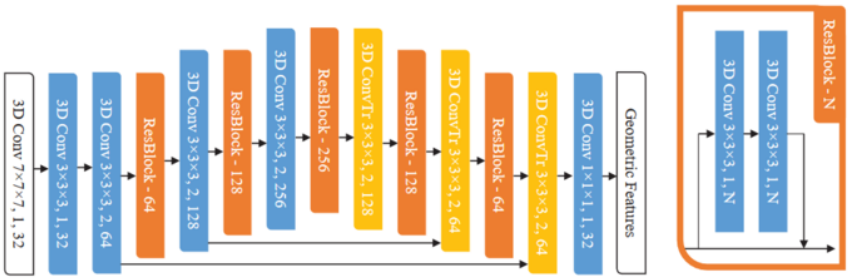


Figure 3.8: FCGF architecture. The kernel size, the stride and the output channel dimension are reported in each layer. Image taken from [15].

trastive loss and the triplet loss. The rationale is that two features (f_i, f_j) should be close to each other when they are similar (*i.e.* they belong to a set of positive correspondences), whereas different features should differ at least by a certain margin (negative correspondences). The contrastive loss can then be defined as:

$$L(f_i, f_j) = I_p \left[d_{i,j} - m_p\right]^2 + I_n \left[m_n - d_{i,j}\right]^2 \tag{3.3}$$

Here $I_p[\cdot]$ returns 1 when the two features are similar and 0 conversely, while $I_n[\cdot]$ does the opposite. Two margins m_p and m_n are used to control the maximum and minimum distances for positive and negative pairs respectively and $d_{i,j}$ represent the distance between f_i and f_j in euclidean space. The triplet loss instead consider one feature f_i and a positive (f_p) and a negative (f_n) pair of features simultaneously:

$$L(f_i, f_p, f_n) = (m + d_{i,p} - d_{i,n})^2 \tag{3.4}$$

In practice, at runtime the pairs of features are picked randomly from positive and negative correspondence sets. In FCGF a *hardest* variant is proposed which impose the selection of the hardest negative sample [186] (Fig. 3.9). The experiments showed that the triplet loss is prune to collapse, so the contrastive loss is preferred in the end.

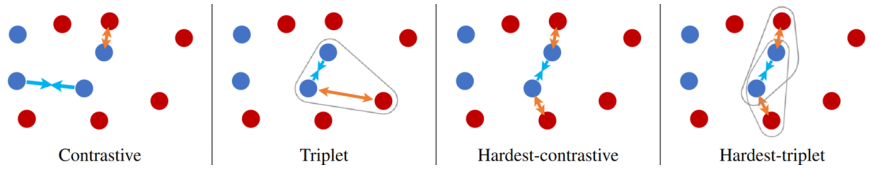


Figure 3.9: Sampling and negative-mining strategy for different losses. Traditional contrastive and triplet losses use random sampling. Conversely, the hardest-contrastive and hardest-triplet losses use the hardest negatives. Image taken from [15].

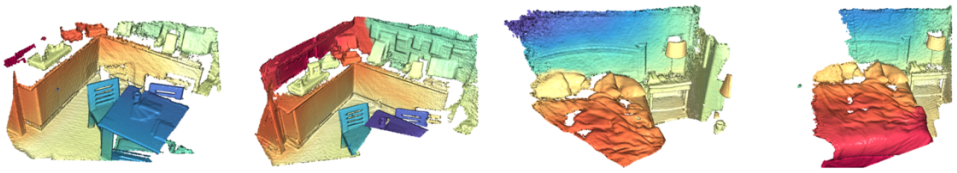
Because of the fully convolutional architecture, the network mitigates the curse of dimensionality, so that no heavy quantization is required for the input (just for the sake of discretization and relaxation on neighborhood density variation) whereas the model can be fed with a dense point cloud and the local information is preserved. Moreover, running sparse convolutions is much faster than dense counterpart and no additional overhead is required to shift paradigm moving from original point cloud input to regular data structures. In Tab. 3.1 we report the results according to [15] obtained from the evaluation of the feature matching recall (FMR) on 3DMatch dataset [6] with $\tau_1 = 0.05m$ and $\tau_2 = 0.10m$ (see Appendices Sec. C for further details). The timing is reported per single feature extraction, so it is not particularly meaningful per se, however it is helpful to compare against other methods. Overall, FCGF is by far the fastest solution and it also has the best FMR, which means that most the valid matches are detected with it. Indeed, FMR is the most important parameter for ensuring a good registration because it allows to rely on valid correspondences in the final step of alignment matrix estimation (a low precision, conversely, can be tackled with *ad hoc* outliers rejection solutions [6]).

Method	**3DMatch [6]**		**Augm**		**Feat**	**Time**
	FMR	**STD**	**FMR**	**STD**	**Dim**	**[ms]**
Spin [94]	22.7%	11.4%	22.7%	12.1%	153	0.133
SHOT [90]	23.8%	10.9%	23.4%	9.5%	352	0.279
FPFH [98]	35.9%	13.4%	36.4%	13.6%	33	0.032
USC [93]	40.0%	12.5%	-	-	1980	3.712
PointNet [9]	47.1%	12.7%	-	-	256	0.171
CGF [177]	58.2%	14.2%	58.5%	14.0%	**32**	1.463
3DMatch [6]	59.6%	8.8%	1.1%	1.2%	512	3.210
Folding [181]	61.3%	8.7%	2.3%	1.0%	512	0.352
PPFNet [178]	62.3%	10.8%	0.3%	0.5%	64	2.257
PPF-FoldNet [180]	71.8%	10.5%	73.1%	10.4%	512	0.794
CapsuleNet [182]	80.7%	6.2%	80.7%	6.2%	512	1.208
3DSmoothNet [176]	94.7%	2.7%	94.9%	2.4%	**32**	5.515
FCGF [15]	**95.2%**	2.9%	95.3%	3.3%	**32**	0.009

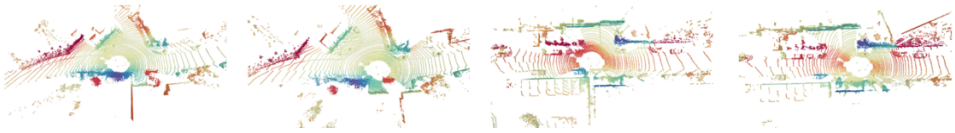
Table 3.1: Feature match recall (FMR) and its standard deviation (STD) on 3DMatch test set. Feat Dim shows the descriptor dimension of each method. Time refers to milliseconds consumed per feature. The table is reproposed from [15].

The network is trained to work not only with 3DMatch indoor but also with Kitti [56] outdoor scenes. Overall, the resulting geometric features seem robust to partial views and clutter as well as to scale size. In Fig. 3.10 a color-coded representation of these features using a t-SNE [187] scalar mapping

is represented. This means that similar features in FCGF 32-dimensional space also have similar color here. It can be appreciated how the different regions of the fragments show effective distinction.



(a) 3DMatch dataset [6]. (Left) `Kitchen`. (Right) `Hotel 1`.



(b) Kitti dataset [56]. (Left) `Pair 1`. (Right) `Pair 2`.

Figure 3.10: Color-coded features on fragments pairs for scenes in (a) indoor and (b) outdoor benchmark datasets. The FCGF features for each pair of point clouds are mapped to a scalar space using t-SNE [187] and colorized with spectral color map.

2.2 D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features

The work by Bai *et al.* [14] shares many common points with the work of Choy in FCGF. In essence the network has a U-Net structure with skip connections which takes as input a dense point cloud $N \times 3$ and outputs a higher dimensional tensor $N \times M$ (see Fig. 3.11). Differently from FCGF, instead of ME, D3Feat uses a density normalized version of KPConv [11] as a backbone to define the ResNet blocks that compose the layers of the U-Net. The density normalization is necessary to overcome the fact that KPConv is not point density invariant. The equation of the general convolution by kernel g at point x in the set $\boldsymbol{P} \in \mathbb{R}^{N\times 3}$ associated with set of features $\boldsymbol{F}_{in} \in \mathbb{R}^{N\times D_{in}}$ is:

$$(\boldsymbol{F}_{in} \star g) = \frac{1}{|N_x|} \sum_{x_i \in N_x} g(x_i - x) f_i \tag{3.5}$$

where N_x is the radius neighborhood of point x and the kernel function is defined as:

$$g(x_i - x) = \sum_{k}^{K} h(x_i - x, \hat{x}_k) \boldsymbol{W}_k$$

Here h is the correlation function between the kernel point $\hat{x}_k$ and supporting point x_i, K is the number of kernel points and $\boldsymbol{W}_k$ is the weight matrix of $\hat{x}_k$.

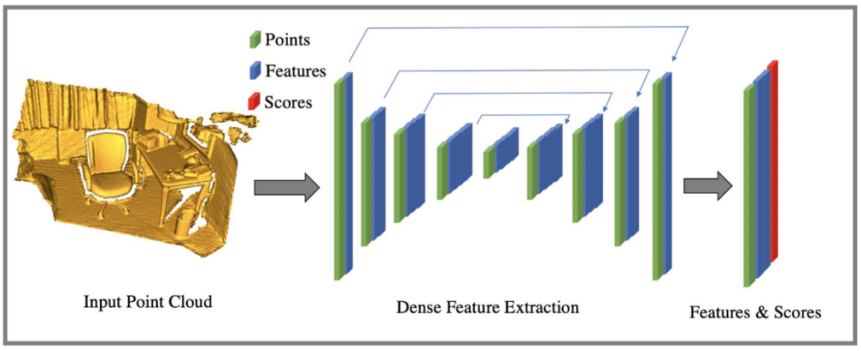


Figure 3.11: D3Feat architecture. Image taken from [14].

Moreover the output of D3Feat has additional information with respect to FCGF. Indeed, besides the descriptor which is again a 32-dim vector, D3Feat outputs also a 1-dim vector to assign a score to each point. Similarly to 3DFeatNet [174] in 3D or R2D2 [172] in 2D, the score represents the distinctiveness of a point, *i.e.* its saliency. It is then possible to sort them out according to the score and find the most reliable points, meaning the keypoints of the cloud. In Fig. 3.12 we report a colorized score map which highlights how the attention of the network prioritized the corners and the sharp edges of the dense point cloud. The method uses a combination of

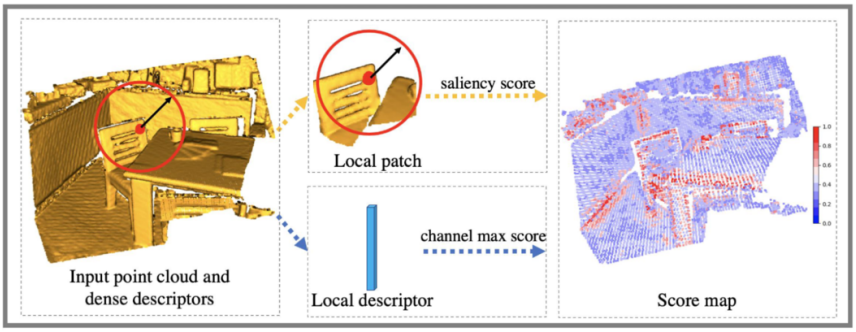


Figure 3.12: D3Feat [14] keypoint detection.

two losses during training. The former is the descriptor loss, which is again the hardest contrastive as for FCGF. In addition there is need to define a detector loss. The idea is then to encourage the network to predict higher scores for matchable correspondences and lower scores for non matchable

ones. The loss is then defined as:

$$L_{det} = \frac{1}{n} \sum_{i} \left[(d_{i,p} - d_{i,n}) \left(s_i + s_p \right) \right]$$

The rationale here is that when $(d_{i,p} - d_{i,n}) < 0$ the correspondence can be matched with nearest neighbor search and the two scores – s_i and a positive match s_p – are encouraged to be high. On the contrary, when the two correspondences are ambiguous, the $d_{i,n}$ is smaller than $d_{i,p}$ and so the scores should be low. According to the authors, D3Feat offers a comparable FMR on 3DMatch dataset and it is even faster to compute than FCGF since KPConv avoids the overhead of hashing to implement sparse convolutions, conversely to what happens with the Minkowski Engine.

3 DL-based frameworks for rigid point cloud registration

In Chapter 1 Sec. 3.1.4 we saw how important is to reject the false positive matches and to retrieve the good correspondences to estimate a coarse alignment based on sparse sub-sets of points. From a data-driven perspective, this problem has been treated as a classification task: the goal is to assign a weight for each putative correspondence in order to represent the likelihood of the match to be real. The first work where we found this idea is [188], where Yi *et al.* address the problem for 2D domain. They propose to use a fully connected network to produce a score for each 2D correspondence. Moreover, the network regresses the final homography which aligns two views of the same outdoor scene. In a similar fashion, Zhang *et al.* present OANet [189], which again regresses the homography transformation to align two images using an ordered-aware network which clusters the unordered input correspondences into a canonical order. The clustering captures the local context, then the clusters are used to extrapolate the global context via spatial correlation. Another interesting solution is SuperGlue [190], a graph neural network which leverages the spatial consistency to find the valid 2D matches in the wild.

Recently, all these methods have been extended to 3D domain. The first work to address the problem of point cloud pairwise rigid 3D registration was 3DRegNet [16] in 2019, which is an end-to-end trainable network based on [188] used to regress the alignment matrix receiving as input a set of 3D corresponences. At the core of 3DRegNet there is a PointNet-like architecture to deal with the unordered set of input correspondences. Other works exploit a similar solution: PointNetLK [165] unrolls PointNet and Lucas and Kanade algorithm [191] into a single trainable recurrent deep neural network. Li *et al.* revisited PointNetLK to include an analytical Jacobian [192] to enhance the generalization. Then, Choy *et al.*, the authors of FCGF,

leveraged again the Minkowski framework for tackling the new task and presented Deep Global Registration network (DGR) [17]. At the same time, the author of 3DSmoothNet, Gojcic *et al.*, introduced their own solution [18] by leveraging on a 3D version of OANet and extended the problem to a multi-view domain. More recently, the authors of D3Feat also proposed PointDSC [142], a network which incorporates the spatial coherence to learn how to register a pair of point clouds.

Moreover, some methods exploit a combination of spatial coordinates and geometric features to guide full matching between points, such as MFG [193], which stands for multi features guidance network, or RMP-Net [194] which additionally uses a differentiable Sinkhorn layer [195] for obtaining soft correspondences. PR-Net [196] uses Gumbel-Softmax with gradient estimation to obtain a sharp and near-differentiable mapping function. In DirectReg [197], Deng *et al.* expolits their PPF-FoldNet [180] for creating oriented-specific poses with a siamese network configuration and then pass the encoded descriptors to a RelativeNet which assigns correspondence-specific orientation and retrieve the alignment pose. Finally Deep Closest Point (DCP) [198] utilizes a sub-network to address difficulties in the classical ICP pipeline, by using DGCNN [199] to extract and merge local features, but it assumes to work with complete point clouds, which is rarely the case.

In the following we review the most appealing solutions that promise to offer robust registration at the expense of a very small computational overhead. Indeed, our hope is to find a solution which is suitable for real-time applications.

3.1 3DRegNet

Pais *et al.* [16] present an extension in 3D of the work presented by Yi *et al.* [188] to estimate homographies in 2D. The main difference here is that the input of 3DRegNet is a set of correspondences of 3D coordinates (*i.e.* a $N \times 6$-dimensional tensor) whereas the input of [188] was made of 2D correspondences (so $N \times 4$ tensor). In Fig. 3.13 we review the architecture of the network. The method comprises two sub-blocks which are devoted to reject the outliers within the initial set and to regress the transformation matrix to align them respectively. The rejection block is indeed a classification network which uses a concatenation of 12 ResNets blocks [200] bounded by a couple of MLP layers to output a N-dimensional vector. Therefore the network maps $\mathbb{R}^{N\times 6} \rightarrow \mathbb{R}^N$ to assign to each correspondence a weight $w \in [0, 1]$ which represents how reliable that correspondence is. Similarly to what it is done in other works [17, 18] such a weight vector could be exploited in a weighted least square algorithm to infer the transformation matrix. However, in the original version of 3DRegNet the authors preferred an end-to-end solution to train the network to infer the proper alignment automatically. The additional regression block aims to do that. While traversing the chain

of ResNets in the classification block, every mini-network returns a temporary $N \times 128$ output that is passed both to the successive ResNet (either as direct input and as skip connection) and to a *context normalization* block [188]. The block normalizes with respect to mean and variance of the distribution and it uses max pooling to retrieve the global information of the inner relationships. At the output we have a $(12 + 1) \times 128$ tensor which feds the regression block defined by 8 convolutional and 2 fully connected layers. The final output is a learned parametrization of the rigid transformation $\hat{\mathbf{T}}$ that aligns the two sets. The $\hat{\mathbf{T}}$ matrix consists of a rotation matrix $\hat{\mathbf{R}}$ and a translation vector $\hat{\mathbf{t}}$. Its size depends upon the desired representation of $\hat{\mathbf{R}}$: in Lie algebra only three parameters are required, four when a quaternion-based representation is used and nine with linear algebra. After the experiments run in the original paper, Lie algebra, is indicated as the best option due to its compactness and the reconstruction perfomance overall.

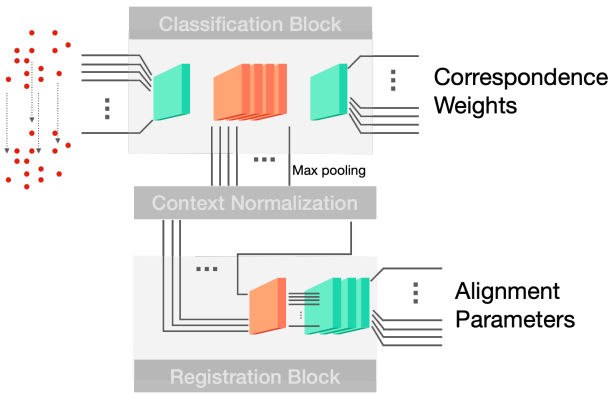


Figure 3.13: 3DRegnet architecture. A set of input correspondences is passed to a classification block which outputs the corresponding weight representing an accuracy score for each match. At the same time, the output of the inner layers within the classification block (specifically a chain of ResNets) is passed to a context normalization layer after a max pooling filtering. The normalized information coming from the classification block serves as input for the registration block, which is composed by a set of convolutional and fully connected layers. The final output is a parametric representation of the transformation matrix which aligns the matching sets.

The learning is fully supervised: a loss for each block is defined, while a ground truth is required to reference the valid correspondences as well as the optimal pairwise registration matrix. The *classification loss* is defined as a cross-entropy function $H(y, \hat{y})$:

$$L_c^k = \frac{1}{N} \sum_{i=1}^{N} \left(\gamma_i^k H\left(l_i^k, \sigma(o_i^k)\right) \right. \tag{3.6}$$

which, for the k-th fragments pair, computes the mean of the cross-entropy over the N correspondences, evaluated for the ground truth label l_i^k and the output of the classification block (o_i^k) activated with a sigmoid function σ. The *registration loss* instead is a simple formulation of the ℓ_1 distance metric function in 3D space. An ℓ_1 distance was experimentally chosen among other metrics because it produced best results in terms of stability and convergence properties. The loss is then defined as the point-wise distance between $\mathbf{x}$ and the transformed version of the corresponding one $\mathbf{y}$ after applying the learned transformation $\hat{\mathbf{T}}$:

$$L_r^k = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{x}_i^k - (\hat{\mathbf{R}}\mathbf{y}_i^k + \hat{\mathbf{t}}) \right)_1 \tag{3.7}$$

At training stage, a batch of correspondence lists of preset length is passed to the network. The final loss objective is a linear combination of the two losses we reported above, evaluated over the mean of K different pairs. Precisely the *total loss* is:

$$L = \alpha L_c + \beta L_r \tag{3.8}$$

where α and β are two hyper-parameters used to properly tune the impact of each sub-loss on the overall cost.

In practice, the FC network requires to have a restricted number of correspondences to evaluate at runtime. The authors of 3DRegNet uses 2000 of such correspondences for their tests.

3.2 Deep Global Registration

Deep Global Registration (DGR) [17] is the proposal from Choy *et al.* to leverage sparse convolutional layers by means of their framework – Minkowski Engine – to reject bad matches in a correspondence set and retrieve the optimal transformation to align two point clouds.

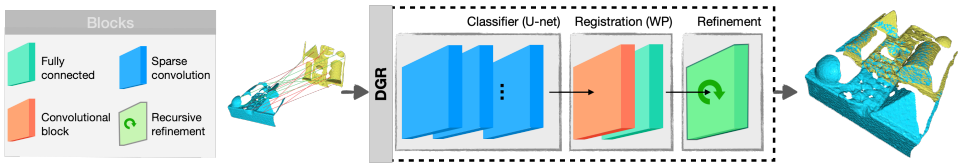


Figure 3.14: DGR architecture. A set of corresponding point is passed as input to a U-shaped network which classifies the correspondences by assigning a confidence weight to each. These weights are used in a Weighted Procrustes-based registration block which infers the registration matrix. A refinement block is in place to further improve the accuracy of the estimation.

In Fig. 3.14 we report the main blocks of DGR. The first component has a similar behavior to the classification block of 3DRegNet, *i.e.* it estimates

the accuracy of a 6-dimensional input set of putative correspondences, but instead of using MLPs to densely connect all correspondences information, it has a 6-dimensional convolutional network aimed to estimate the final weight tensor. The structure of this network is almost identical to the one proposed in FCGF: a U-Net with consecutive ResNet blocks containing a couple of sparse convolutional layers each and skip connections to connect the initial layers with the final ones that in the end returns the N-dimensional tensor. As for 3DRegNet, a cross entropy loss is used for training the model using the ground truth correspondences set as reference.

The second block of this method is then a differentiable Weighted Procrustes solver. In general, the Procrusets method [201, 202] provides a closed-form solution for rigid registration in the special euclidean group $SE(3)$, which is defined as the linear combination of a rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and a translation $\mathbf{t} \in \mathbb{R}^3$. In order to have a end-to-end differentiable registration, the standard approach the method is to pass gradients through coordinates. The energy function to minimize is then the mean squared error between corresponding points:

$$E = \frac{1}{N} \sum_{(i,j)\in\mathcal{M}} \left(\mathbf{x}_i - \mathbf{y}_j \right)^2 \tag{3.9}$$

where N is the number of keypoints and $\mathcal{M}$ is the set of correspondences. However, this solution is a $O(N^2)$ complex problem. On the contrary, weighted Procrustes method of DGR passes gradients through the weights associated with correspondences and it allows to reduce the burden of computation and it exploits a dense correspondence set rather than sparse keypoints. The formulation is:

$$E^2 = \sum_{(i,j)\in\mathcal{M}} \left((\mathbf{y}_j - (\mathbf{R}\mathbf{x}_i + \mathbf{t})) \right)^2 \tag{3.10}$$

In the end, the registration loss adopted for training this module is

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i}^{n} \phi \left(w_{(i,J_j)} \right) L \left(\mathbf{y}_{J_i}, \mathbf{R}\mathbf{x}_i + \mathbf{t} \right) \tag{3.11}$$

where J_i is the j matching coordinate to i in the opposite set of points, ϕ is a prefiltering function to threshold only correspondences with an estimated accuracy above $\tau = 0.5$ and L is the Huber loss [203].

Finally, an optimization module is used to fine-tune the initial alignment and incorporates a failure detection module to evaluate whether the number of valid correspondences returned by the classification module is sufficient for running the Weighted Procrustes solver or not. Later in Sec. 4.4 we will discuss more about the contribution of this module for the pose refinement phase.

3.3 Learning Multi-view Point cloud Registration (LMVA)

The work of Gojcic *et al.*, LMVA [18], is a step forward with respect to the previous solutions we saw. It is again an end-to-end learnable 3D point cloud registration, however it does not limit to pairwise configuration but indeed it extends the approach to a multi-view reconstruction scenario. As the name suggests, multi-view registration considers the contribution from multiple agents to find a tight solution to align several point clouds all together simultaneously. Standard applications usually follows a two-stage pipeline: an initial pairwise alignment is evaluated for all the pairs involved and then a globally consistent refinement is performed. Such a global refinement aims at establishing the cyclic consistency across multiple scans and helps in resolving the ambiguous cases due to the ambiguity of certain pairwise alignment. Indeed, pairwise registration can be affected by the lacking of sufficient overlap between neighboring points, parts symmetries, or repetitive scene areas, and so on.

In practice, the task of global refinement can be approached in several ways. One option is to exploit a brute force correspondence search along with ICP-based alignment to optimize the camera poses. Due to the increased complexity of correspondence estimation, some methods have only the motion as objective for optimization. More recent solutions exploit the strategy of synchronization [204, 205, 206] as backbone for optimization. These methods creates a graph of interconnected poses initially established from pairwise registrations and then use the global cycle-consistency to synchronize all the nodes simultaneously. DeepMapping [207] is one of the first attempt to develop a neural network to tackle the problem. It proposes a global point cloud registration approach by first estimating the pose with a registration network and then it uses a second network for estimating the occupancy status of global coordinates to model the scene structure. Huang *et al.* [206] instead propose to learn a weighting function to use for the edge in the graph during synchronization, similar to what also LMVA does, which in addition also learns the descriptor representation using FCGF and the pose estimation with a revisited version of OANet. In Fig. 3.15 the architecture of LMVA is illustrated.

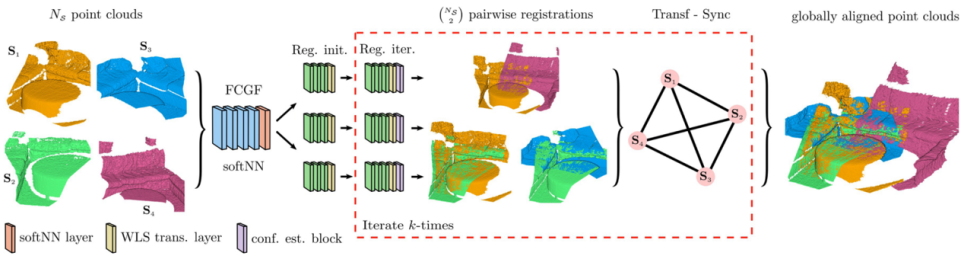


Figure 3.15: LMVA architecture. Image taken from [18].

It is worth mentioning that the method offers two distinct pipelines at implementation level. The first one solve solely the pairwise registration problem, while the second one incorporates all the pairwise relationships into a global framework for multi-view refinement. Up to now, the authors provided only the code for the former solution. In 4.4 we review such implementation with more details.

4 Cross-domain assessment of deep learning-based alignment solutions for real-time 3D reconstruction

After reviewing the state-of-the-art for DL-based registration solution, we make a set of observations. First, the Most prominent DL-based contributions for 3D view alignment appeared very recently and almost concurrently, thus there is still the need for an in-depth comparison among them aimed at analyzing opportunities they offer and possible limitations. Moreover, since these technologies have been proposed and tested on data from low-cost scanners and with medium-large scales (about 1 to 10 meters), there is the need to understand if the same solutions are equally effective when dealing with data acquired with hand-held scanners targeted for quality reconstruction of single objects/subjects rather than entire scenes/environments, and operating on smaller scales and ranges (from cm to few meters). Finally, beyond scale-related aspects, 3D alignment performance can be conditioned by other sensor-related and/or data-dependent aspects (e.g. point density, geometric features, noise, artifacts) and this may require due attention for design, an appropriate data collection and opportune adaptation strategies.

Hereafter, we address the above points both singularly and jointly aiming at a better understanding and exploitation of DL-based technologies in the challenging context of real-time 3D reconstructions. This is done following a cross-domain assessment of the considered DL methods with the objective of verifying/refining their robustness and generalization properties. This work was part of two publication, [C3] and [J1].

4.1 Data

Our goal is to evaluate the performances of novel 3D-DL models to perform a pairwise registration of point clouds created with the scanner which is currently under study. We already introduced the device in the previous chapter (Chapter 2 Sec. 1) and we explained the type of data it produces. In brief, the device targets the reconstruction of small scale objects by ensuring a high level of accuracy. The point clouds we extract from each frame during the scanning, represent a small area using a lot of points such that we refer to them as a dense data. Then, in order to benchmark new solutions, we

created a dataset, we called it DenseMatch and we presented it in Chapter 2 Sec. 3. This is the first dataset we aim to assess here. The second dataset is 3DMatch [6]. Indeed, we already introduced also this dataset so that the reader can refer to Chapter 1 Sec. 2.1 for additional details. Nevertheless, in Tab. 3.2 we summarize both the datasets by giving their main specification for the sake of a quick comparison. Moreover, in Fig. 3.16 we show again few examples extracted from both with an additional visualization of the different scales we are dealing with.

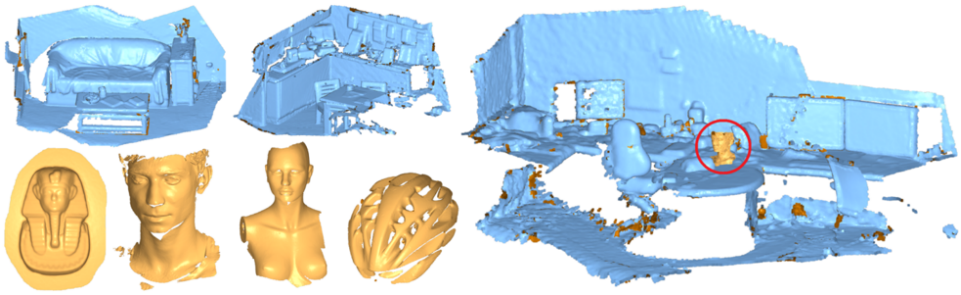


Figure 3.16: Example of scenes for 3DMatch [6] (blue) and DenseMatch [J2] (yellow). On the right two scenes are represented together to highlight the scale difference of the two data.

Dataset name	Num of Scenes	Num of PC	Data type	Scanner used	Scanner type	Camera Resolution	Num Points per Frame (avg)	BBox Volume per Frame (avg)	Point Spacing
3DMatch [6]	64	2601	indoor reconstruction	Kinect, Xtion Pro, Real Sense D 415	All handheld optical scanners	Stereo Cam (640x480)	[11k - 640k] (150k)	[0.35 - 134] (20) m^3	[6.5 - 8.2] (7.10) mm
DenseMatch [J2]	19	3140	object reconstruction	InSight3 (prototype)	Handheld optical scanner	Stereo Cam (1280x1024)	[60k - 1M] (500k)	$[10^{-4}$ - $10^{-2}]$ (3×10^{-3}) m^3	[0.18 - 0.35] (0.25) mm

Table 3.2: 3DMatch and DenseMatch main specifications. BBox stands for Bounding Box. The oriented BBoxes are estimated using Open3d [55] as well as the average points inter-distances that have been used to define the spacing.

It is worth saying that we chose to pick 3DMatch because it was involved in almost all the novel methods we are going to assess, both for training the models and testing them. Since 3DMatch is basically an indoor reconstruction benchmark dataset, our goal is twofold: we want both to double-check the validity of the results each work proposed for it and to discuss about how each method can generalize in a cross-domain application scenario, since the input (*i.e.* pairs of point clouds) is the same.

The idea is to test all potential point cloud pairs (i.e. $PC_i - PC_j$, $i \neq j$) for each dataset. A minimum overlap between the two surfaces is required at this stage to define valid matching pairs. Therefore we follow a common practice and we choose a threshold equal to the 30% of the smallest point cloud in the pair as the minimum overlapping amount of points. In order to determine whether a point has overlap or not, we set a maximum distance

threshold equal to 3 times the voxel size we use for sub-sampling. In addition, we do a further skimming in the DenseMatch dataset: indeed, most of its frames are redundant due to the high rate we used to acquire them which produce a large overlapping area between consecutive fragments. According to the above mentioned criteria the test set would have ∼70k trials (while 3DMatch has 1623 of them). For this reason, we sample decimate the test pool by picking one out of 10 pairs. The final test set contains 7261 valid pairs.

We now present the strategy we used for the assessment.

4.2 Method

Here we focus on recent architectures that start from a set of 3D point correspondences that can be extracted either from traditional approaches (FPFH [98], presented in Chapter 1 Sec. 3.1.2) or even DL-based ones (FCGF [15] and D3Feat [14], presented in Sec. 2.1 and 2.2 respectively). A schematic overview of the selected solutions, showing how they are inserted into the comparative context, is presented in Fig. 3.17, while the principal aspects of each network involved in the process (including feature descriptors) are summarized in Tab. 3.3.

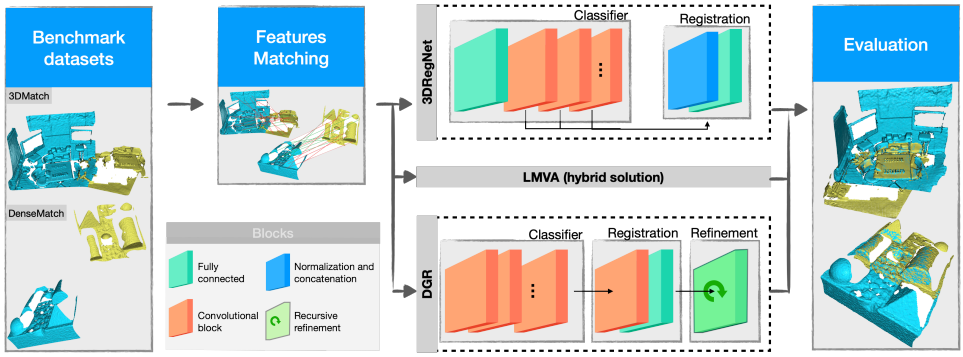


Figure 3.17: Overview of the used testing pipeline and architectural differences of the considered methods. We evaluate two benchmark datasets. The preliminary step requires to down-sample and to describe via geometric descriptors each point cloud. Then, on the descriptor space, we perform the matching of similar points between a couple of point clouds and we produce a set of putative correspondences pairwise. Moreover, we feed the networks with this set. Their output is the estimated transformation matrix that align the source point cloud with its target. Finally, given the ground truth poses of each point cloud in the dataset, we are able to evaluate the quality of the alignments.

Method Name	Input type	Network type	Keypoint detection	Feature extraction	Feature dimension	Pairwise registration	Multi-View registration
D3Feat [14]	Point Cloud ($N \times 3$)	fully convolutional (KPConv [11])	✓	✓	32	-	-
FCGF [15]	Point Cloud ($N \times 3$)	fully convolutional U-shaped (ME [12])	-	✓	32	-	-
3DRegNet [16]	Corresp List ($N \times 6$)	fully connected + pooling + conv (PointNet [9], Yi [188])	-	-	-	✓	-
DGR [17]	Corresp List ($N \times 6$)	fully conv U-shaped (ME [12]) + weighted Procrustes	-	-	-	✓	-
LMVA [18]	Corresp List ($N \times 6$)	fully connected (OANet [189]) + weighted Procrustes	-	-	-	✓	✓

Table 3.3: Summary of the considered DL methods. For each one, the type of required input data, the network structure and purposes are reported.

4.2.1 Creation of the putative correspondences set

Before starting the test, we point out that the networks are agnostic to the method used to create the putative correspondences. In the original articles, 3DRegNet uses FPFH to evaluate correspondences, while LMVA chooses FCGF. DGR uses both: it picks FCGF for testing its own method while it uses FPFH for the handcrafted estimators. In order to maintain fairness in the way we formulate the tests, we decide to use a unique descriptor for all the methods instead. Moreover, since we are interested into investigating systems largely based on DL solutions, motivated by the promising results presented in the original works, we decide to leverage a data-driven approach also for feature extraction. Nevertheless, in the ablation study section 4.4 we will also consider FPFH as an alternative solution and we will see how it performs in comparison. Overall, we follow the DGR policy to build the set of putative correspondences: first, each point cloud is quantized according to a voxel size; secondly, a 32-dimensional FCGF descriptor is assigned to each point. It is worth to notice that we should not use the same parameters for both 3DMatch and DenseMatch, since the default voxel-size was 5 cm for the former. By using such value we would definitely lose the geometric context in the latter case. Moreover, after we extracted the features we match them as follows: each feature in the source point cloud is paired with the closest one in the feature space of the target point cloud. In practice, if we extracted 1000 and 2000 features from the source PC and the target respectively, we would end up with 1000 matches nevertheless. This means that some target points can be matched multiple times with different source points. Indeed we do not impose mutuality to speed up the matching process. Finally, we save the putative matches and we use them to feed all the registration methods without requiring to recompute them at runtime, so that we save time during training.

4.2.2 Evaluation metrics

The metrics used to analyze the quality of the registration methods are in common with other works [6, 16, 17, 18]. In particular, we focus on the accuracy of the estimated pairwise transformations by $\hat{\mathbf{T}} \in \mathbb{R}^{4\times4}$. In order to that, we first split the matrix into its rotational and translational components, *i.e.* $\hat{\mathbf{R}} \in \mathbb{R}^{3\times3}$ and $\hat{\mathbf{t}} \in \mathbb{R}^3$ respectively. Then we define the rotation errors (RE) as:

$$\mathrm{RE} = \cos^{-1} \quad \frac{\mathrm{trace}(\hat{\mathbf{R}}^{-1}\mathbf{R}_{gt}) - 1}{2} \Bigg) \Bigg(\tag{3.12}$$

where $\mathbf{R}_{gt}$ is the ground truth rotation. The main idea behind such definition is to represent the two rotation matrices in their axis-angle form (as a mapping from Lie group $SO(3)$ to Lie algebra $\mathfrak{so}(3)$) and then to compute the angle between the two vectors. Therefore the RE value is expressed in degrees (deg). For further details the reader can refer to the book of Yi Ma *et al.* [208]. The translation error (TE) is instead defined as the distance in norm ℓ^2 between the two translation vectors, expressed in mm:

$$\mathrm{TE} = \quad \mathbf{t}_{gt} - \hat{\mathbf{t}} \tag{3.13}$$

Finally, following Choy's example [17], we choose to use two thresholds on these values to mark a test as positive when it satisfies the conditions $\mathrm{RE} < \mathrm{RE}_{thr}$ and $\mathrm{TE} < \mathrm{TE}_{thr}$. We compute the recall as the percentage of positive tests out of the total number of tests carried out. For all tests, TE and RE reported values are evaluated exclusively on the pool of results with a positive outcome.

4.3 Experimental setup

We start our test on 3DMatch dataset by using the default implementations as well as the models provided by the authors off-the-shelf. DGR and LMVA models are obtained after training their architectures using the 3DMatch training set [6], which contains 56 indoor scenes. The dataset also provides 8 testing scenes. On the contrary, 3DRegNet, used a smaller training dataset. To cope with this discrepancy, we train the network from scratch using the full training set instead. Unfortunately, as mentioned also by Choy in [17], we have been experiencing problems with the last available 3DRegNet network implementation. In particular, even though we are able to reproduce the original results using its own test dataset, we obtain very bad results on the full test dataset we evaluate also for DGR and LMVA. Such bad results are independent from the type of model we use, either it is the original one or the one we trained from scratch. In the results section, we report more details about the test we performed on this network and our surmise about such faults.

Finally, although LMVA is designed for multi-view registration, so far the authors provided the code for pairwise registration only. Nevertheless, our main focus, for the moment, is related to pairwise rigid registration approaches evaluation, therefore this is not a limitation for us. In order to investigate how the different networks are able to generalize, we then repeat the same tests with the same models using the new DenseMatch dataset. After the first round of experiments with default setups, we try to improve the obtained results. To this aim, we move in two directions: we first took a deeper look at the network setups and we tweak their parameters to better understand their footprint on the performances. Secondly, we refine the pre-trained model: although the networks show already a good level of generalization, we assess the refinement by creating a training subset based on DenseMatch. With this, we want to investigate if the robustness of the models can be boosted by showing to the networks a new paradigm of data, which differs from the original set in terms of type of scene, reconstruction target and point density. To make the comparison as complete as possible, we try to apply the refinement stage to all the models under discussion. Apart from the already mentioned issues with the 3DRegNet implementation, we succeeded at refining FCGF and LMVA, while we did experience some issues with DGR (details are reported in Section 4.4).

Hardware and software information All tests were performed on a Linux Ubuntu 18.04 machine with a Titan V video card, AMD Ryzen 1950x processor, and 64 GB of RAM (DGR and LMVA use the MinkowskiEngine library [12] which currently does not support Windows machines). For 3DRegNet the framework used is TensorFlow v1.14 while for DGR and LMVA we used PyTorch v1.5. For all the handcrafted algorithms instead, we used the implementations developed within the open-source library Open3d v0.10 [55].

4.4 Results

4.4.1 Methods comparison using 3DMatch dataset

We compare the DL methods introduced in the Sec. 2 and 3 with the addition of classic baseline registration methods alternatives, namely RANSAC [114] and Fast Global Registration (FGR) [129]. Open3D offers two implementations for RANSAC: the first one is based on correspondences matching (we call it RANSAC C), which means that it works with the set of correspondences we created, similarly to the other methods. By default, the number of points the algorithm considers simultaneously to infer a single transformation is set to 6. The other option is to use the RANSAC based on feature matching (therefore RANSAC F). In this case the function requires to compute the correspondences at runtime. Actually there is no difference with the

strategy we chose to create such a set, however the default parameter for the number of points to use is set to 4 in this case. Since this is the method that Choy uses in DGR evaluation, we report the results also for this method. Indeed also the implementation of FGR requires additional correspondence creation at runtime, so we need to consider this aspect when we evaluate the timings for this solution. For the classic methods, we performed some tests on the fundamental parameter of the number of iterations but we did not find significant performance changes, as also confirmed by the authors of DGR. Moreover, in regards to DGR, we also assess the performance of the optimization block located at the end of the alignment chain. This test is done with the goal of evaluating the quality of the estimator model based on a Weighted Procrustes algorithm which could be sufficient in order to limit ourselves to an initial coarse recording.

Finally, we remark that that other well-known methods, such as ICP and its recent DL-based counterpart, Deep Closest Point [198], have not been included, since both are based on assumptions not considered here, such as an initial favorable alignment for ICP (the models are instead significantly misaligned) or the matching between complete 3D models (i.e. with total overlap) for DCP [198], which is in contrast to partial overlap conditions in our setting.

Tab. 3.4 (upper part) shows the global results related to all tests on 3DMatch, along with the average alignment times (in *ms*) for the various techniques. We have chosen to keep the thresholds already selected in other methods (*e.g.* DGR): a TE_{thr} value equal to 6 times the voxel size (thus 30 cm) and 15 deg for RE_{thr}. These values are not cherry-picking since, as it can be seen, the mean error and std dev. remain far enough from these upper limits.

A qualitative example of the reconstructions is provided in Fig. 3.18. Furthermore, we expand the results obtained with the different methods by reporting in Fig. 3.19 the recall obtained for each of the 8 test scenes of 3DMatch. It can be seen how results are pretty much consistent across the sets, with some slight decrease of performance in `Home 2` and `Study`. We will deepen these results in the ablation study section 4.4. Overall, we see that DGR and LMVA perform on par with RANSAC. It is interesting to notice how RANSAC F scored a higher recall in our test with respect to the one presented in the paper of DGR [17]. Since we used the same Open3d implementation, we assume we picked better settings on this function. RANSAC C instead was not considered originally and it seems to be even better on average. Nevertheless, LMVA is the method with the smallest mean errors and standard deviations, and it typically succeeded where the other solutions struggled the most. DGR (in its version with no optimization module enabled) is close to this result and it is also 8 times faster than LMVA and 20 times faster than the fastest handcrafted method.

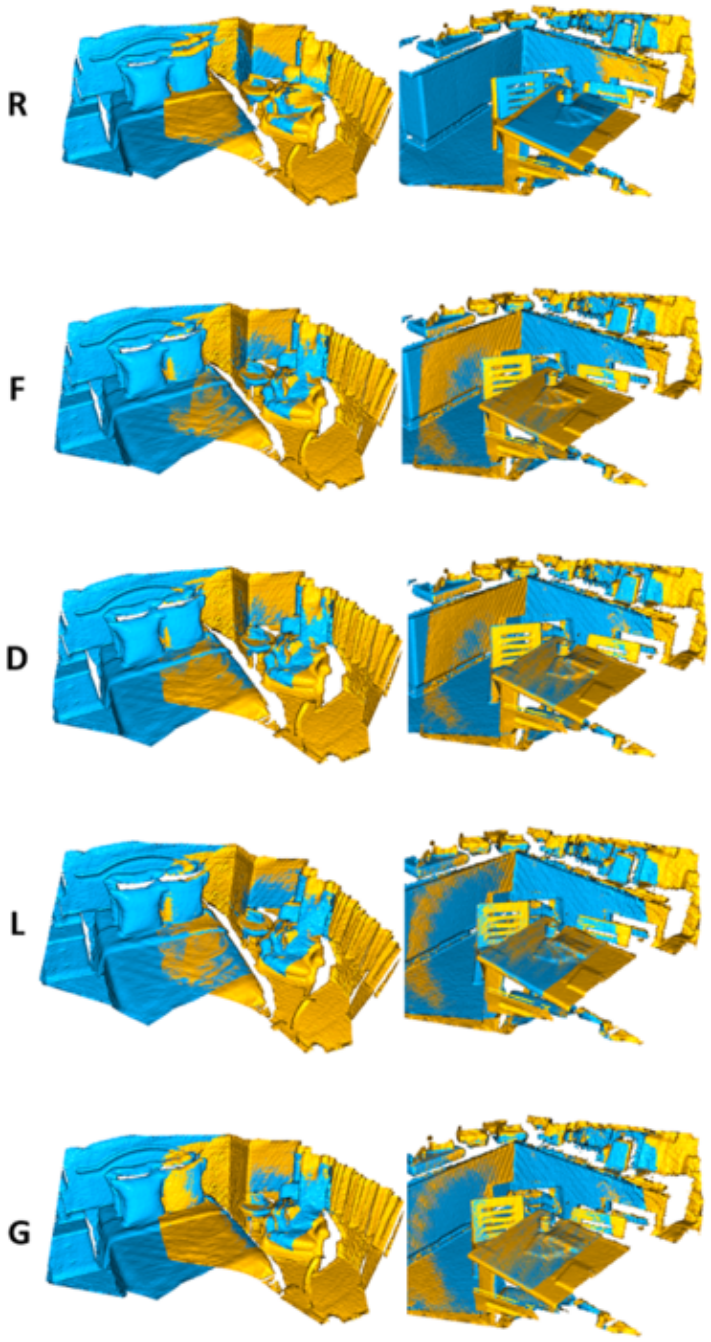


Figure 3.18: Pairwise alignment examples on 3DMatch [6] dataset using **R**ANSAC F [114], **F**GR [129], **D**GR (no optim) [17], **L**MVA [18] and providing the **G**round Truth.

3DMatch dataset [6] - voxel size: 50 mm						
Method	**Recall**	**TE [mm]**		**RE [deg]**		**Time [ms]**
		Mean	**Std**	**Mean**	**Std**	
RANSAC [114] C	85.58%	97.5	64.1	3.29	2.32	41.6
RANSAC [114] F	85.99%	106.6	54.7	3.46	1.93	327.9
FGR [129]	79.42%	85.0	66.7	2.89	2.42	547.0
DGR [17] wo optim	81.52%	77.4	58.2	2.56	2.12	**2.8**
DGR [17] w/ optim	86.57%	73.6	55.6	2.34	1.88	440.8
LMVA [18]	**88.90%**	**71.1**	52.7	**2.32**	1.83	18.4
3DRegNet [16]	6.16%	210.3	61.8	8.77	3.29	11.6
DenseMatch dataset [J2] - voxel size: 5 mm						
Method	**Recall**	**TE [mm]**		**RE [deg]**		**Time [ms]**
		Mean	**Std**	**Mean**	**Std**	
RANSAC [114] C	**58.88%**	13.04	7.52	2.64	1.67	19.4
RANSAC [114] F	56.98%	14.41	7.71	3.04	1.83	161.9
FGR [129]	56.75%	10.83	7.45	2.16	1.65	109.8
DGR [17] wo optim	54.54%	10.26	7.27	1.93	1.44	**2.1**
DGR [17] w/ optim	56.51	**10.24**	7.31	**1.91**	1.45	1674.4
LMVA [18]	53.60%	10.88	7.31	2.18	1.58	15.9
3DRegNet [16]	failed	-	-	-	-	11.1

Table 3.4: Results for the 3DMatch dataset and DenseMatch dataset.

4.4.2 Direct extension of the methods comparison on a denser dataset for object reconstruction (DenseMatch)

We are interested now in direct extension of the assessment of the same methods tested on 3DMatch on the new DenseMatch dataset at first, without changing or tuning the DL models. The one thing that is opportune doing before running the experiments on DenseMatch is to define a suitable voxel size for the initial PC quantization. This value is crucial because it defines both the quality of the FCGF descriptors and the number of correspondences to be treated. Thus, we must find a balance between spatial resolution (the larger the voxel and the coarser the data) and computational demand (the smaller the voxel the greater the complexity of the model, with consequent timings degradation). We used DGR and LMVA as a baseline for these tests and report the results in Tab. 3.5. As can be seen, in the case of a very small voxel size, i.e. at the level of the data resolution (1 mm), feature extraction time tends to rise significantly (almost half a second, making a real-time approach unfeasible), while the timing for recording is still satisfactory even if the performances are slightly worsened because the inlier ratio is probably lower (many more points, many more false positives). Conversely, with a voxel size of 1 cm, we have the best timings but the alignment performance collapses because we have too few points to work with, producing too coarse

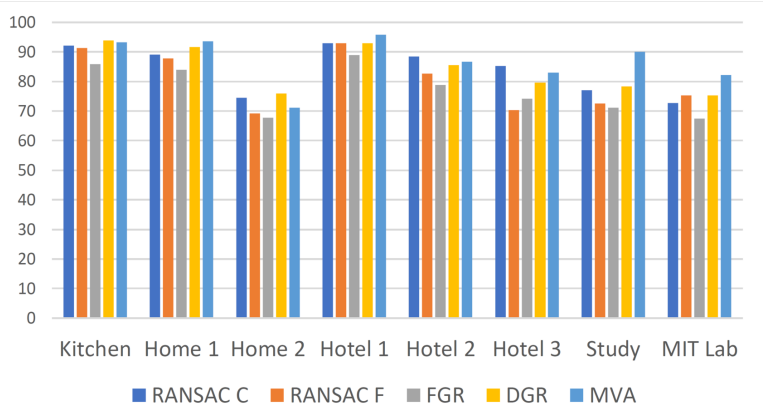


Figure 3.19: Scene wise recalls for all the methods tested on 3DMatch dataset.

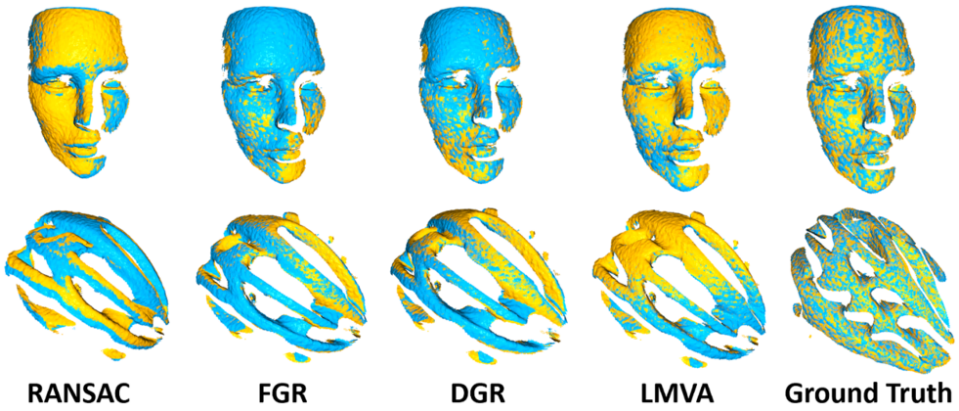


Figure 3.20: Pairwise alignment examples on DenseMatch [J2] dataset using RANSAC F [114], FGR [129], DGR (no optim) [17], LMVA [18] and providing the Ground Truth.

results. With the intermediate value of 5 mm, we obtain the best alignment performance results with very little computational penalty. As for 3DMatch, we set TE_{thr} equal to 6 times the size of the voxel (i.e. 30 mm), while RE_{thr} is set to 10 degrees, aiming at being a little more stringent in an ICP refinement perspective. The results are shown in Tab. 3.4 (lower part), while a qualitative example of the alignment results is given in Fig. 3.20. Again, RANSAC proved to be a valid solution, especially considering RANSAC C. However, its inherent randomness and iterative approach is critical for its real-time application. Nevertheless, DGR (without optimization) is still 9 times faster with a final average error and standard deviation that is smaller both for rotation and translation. Moreover, because of the voxel quantization performed in preprocessing, the average number of points per point

		DGR [17]		LMVA [18]	
Voxel	**Feat Extr. T**	**Recall**	**Registr. T**	**Recall**	**Registr. T**
1 mm	457.60 ms	49.46%	7.68 ms	48.15%	41.50 ms
5 mm	61.20 ms	**54.54%**	2.07 ms	**53.60%**	15.85 ms
10 mm	**46.40 ms**	23.01%	**1.81 ms**	25.40%	**8.40 ms**

Table 3.5: Study on voxel size for the DL methods.

cloud drops to less than 5k. With a slightly smaller voxel size (3mm instead of 5mm) this number increases nearly by a factor of 4 and we experimented that the average running time by RANSAC C goes up to above 100 ms, while neural network-based solutions increase their memory consumption but they keep almost the same timings. That being said, we think that the considered DL-based solutions indeed represent a very interesting alternative to classic reference ones, especially considering runtime speed.

In Fig. 3.21 we report the recall obtained on the test scenes for DenseMatch (DGR refers to the solution without optimizer). As we have already discussed, the recall is lower on average for such a dataset. We attach this fact mainly to three aspects: 1) in these tests the DL-based methods were trained on 3DMatch training dataset, 2) each scene reconstruction in 3DMatch is typically easier to handle due to the presence of highly varied geometry, 3) we set more rigid error thresholds for the DenseMatch dataset. In this case, there is a larger variation between each test scene and we see that all the methods struggled significantly with some particular objects, as the `Helmet`, or human body reconstruction, as `Body 3`. Additional tests and observations are given in the following section.

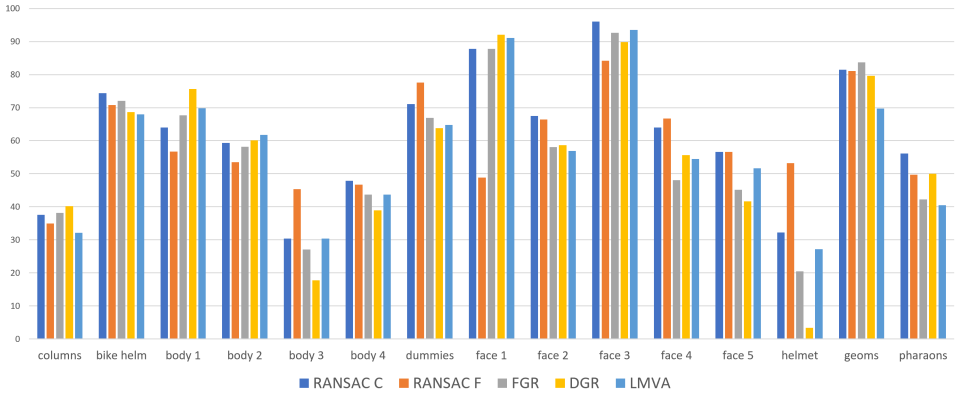


Figure 3.21: Scene wise recalls for all the methods tested on the DenseMatch dataset.

4.4.3 Main issues and failures

We here extend the analysis on the use of DL-based architectures. In the first two paragraphs, we investigate the criticalities that occurred, with a focus on 3DRegNet in the former and the evidence of failure cases for the two considered datasets in the latter. Then, we experiment variations and adaptations of the key elements of the original solutions. First, we evaluate how different methods for feature description perform against FCGF. Then in the next two paragraphs we revise the settings of the two registration networks, DGR and LMVA. We discuss how different configurations can alter each method and how these changes fit in the context of an online reconstruction scenario with our type of data. Finally, we fine-tune the models using DenseMatch aiming at improving the quality of the final registration. This will enable us to draw some conclusions about cross-domain generalization aptitudes.

Analysis of 3DRegNet issues As anticipated, we were not able to achieve satisfactory results with 3DRegNet. We found other authors, such as Choy in [17] to have similar issues. In practice, the path we followed for both datasets was initially identical: first, we tested the datasets against the original implementation provided by the authors together with the trained model that they used to present their results. It's interesting to notice that the authors used the Sun3D dataset [60] for training and testing (together with synthetic data), which is actually a sub-part of 3DMatch. Nevertheless, the results with such configuration failed the test. Then we decided to train again the network using the training data from 3DMatch as described in the test design section 4.2. In this phase, we tried all the available configurations for the Rotation Matrix parametrization, all the reconstruction losses proposed, we performed a grid search on the learning rate and we even extended the maximum rotation for data augmentation, from 50 to 180 degrees. However, none of these trials gave us something on par with the other solutions. Furthermore, we tried a different reconstruction loss, which focuses more on the quality of the rotation matrix inference (expressed with quaternions):

$$L_q^k = \quad \mathbf{q}_{gt} - \frac{\hat{\mathbf{q}}}{\|\hat{\mathbf{q}}\|}$$

With such a solution, we doubled the previous recall, by getting a final 12.93% overall on 3DMatch. However, this is still not enough. Finally, we also investigated the test dataset used by the authors in their original work. In Tab. 3.6 we report the comparison between the Sun3D dataset, used in 3DRegNet as test data, with respect to the 3DMatch and DenseMatch test datasets. As we can see, not only the average inlier ratio is doubled in the test set for 3DRegNet (on average, two-thirds of the correspondences are valid), but also the average transformation to infer is much less challenging. Based on all these facts, we assume that the actual version of the 3DRegNet

architecture, despite being interesting and inspiring, is still below its competitors in terms of the ability to perform well with generic and possibly heavily misaligned data.

Dataset	Inlier Ratio	RE [deg]	TE [m]
Sun3D - 3DRegnet	68.64%	5.55	0.10
3DMatch	35.11%	36.07	0.99
DenseMatch	37.78%	107.13	0.50

Table 3.6: 3DRegNet results on different datasets: Sun3D [60] originally used in [16], 3DMatch [6] and our DenseMatch. Values of Inlier Ratio, Rotation Error and Translation Error are averaged for each dataset. For DenseMatch the average rotation error is higher due to the random 3D rotations we applied from 0 to 2π.

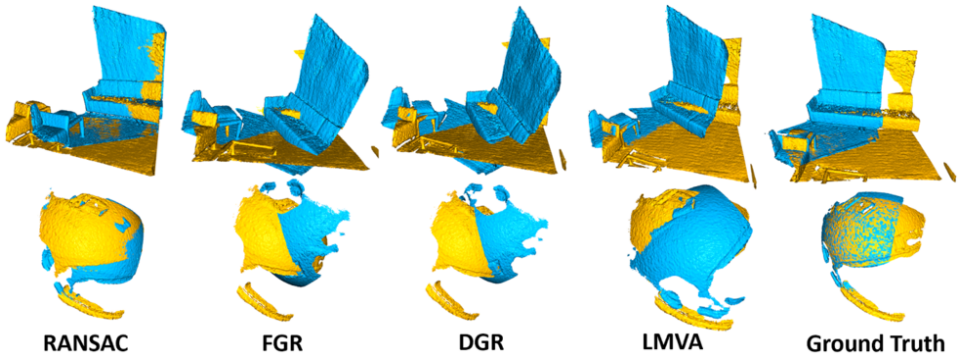


Figure 3.22: Failure examples pairwise on 3DMatch [6] (top) and DenseMatch [J2] (bottom) dataset using RANSAC F [114], FGR [129], DGR (no optim) [17], LMVA [18] and providing the Ground Truth.

Failure Cases We previously saw that not all the test scenes proposed the same difficulty level. Indeed, both in 3DMatch and in DenseMatch cases all the methods under test struggled with some scenes, especially for DenseMatch. In Fig.3.22 we show an example for each dataset. In both, we see that the two fragments under test present large areas without texture and a lot of redundancy in geometric features. The difficulty the matching block has encountered is corroborated by the values of the inlier ratio in both, which are 11% for 3DMatch and 9% for DenseMatch, whilst on average they are close to 35%. Therefore, the features were less informative than usual for these cases. Under these circumstances, we failed to register whatever the selected method. We think that combining an improved saliency-based system to detect the key points as D3Feat with a highly descriptive method as FCGF, with the addition of 2D textures, could help to improve the performance even in these complex scenarios.

3DMatch dataset [6] - voxel size: 50 mm						
Method	**Recall**	**TE [mm]**		**RE [deg]**		**Time [ms]**
		Mean	**Std**	**Mean**	**Std**	
RANSAC [114] F	73.75%	113.51	61.72	3.81	2.34	306.7
FGR [129]	37.40%	112.03	72.61	4.37	3.22	148.4
DGR [17] wo optim	39.75%	77.09	57.68	2.58	2.28	**2.6**
DGR [17] w/ optim	40.30%	78.24	60.33	**2.51**	2.12	1037.7
LMVA [18]	**75.00%**	**74.87**	63.89	2.67	1.72	21.0
DenseMatch dataset [J2] - voxel size: 5 mm						
Method	**Recall**	**TE [mm]**		**RE [deg]**		**Time [ms]**
		Mean	**Std**	**Mean**	**Std**	
RANSAC [114] F	56.29%	14.57	7.58	3.11	1.87	240.3
FGR [129]	42.77%	11.04	7.68	2.28	1.74	37.6
DGR [17] wo optim	20.68%	**8.49**	6.55	1.69	1.35	**2.0**
DGR [17] w/ optim	20.74%	8.50	6.60	**1.68**	1.36	1212.6
LMVA [18]	23.50%	10.93	8.75	2.26	1.81	15.2

Table 3.7: Results for the 3DMatch dataset and DenseMatch dataset using FPFH [98] as feature descriptor.

4.4.4 Alternatives to FCGF

Alternative to FCGF: handcrafted features FPFH Feature description is clearly crucial to create a good set of putative correspondences. Since we wanted to investigate how the novel DL solutions can contribute to our case study, we initially concentrated on a data-driven solution, *i.e.* FCGF [15], which we chose because of its promising results. To confirm this choice, we extend our analysis by also considering one of the most common handcrafted descriptors, FPFH [98]. According to [87], this type of descriptor is well suited for 3D modeling, it is highly descriptive and it has higher performances with high-resolution datasets. Moreover, it is the handcrafted descriptor used in [17] and [16]. We leverage the FPFH implementation available in Open3D [55] and we report the results in Tab. 3.7. If we compare them with the original set of results obtained with FCGF in Tab. 3.4 we can see that the data-driven solution is indeed significantly preferable. In fact, consistent with what observed in [15], the DL descriptor provides better initial guesses for the transformation estimators allowing an improvement in terms of registration recall. On the sidelines of this, in Tab. 3.7 we see how RANSAC F and FGR perform similarly to the results presented in [17] for 3DMatch. This is due to the fact that also there FPFH was used as a feature descriptor for these methods.
Now we have a full comparison for the considered methods with both descriptors, and we derive that FCGF remains a preferable solution also for DenseMatch, even though the model was not trained on this type of data.

	3DMatch		DenseMatch	
Method	**Recall**	**Time**	**Recall**	**Time**
FCGF [15]	**83.00%**	89 ms	**69.11%**	60 ms
3DFeat [14]	69.87%	85 ms	46.61%	15 ms

Table 3.8: Comparison between two DL methods for learning pointwise descriptors. Recall refers to successful registration after running RANSAC with the learned descriptors.

These results prove the quality of this DL-based descriptor, which allows for better registration performance with similar computational efficiency. In fact, both descriptor extractions took around 60 ms on DenseMatch with a voxel size of 5 mm.

Alternative to FCGF: DL-based features D3Feat We also wondered whether there is a valid alternative to FCGF among deep learning solutions. Here we focus on D3Feat [14], which exploits fully convolutional layers too but it uses a variant of KPConv [11] instead of sparse tensors. This approach is interesting because it also addresses the problem of key-point selection. The comparison we made leverages the pre-computed weights provided by the authors. Similar to FCGF, this model is trained on the 3DMatch dataset. We exploit the full test set for 3DMatch and a subset for DenseMatch. After computing the descriptors (both types are vectors of length 32), we pass them through the Open3D RANSAC implementation for a fair comparison. As shown in Tab. 3.8, 3DFeat proves to be faster than FCGF at computing descriptors (as already pointed out in the original article), however, FCGF seems to be better at giving informative content for geometry context. In fact, the latter outperforms 3DFeat at the registration stage, with a remarkably higher recall on both datasets. We point out that we tried to refine the training on 3DFeat using the remaining scenes in DenseMatch, but without success. It seems that a deeper study on this specific network would be needed to possibly improve its performance, especially on data like ours.

4.4.5 Additional configurations

Additional Configurations for DGR Previously, we discussed the optimization module adopted by DGR [17] at the end of the registration chain and we presented the results we obtained either by using this block (with default settings) or by removing it. Now we deepen the analysis on that, by tuning the parameters that determine the convergence of the optimizer. There are four parameters involved: the *maximum number of iterations*, the *loss minimum* value, the *breaking threshold ratio*, and the *max break counter*. The optimization is then performed by minimizing the distance between the points of the (transformed) source and the target point clouds. The opti-

Algorithm 1: Loss Optimizer in DGR [17]

Result: $\hat{\mathbf{T}}$
$(\mathbf{x}_1, \mathbf{x}_2)$ source and target points;
Weighted Procrustes $\rightarrow \mathbf{T}_{init}$;
$\mathbf{x}_1^{trx} = \mathbf{T}_{init}(\mathbf{x}_1)$;
$L_{init} = loss\,func(\mathbf{x}_1^{trx}, \mathbf{x}_2)$;
$iter = 0$;
$break\_counter = 0$;
$\hat{\mathbf{T}} = \mathbf{T}_{init}$;
$L = L_{init}$;
while $iter < max\_iter$ **do**
  $\mathbf{x}_1^{trx} = \mathbf{T}(\mathbf{x}_1)$;
  $L_{new} = loss\,func(\mathbf{x}_1^{trx}, \mathbf{x}_2)$;
  if $L_{new} < L_{min}$ **then**
    break;
  else
    update $\hat{\mathbf{T}} \leftarrow \mathrm{SGD}(L_{new})$;
  end
  if $|L - L_{new}| < L * break\_thresh\_ratio$ **then**
    $break\_counter++$;
    if $break\_counter > max\_break\_counter$ **then**
      break;
    end
  end
  $L = L_{new}$;
end

mization is performed via Stochastic Gradient Descent (SGD) with Adam.

For the sake of clarity, we report the pseudo-code in Algorithm 1 while, for more details please refer to [17]. The default parameters provided by the authors, presumably tuned according to 3DMatch data, are: $max\_iter = 1000$, $L_{min} = 10^{-7}$; $break\_thresh\_ratio = 10^{-4}$; $max\_break\_counter = 20$. The optimizer uses $learning\_rate = 10^{-1}$. We tried different configurations on DenseMatch dataset, especially by working on the max_iter and $break\_thresh\_ratio$ because of their predominant impact as convergence criteria on the whole algorithm. Tab. 3.9 presents the results of these tests. At the top of the table, we see that the default learning rate provides the best recall on an extended set of iterations. However, such a configuration is the most time-consuming because it tends to exploit all the available iterations. Moreover, when we reduce the number of maximum iterations to 10, it happens to be far from the optimal solution and so the recall drops

drastically. As expected, this does not happen with a very small learning rate, for which the result stays close to the initial guess (the result without optimizer is reported in Tab. 3.4). By looking at the last column in the table, we see that a smaller learning rate than the default one seems to be a reasonable choice since it does not deviate too much from the initial guess and it converges to a good result much faster. The *break_thresh_ratio* is another key factor for the efficiency of the process: as expected the timing increases inversely to this parameter value, which still provides a good recall when it is relaxed up to 0.01. Overall, besides few extreme cases, the performances are quite similar for all the configurations. For this reason, we presume that we reached the upper limit of the model for the selected quantization step of 5 mm. Therefore, we also tested another configuration with a smaller voxel size, by hoping to improve the contribution of the optimization step. Indeed, as reported in Tab. 3.10, we can see that both recall and timings increase when we halve the voxel size. In the end, the user will need to choose the best trade-off according to the requirements. When the time constraint is crucial, it is probably preferable to have a faster solution even if it drops the performance a bit. According to all the tests we made, our choice is to leverage the fastest solution, *i.e.* the DGR without its optimization block. This method becomes the preferred configuration when we need to provide on-the-fly tracking recovery for our handheld scanning device.

BT= 10^{-4} / LR= 10^{-1}			**BT= 10^{-4} / LR= 10^{-4}**		
MI	***Recall [%]***	***Time [ms]***	**MI**	***Recall [%]***	***Time [ms]***
10^1	4.89	24.61	**10^1**	54.56	**18.06**
10^2	43.56	201.71	**10^2**	54.80	94.20
10^3	**56.51**	1674.45	**10^3**	54.90	483.94
MI= 10^3 / LR= 10^{-1}			**BT= 10^{-4} / MI= 10^3**		
BT	***Recall [%]***	***Time [ms]***	**LR**	***Recall [%]***	***Time [ms]***
10^{-3}	55.74	1123.64	**10^{-4}**	54.89	483.94
10^{-2}	54.26	898.42	**10^{-3}**	55.64	531.28
10^{-1}	44.83	632.21	**10^{-2}**	56.28	583.30

Table 3.9: Recalls and timings on DenseMatch for different configurations within the optimization block of DGR [17]. Here "BT" stands for *break_thresh_ratio*, "MI" refers to *max_iter* in Algorithm 1 and "LR" refers to *learning_rate*. In each test we fixed two parameters and we tested the pipeline by varying the other one.

Iterative block on LMVA As for DGR, LMVA offers the opportunity to refine the pose estimation through an iterative block that leverages the weights extracted from the set of correspondences. In this case, the iterative component is part of a wider context in which an end-to-end learned multiview alignment is performed via the Iterative Reweighted Least Squares approach. Nevertheless, the code for the multiview algorithm is not available

Voxel Size	BT Ratio	*Recall [%]*	*Time [ms]*
5.0 mm	10^{-1}	44.83	632.21
5.0 mm	10^{-3}	56.51	1674.45
2.5 mm	10^{-1}	47.12	650.22
2.5 mm	10^{-3}	65.91	2559.66

Table 3.10: Results on DenseMatch using multiple configurations for DGR [17] optimization block and different quantization voxel sizes. *BT Ratio* stands for *break_thresh_ratio*.

yet and we have only the local pairwise functional block. The authors set to 4 the number of iterations proposed in the paper but then they halved this value in the best model they provided. Nevertheless, we study how the drop of the iterative-based refinement can impact the performance. In Tab. 3.11 we report the test we performed on both the datasets using LMVA without the refinement and we compare them with the results already presented in the previous section. Again, we can see that the refinement was helpful only in few cases since the recalls did not increase significantly for any of the two datasets. As expected, in both tests the timings are halved, getting LMVA even closer to DGR in such terms.

Configuration	Recall	TE [mm]	RE [deg]	Time [ms]
3DMatch w/o IR	88.10%	76.67	2.53	**9.6**
3DMatch with IR	**88.90%**	71.10	1.83	18.4
DenseMatch w/o IR	48.20%	9.95	2.45	**9.3**
DenseMatch with IR	**53.60%**	10.88	2.18	15.8

Table 3.11: Results for LMVA on the 3DMatch and DenseMatch datasets, either by using or not the iterative refinement (IR)

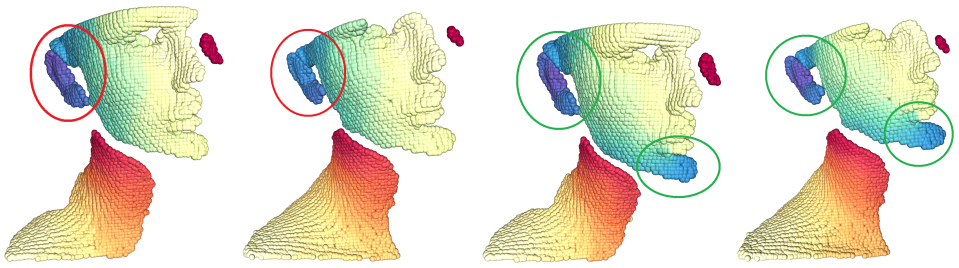
4.4.6 DenseMatch model fine-tunings

With the same spirit as the previous paragraph, we refine the FCGF model (pre-trained on 3DMatch data only), by splitting DenseMatch into two sets. In particular, we keep 4 scenes, acquired onto two significantly different subjects (*i.e.* `dummies` and `geoms`) for testing. The remaining data is used for training, excepting the `helmet` scene, which is used for the validation. The learning rate is set to $10^{-}2$, while the batch size is equal to 4. We stop the training after 100 epochs. In Tab. 3.12 we report the results of the renewed tests with the resulting model: the recall increases in all cases for `dummies` scene (a data that lacks of complex geometry) and proves to be solid also with the `geoms` case. A better understanding can be drawn from tSNE, as shown in Fig. 3.23. The refined model presents a better distinction with respect to the previous one in terms of the different geometric features

contained in two frames of *dummy 01* (especially the chin and the right ear).

	RANSAC F [114]		DGR [17]		LMVA [18]		*LMVA ref*
Scene	*FCGF [15]*	*FCGF ref*	*FCGF [15]*	*FCGF ref*	*FCGF [15]*	*FCGF ref*	*FCGF ref*
dummy 1	74.00%	**76.50%**	62.50%	**85.80%**	62.80%	71.70%	**74.80%**
dummy 2	80.40%	**82.40%**	62.50%	**81.10%**	66.20%	73.60%	**77.70%**
geom 1	94.0%	**95.60%**	**89.10%**	83.10%	84.20%	81.40%	**83.60%**
geom 2	74.3%	**79.20**|%	**74.60%**	74.30%	62.10%	63.00%	**64.70%**

Table 3.12: Comparison between registration results with default and refined (marked with *ref*) FCGF model. The last column reports also the result after refining the LMVA model. The refinement has been performed on a subset of scenes from DenseMatch. The remaining scenes were used for testing.



(a) Frame 65 (b) Frame 70 (c) Frame 65 (d) Frame 70

Figure 3.23: Color-coded features on fragments pairs in scene `Dummy` from DenseMatch [J2]. Point clouds are mapped to a scalar space using t-SNE [187] and colorized with spectral color map. (a-b) FCGF *pre* model used to describe each point. (c-d) FCGF *ref* used instead.

Moreover, we perform a fine-tuning of the registration models. For DGR we made several attempts but unfortunately, we were not able to refine the model due to an unsolved bug in the code (we attribute the problem to the instability across different versions of the MinkowskiEngine, which is the back-end developed by the same author of DGR). We also tried to retrain the network from scratch. However, in this case, the process was unfeasible due to the too slow training time (it took 3 days for 20 epochs only). We saw that the problem is already raised in the project repository by other users, however, it still results as open. Differently, we succeeded to improve LMVA. For its fine-tuning, we use the same training set we used for FCGF. We train the network for 150 epochs, we set the batch size equal to 4, the learning rate equal to 10^{-4} while the weight of the transformation loss has been set equal to 4 times the classification loss. All the other parameters are kept default, except for the distance threshold to define ground-truth labels. This was a hard-coded value into the original code and caused some pain in debugging. It is indeed critical to define ground-truth correspondences between aligned

pairs. We set this value to as 3 times the used voxel size, which is again 5 mm. In the last column of Tab. 3.12 we report the new recall values for the scenes of `dummies` and `geoms`. As sought, also in this case the refinement helped the network to learn to manage better this different (denser and smaller) kind of data.

Overall, by refining the models we obtained significant improvements. This justifies our comparative work and the provision of the new DenseMatch dataset to allow more comprehensive benchmarks. The fact that fine-tuning is more effective on the `dummies` scene rather than on the `geoms` ones is not surprising since the former is more representative of the differences with respect to 3DMatch carried by the new kind of acquisitions. However, this also evidences that data variability remains a critical factor. This is why we remain committed to future updates of the DenseMatch dataset with new acquisitions.

5 Conclusions

Aiming at finding new approaches to solve critical aspects affecting real-time reconstruction systems, we reviewed the current state-of-the-art for DL-based solutions addressing the problem of 3D registration, including 3D local feature extraction and robust correspondence matching for coarse alignment estimation.

Then we analyzed and compared, for the first time, very recent DL-based contributions in the field of 3D view alignment, specifically for the key activities aforementioned. To extensively challenge these methods, other than using a popular benchmark dataset (3DMatch), we introduced a comparison with the new set of data DenseMatch. This dataset contains a series of scans coming from a handheld device that targets object reconstruction with a high level of detail and produces much more dense data than the counterpart in 3DMatch, which targets indoor reconstruction instead.

From our evaluation, two networks have emerged as valid alternatives for coarse registration against the well-established handcrafted solutions, specifically DGR [17] and LMVA [18]. These networks proved to offer a fairly good registration recall in the evaluation of alignment performance. The former proved to be the fastest method and outscored, by a large margin, the handcrafted baseline used for comparison. The computational speed is indeed a critical parameter if we want to deal with real-time working constraints, and novel deep-learning solutions offer a significant boost in terms of speed, remaining at least on par with widely diffused methods for robust registration, such as RANSAC.

We then fine-tuned and improved the performance of original DL models by leveraging on the newly introduced dataset. This demonstrates the benefits of the introduction and adaptation to novel datasets to better cope

with a wider range of real use cases.

Overall, the assessment has been revealing to understand we can leverage on this technology to develop a real-time working pipeline which can benefit from the advantages we highlighted in our examination. In the last chapter of this work we are then going to explore how we can develop a deep learning framework to provide a tangible improvement for the user experience and for final reconstruction quality of the scanner under study.

Chapter 4

Real-time 3D reconstruction pipeline in a DL framework

In the previous chapter we evaluated new DL-based technologies to address critical tasks for 3D registration. Our assessment highlighted how such methods can offer a robust solution to deal with complex conditions and proved to achieve high-end performances thanks to the efficient implementation which is powered by GPU-based parallel computation. The effectiveness of these solutions can be helpful to design a new product that can target very compelling and challenging tasks as the reverse engineering [25], the object modeling in industrial applications [24] and even in the bio-medical field, where for instance it should be highly appreciated to rely on an affordable yet accurate and easy-to-use device to address the task of human body scanning for orthotics [209]. Therefore, in the last part of this thesis, our goal is to leverage the know-how we acquired to develop a new pipeline for our scanner. Such a new pipeline has to guarantee high standards in 3D reconstruction, both for accuracy and robustness and also to comply with the real-time requirement. Overall, in our framework we mix DL-based solutions with revisited classical methods to tackle the critical tasks of robust camera tracking, pose optimization and dynamic reconstruction. A careful combination of the two worlds produces our final result, which we arrive at in two steps.

At first, in Sec. 1, we present a module which offers a safeguard system for rapid and robust camera relocation by exploiting the DL methods we analyzed earlier. The system is meant to be activated whenever is necessary to align a frame to the model with no *a priori* information regarding the camera pose, that is when ICP alone is no longer reliable. The two common scenarios in which this happens are when we restart the scanning after a pause or when we lose tracking (typically due to too fast movement) in the middle of a scan. In particular, the current pipeline, which is based on KinectFusion [20], does not offer a solution to the former failure cause while it adopts a naive approach for the latter, by simply notifying the user

about the tracking loss and asking to come back to get closer to the last valid camera pose. However is difficult to do this properly in practice, and the user experience suffers from the frequent high difficulty of recovering the scan after losing tracking. We evaluate the positive impact of the new safeguard module on the reconstruction pipeline while keeping KinectFusion for reference.

Finally, in the 2 we make a decisive step forward to complete our pipeline. Indeed, the safeguard module is helpful to solve tracking loss but it does not account for other issues. Specifically, the workflow still needs a solution to adjust the poses after a global evaluation. The adjustment is crucial to fix an eventual misalignment or to refine the model. Therefore, partially inspired by how these issues are tackled in BundleFusion [19], we propose a new framework which exploits global adjustment of frame chunks (derived from a 2-level hierarchical structuring of the 3D frame sequence) and uses modern data-driven methods for the geometry-based inter-chunk coarse alignment. The proposed system is also conceived to overcome some limitations of BundleFusion that arise in the context of Real-Time 3D object scanning. Due to its properties, we call this framework Deep-BundleFusion. Moreover, along with the method explanation, we provide the results of several tests we run to show how our solution can deal with specific applications that revealed to be challenging for BundleFusion. A discussion about the results completes the chapter and opens to considerations about future possible works.

1 DL-based safeguard module for fast tracking recovery

The original pipeline designed for our scanner – the InSight – was introduced and analyzed in Chapter 2 Sec. 1.2. Such a pipeline is basically an adaptation of the one proposed by Newcombe *et al.* in KinectFusion [20] (for the sake of clarity reference we depict it once again in Fig. 4.1). The method is known to lack of a non-trivial solution for addressing the episodically tracking loss, since the only strategy available is to ask the user to move back the camera nearby the last valid pose hopefully to recover the track and continuing the scanning.

In this section, we propose a first DL-based workflow revision by adding a safeguard module which is silent through the regular scanning and it is activated whenever a tracking loss is detected. We start by describing the method, especially the new block and the implementation setup. Then we explain the tests we performed and finally we comment the results of the analysis.

The core of this solution has been presented as part of the publication [J1].

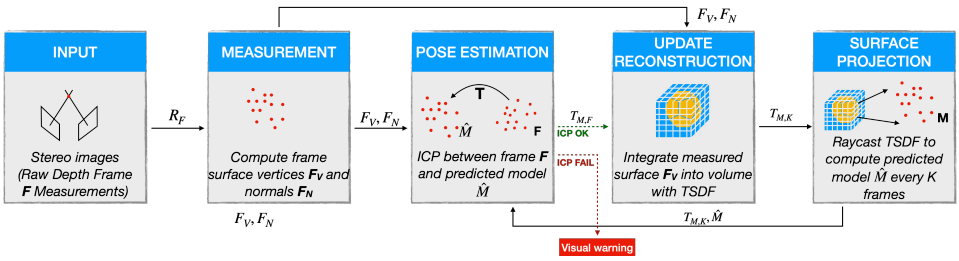


Figure 4.1: InSight scanner starting pipeline for 3D reconstruction.

1.1 Method

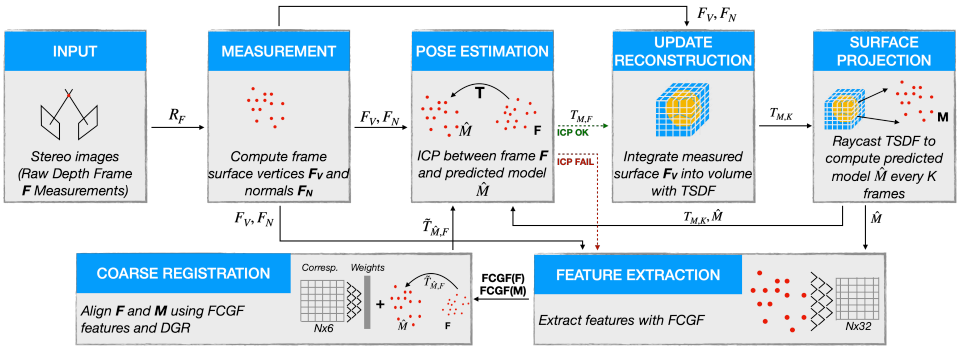


Figure 4.2: Revised pipeline for 3D reconstruction. With respect to the original one, we add two blocks at the bottom, aimed to provide a safeguard solution with robust 3D registration to address ICP failure cases (meaning that tracking is temporarily lost). The two blocks exploit DL-based method for 1) extracting geometric features (FCGF [15]) from the dense sets of points of the frame-model pair and 2) computing the coarse registration matrix (DGR [17]) to re-establish a good estimate of the camera pose.

Firstly, we initialize a volume with preset values for `voxel size` and `sdf trunc` for SDF truncation length [62]. Then we load the first RGB-D frame and we integrate it into the volume. The coordinate system of the first frame is then used as reference moving on. We then ray cast the volume to extract the first key frame which is used to align the next frame. From these two RGB-D images we extract the surface vertices $\boldsymbol{F}_V$ and normals $\boldsymbol{F}_N$ and we use ICP point-to-plane as aligning algorithm. We repeat the operation for the first 10 frames, by ray-casting a new model after every integration to ensure to have a reliable model at the beginning of the construction. At each new frame we exploit the previous alignment to transform the current view. Such a pre-alignment is useful to put ourselves in the known closest camera pose so that ICP is theoretically well conditioned. Obviously, the assumption does not hold when we lose the track temporarily or when we restart the scanning

after an interruption. Starting from the tenth frame we move forward by following the workflow presented in Fig. 4.2. Differently from the previous approach (Fig. 4.1), when we detect an ICP failure we do not just warn the user but we also try to solve the next ill-conditioned registration using a coarse registration method, implementend in the safeguard module.

1.1.1 Safeguard module

At the *Pose Estimation* stage we make a sanity check of the ICP result by setting two thresholds: the first one refers to the minimum fitness, which indicates the percentage of closest points having distance below a preset value (see `icp dist` in Tab. 4.1), referenced as *overlapping points* from now on. The amount of overlapping points O is evaluated for a set of closest points $\mathcal{C}$ as:

$$O = \sum_{(i,j)\in\mathcal{C}} \mathbb{I}\left[d_{ij} < \texttt{icp dist}\right] \tag{4.1}$$

where $\mathbb{I}[\cdot]$ is 1 when $\cdot$ is true and 0 elsewhere, while d_{ij} is the point distance. We set the minimum fitness – τ_{fit} – to be 50% for the source point cloud. In practice, for a source point clouds having N total points and O overlapping points, the check on fitness is:

$$\begin{cases} \text{Valid} & \text{if} \quad \frac{O}{N} > \tau_{fit}, \\ \text{Invalid} & \text{elsewhere} \end{cases} \tag{4.2}$$

This is a reasonable value in our scenario since the furthest point cloud to align is at most $K = 10$ frames apart from the key frame, and in practice we found ourselves well above the threshold when no error occurs. The second threshold is for the Root Mean Squared Error (RMSE), estimated over the overlapping points only. If the value is extremely close to the minimum acceptable distance for ICP `icp dist`, it should be possible that we are converging to a local minima, so that a false alignment is more plausible than the case in which RMSE is well below that value. We set the threshold τ_{rmse} such that the ratio between the current RMSE and maximum expected one is below 90%. The check on RMSE is:

$$\begin{cases} \text{Valid} & \text{if} \quad \frac{RMSE_{curr}}{RMSE_{max}} < \tau_{rmse}, \\ \text{Invalid} & \text{elsewhere} \end{cases} \tag{4.3}$$

When the sanity check returns an invalid alignment, the volumetric integration is skipped and, instead, a warning about the tracking problem is printed on screen. Then, the next frame is passed to the coarse registration module: a set of geometric features is extracted both from the working fragment and the current model, the set of putative matches is created by means of the nearest neighbor search in the euclidean space of the descriptors and then

it feeds the registration block, which outputs the estimated registration matrix $\hat{\boldsymbol{T}}$ in the end. Due to the combination of reliable and fast computation emerged from our assessment (see Chapter 3 Sec. 4.4), we select FCGF [15] for extracting the features and DGR [17] for the coarse registration. Regarding DGR, we opt to remove the optimization block, since a coarse estimation is sufficient at this stage and, according to the examination we just mentioned, we can trade the incremental accuracy we could gain with the optimization for a much faster solution without it.

1.1.2 Development and Settings

To develop our pipeline we relied on Open3D [55] (v11.0) as the main framework for running registration algorithms (ICP and RANSAC), managing RGB-D images, point clouds and TSDF volume with hashing voxels [139] and visualization. In order to align the point clouds in a feasible timeframe, we down-sample them using `voxel_down_sample()` built-in function in Open3D `PointCloud` class. A higher down-sampling factor is then used to quantize the input for coarse registration, chosen to be 5 mm accordingly with the evaluation we performed in Chapter 3 Sec. 4.4. In Tab. 4.1 we report the main settings used to run our tests.

Name	Scope	Value	Note
`sdf voxel size`	Integration	1 mm	Resolution of the building volume
`sdf trunc`	Integration	6×`voxel size`	Truncation distance for TSDF (see [62])
`z min`	Integration	0.5 m	Min. depth in RGB-D for volume integration
`z max`	Integration	0.7 m	Max. depth in RGB-D for volume integration
`z min rayc`	Ray Cast	0.5 m	Starting distance from camera for ray casting
`z max rayc`	Ray Cast	2.0 m	Maximum distance from camera for ray casting
`chunk size`	Ray Cast	10	Number of frame to align to same model before updating
`fine sampling`	Registration	2.5 mm	Sampling size to apply before fine registration
`coarse sampling`	Registration	5.0 mm	Sampling size to apply before coarse registration
`icp dist`	Registration	1.5×`voxel size`	Max. distance to consider two points as overlapping
τ_{fit}	Registration	50%	Min. fitness for valid ICP (see Eq. 4.2)
τ_{rmse}	Registration	90%	Min. RMSE ratio for valid ICP (see Eq. 4.3)

Table 4.1: Settings used to run 3D reconstruction pipeline during safeguard module evaluation.

1.2 Experimental setup

Overall, we test 5 different setups:

- `w/o Safeguard` refers to the original solution (Fig. 4.1), which we can reference also as our own implementation of KinectFusion.
- `w/o Safeguard` X means that ICP alignment algorithm is replaced with method X. We try both a classical approach (RANSAC) and a DL-based one (DGR).

- `w/ Safeguard` (X) is the new proposal with the additional module for a safe camera relocation. As before, we test RANSAC and DGR (this time for coarse registration only, while ICP is maintained as standard registration approach during the regular flow).

Regarding RANSAC, we adopt the solution implemented in Open3D [55] using correspondences directly.

1.2.1 Data

In order to evaluate the performance of the new system, we rely on the DenseMatch dataset (Chapter 2 Sec. 3 [J2]). We select a bunch of scenes from the whole collection (Fig. 4.3), by picking the best representatives for different challenges we can encounter in real applications: we use the `bike helmet` and the `column` because of their intricate but repetitive patterns, the `dummy` and the `statue` due to the lack of reliable geometric features over most of their surfaces and finally the `geom` piece of marble as counterexample. Moreover, we try a new expedient to simulate a complex scenario: instead of scrambling the sequence of one scanning session, we stick together a pair of session of the same scene. In such a way, we introduce a hard transition from one session to the next, which will produce an ICP failure for sure. This is not strange, indeed it is totally plausible, in a real context:: a scanner can have such a big jump if we voluntarily interrupt the acquisition for any reason or if we simply miss the target object with the active field of view of the scanner for a moment and then re-frame it from a different position.

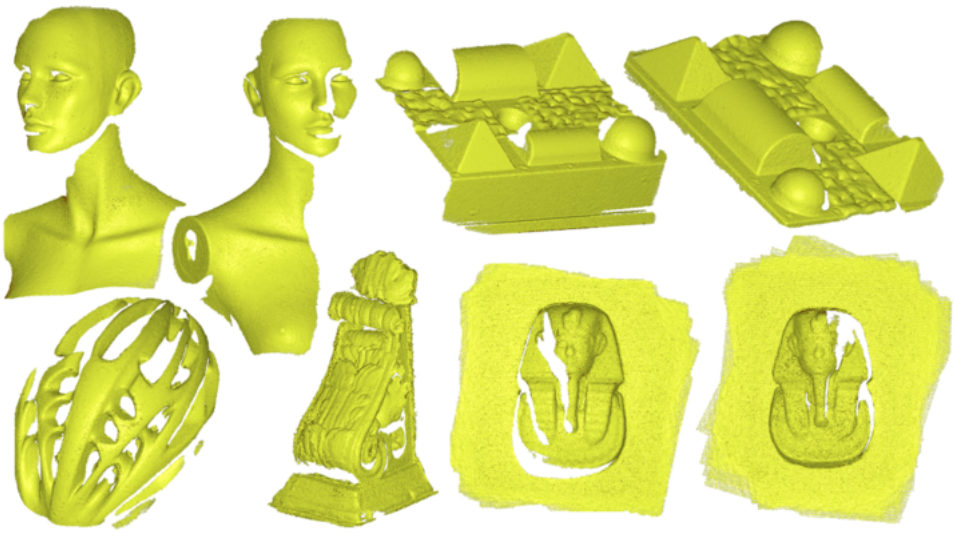


Figure 4.3: 3D models from DenseMatch [J2] we use for testing the pipeline with the safeguard module.

1.2.2 Evaluation Metrics

The metrics we analyze are:

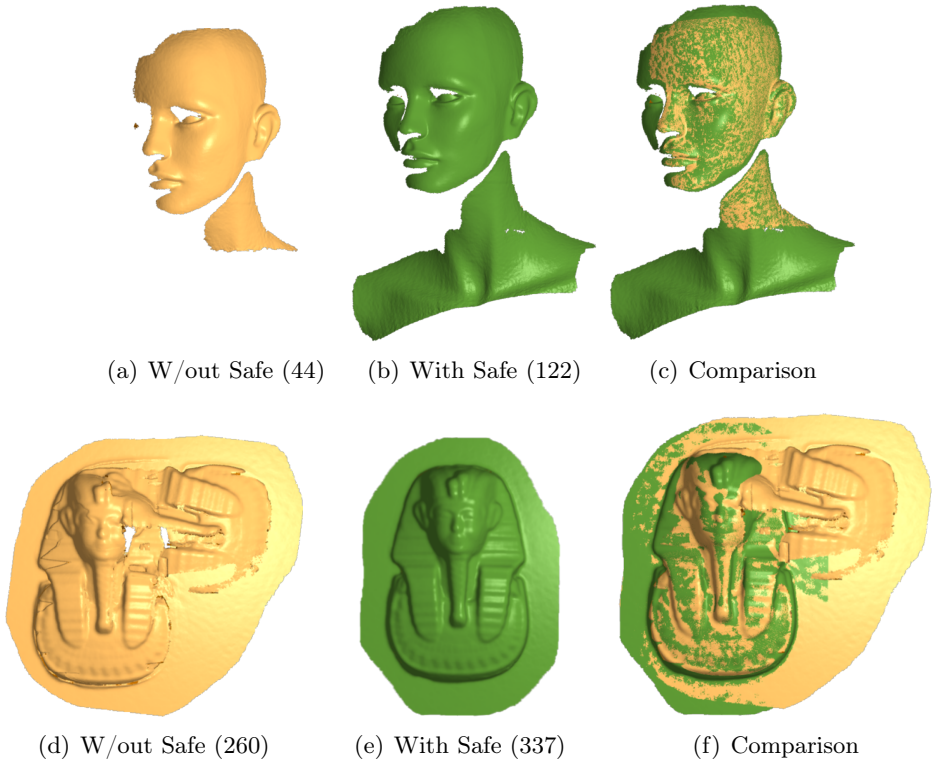
- The percentage of integrated frames with respect to the total per scene.
- The average closest-point distance between the estimated 3D model and the ground truth we provide in DenseMatch, in order to determine whether particular errors occurred during the reconstruction process.
- The timings to run the main steps in the safeguard mode, to evaluate the real-time feasibility.

1.3 Results

1.3.1 3D Reconstruction performance and robustness

In Table 4.2 we report the results of our tests related to the 5 configurations described previously. Initially we focus on the reconstruction performance of the first and the last two columns, namely on the original pipeline and the one with the DL-driven safeguard module, in the two analyzed configurations for the coarse registration module. A first interesting case is the one referring to the two scenes of *dummy*. In both cases, using the original pipeline, we completely failed to recover the track after the first failure. The final distance from the ground truth is low but yet it refers to a sub-part of the whole model and so it is incomplete. In these cases, the safeguard methods solved the issue completely by registering the next frame after the failure (actually RANSAC failed the first attempt but succeeded at the second). The resulting models are indeed almost complete and the distance from the ground truth shows that no false alignments occurred after the breaking point. This is evident, in Fig. 4.4(top), since the safeguard-based method reconstructed `dummy 02` (in green) properly whereas the original solution (baseline, yellow) lost the alignment moving from the face towards the neck and reconstructed the scene partially. In general, the DL-based method has been always on par or incrementally better than RANSAC from a performance standpoint. This fact already justifies the choice of adopting DGR over RANSAC, but such decision will be further strengthen by looking at the timings analysis we perform below.

Another very interesting case is `statue 01+02`: this test scene comes from merging two scans of a `statue` object. In the first acquisition, we covered almost all the statue from left to right, and in the second run we filled the holes by acquiring in the opposite direction. In particular, we started a new session from a different angle with respect to where we ended before, and then we moved around the object also by revisiting some spots. With the traditional method, we aligned correctly all the frames from the first scene. However, as expected, during the transition to the new starting



(a) W/out Safe (44) (b) With Safe (122) (c) Comparison

(d) W/out Safe (260) (e) With Safe (337) (f) Comparison

Figure 4.4: 3D reconstruction comparison between the method with the baseline pipeline (in yellow) and the new solution with DL-based safeguard (in green). (top): the `dummy 02` model. (bottom): the `statue 01+02` model. Between the brackets is reported the number of integrated frames. The third column shows the two reconstructions overlapped, to highlight the differences.

position we lost track and we did not recover it until we re-framed the scene – later in the sequence – from a position close to the latest valid one. In total, with the baseline setup, we lost 78 frames over a total of 338. In addition, we got a false positive alignment (we ascribe the error to a sub-optimal ICP failure detector), so that the final reconstruction presents two distinct reconstructions of the statue fused in one volume (see Fig. 4.4(bottom)). On the contrary, the new setup recovered the track immediately after detecting the first error (due to the transition). The coarse alignment was good enough to help ICP to converge again to a valid solution and the scanning continued smoothly.

For further comparison, we also try to replace the fine registration (ICP) with coarse methods only. In the central columns of Table 4.2 we then report the results of reconstruction using either DGR or RANSAC instead of ICP as first alignment choice. We remind that we are allowed to use a fine

Scene (frames)	w/o Safeguard ICP		w/o Safeguard RANSAC		w/o Safeguard DGR (with opt)		with Safeguard RANSAC		with Safeguard DGR (w/o opt)	
	Integr.	**Dist.**	**Integr.**	**Dist.**	**Integr.**	**Dist.**	**Integr.**	**Dist.**	**Integr.**	**Dist.**
`bike helmet` (168)	**100%**	**0.1**	50%	4.1	38.7%	3.2	**100%**	**0.1**	**100%**	**0.1**
`column 01` (232)	96.5%	**0.2**	17.2%	6.3	18.1%	6.9	99.1%	**0.2**	**99.6%**	**0.2**
`dummy 01` (139)	43.2%	**0.1**	**100%**	8.0	99.3%	7.4	99.3%	0.2	99.3%	**0.1**
`dummy 02` (146)	30.1%	**0.1**	**95.9%**	6.4	59.6%	6.4	75.3%	0.2	83.6%	**0.1**
`geom 01` (80)	**100%**	**0.1**	26.3%	4.3	**100%**	2.9	**100%**	**0.1**	**100%**	**0.1**
`geom 02` (132)	87.9%	**0.1**	14.2%	7.0	31.8%	7.1	**97.7%**	**0.1**	**97.7%**	**0.1**
`statue 01+02` (338)	76.9%	5.7	15.4%	34.2	14.2%	29.7	**99.7%**	**0.1**	**99.7%**	**0.1**

Table 4.2: Multiple pipeline configurations comparison for 3D reconstruction with the handheld InSight scanner. *Integr.* is the percentage of integrated frames, with respect to the total contained in each scene. *Dist.* is the average closest-points distance (in mm) between estimated model and ground truth.

registration method straightforward because we assume the local proximity between the model and the frame we want to align. This is valid only (and not always) in particular conditions such as the case of high rate acquisition with an handheld scanner. However, other situation does not consider such local property, therefore we try to evaluate also the case in which we cannot rely on it. For DGR we reintroduce the optimization block, aiming at improving the accuracy of the final registration. We keep the same sanity check on the registration as we did for fine registration, but the minimum distance to define overlap (Eq. 4.1)is scaled with the same ratio as we scaled `coarse sampling` with respect to `fine` in Tab. 4.1. In these scenarios, it is interesting to notice that we did not just get worse quality results for the final estimation, but we also had lower percentage of integrated frames with respect to the ICP-based case. This is easily explained by the fact that our model is growing by means of a volumetric integration: whenever the data is sub-optimally aligned to the model and consequently integrated into it, it then creates artifacts which reflects in the next model update. In practice, this is caused by noise accumulation derived from each alignment independently. After a certain amount of badly frames is accumulated, it can happen that the model is too noisy and the alignment fails. Indeed we could increase the quantization level to compensate the noise during the integration, but we do not want to loose too much resolution in this context. A warning signal is the result we obtained with `dummy` dataset in the RANSAC only configuration: even if the reconstruction is wrong, the method continued to pass the ICP failure detection until the end.

Overall, this is just a trivial test aimed to highlight the fact that we cannot rely exclusively on DGR to fulfill the reconstruction. Moreover the decision of reintroducing the optimization for DGR has cost in terms of computational speed, as we already deepened when evaluated the method in Chapter 3 Sec. 4.4. On average, in fact, it is slower than ICP and it can not be considered as a valid alternative for a fine alignment on-the-fly. Such a problem stands true also for RANSAC. Moreover, if we used the non

Scene	Feat. Extr. (FCGF) [ms]	Feature Match. [ms]	Registr. RANSAC [ms]	Registr. DGR [ms]	Safe (FCGF) Total [ms]	Safe (DGR) Total [ms]
bike helmet	27.3	9.5	82.4	2.8	119.2	39.6
column 01	31.4	9.3	70.0	3.1	110.7	43.8
dummy 01	24.1	8.5	72,5	2.4	105.1	35.0
dummy 02	25.9	9.2	64.2	2.7	99.3	37.8
geom 01	27.5	9.1	94.6	2.4	131.2	39.0
geom 02	27.7	9.3	88.9	2.6	125.9	39.6
statue 01+02	40.2	12.4	131.1	3.4	183.7	56.0
Avg	29.1	9.6	86.2	2.8	125.0	41.5

Table 4.3: Average timings for the coarse registration algorithm, comprising feature extraction (FCGF [15]), matching and registration (RANSAC [114] and DGR [17]).

optimized version of DGR, we would end up with an even more rough result, which again brings the above mentioned problems regarding the noisy model.

1.3.2 Timings and real-time operation

Finally, in Tab. 4.3 we report the average timings for the coarse registration modules. We point out that the estimate of the frame-rate of the system heavily depends on the amount of interventions we need from the safeguard module. Indeed, if we consider a regular iteration (comprising frame acquisition, ICP registration, volumetric integration and visualization), we estimated that it takes 150 ms on average when running solely on CPU, which means we get close to 7 fps. In our experience, such a value is borderline to get a real-time feedback. Then, every time we need to run the coarse registration we inherently worsen such value. In particular, when we uses the data-driven approach, we were able to perform the coarse registration task taking 41.5 ms on average (in such evaluation we consider all the necessary steps, namely feature extraction, feature matching, and registration estimation). Overall, this value sounds reasonable since we need to apply it sporadically. Indeed, during our tests the only case which required more than two coarse registration attempts was `dummy 02`, which failed 7 distinct times the fine registration alone throughout the scanning (probably due to the critically flat area of the abdomen in which ICP drifted consistently). Since we lost a total of 24 frames, in such a case, it means that the safeguard added nearly 0.9 seconds to the total processing. Therefore, on average, we dropped from 6.7 to 6.5 fps, which is marginal. On the other hand, RANSAC took considerably much time for processing with respect to DGR. Specifically, the solely RANSAC registration accounts for 64 ms on average which means almost 100 ms for the entire coarse registration. For instance, in the example of `dummy 02` we add 2.4 seconds in total. Such an overhead produces a drop of more than half frame per second, moving to 6 fps, which is a 10% worsening. In the hardest case, *i.e.* the `statue 01+02` scene (a small statue of a pharaon is placed onto a table and a portion of it around the statue is

also scanned, so the final RGB-D is almost full of valid points), this timing goes up to 180 ms. Such an overhead, even though it is sporadic, it produces a visible delay in the operation and it impacts on the user experience overall.

1.4 Conclusion

To conclude, the proposed pipeline shows how a robust data-driven method can help the workflow also in the context of an additional acquisition from a new vantage point. We think that the results we obtained are already highly significant and clearly indicative of how to fully exploit the concurrent good accuracy and timing performance of DL-solutions for 3D view alignment, in real-time 3D reconstruction systems.

However, some questions remained open: thus far we just addressed the problem of robust tracking, but we did not cover the topic of model refinement. Moreover, we tested the performance of the coarse registration module but we did not propose any real sort of application yet. Therefore, in the final section of the thesis we finally try to deal also with these topics.

2 Deep-BundleFusion (DBF)

Having a system to rapidly recover the camera pose after an interruption (whether it is accidental or intentional) it is certainly critical to guarantee the flow of the scanning to be smooth, so that the user experience is valuable and the scanner results easy to use. In the previous section we saw that we can leverage advanced models to handle the problem of pairwise coarse registration in a robust fashion. However, such a solution addresses only the local registration of a fragment to the model which is usable at the moment. It is then a partial contribution which does not account for the whole information of the model. As expressed by Dai *et al.* in BundleFusion (BF) [19], ensuring the *global model consistency* is necessary to overcome the issues related to local error accumulation and compensate for drift, which inherently affects also the frame-to-model registration approach.

Inspired by their work, we then try to rethink our workflow to achieve a similar sophistication in order to reach the higher global model consistency, while concurrently exploiting the solutions that qualify our pipeline targeted to the devices and application scenarios of interest in this work. The three critical components we address in our final solution are: 1) the camera tracking, once again (but without wanting to overturn it, as explained below); 2) the pose optimization, which is crucial to the global consistency; 3) the dynamic reconstruction to refine the model on-the-fly, which creates a better user experience and still helps the frame-to-model registration, by constantly adjusting the model we rely on.

In the following section, we review these components by introducing the new framework, which we rename *Deep-BundleFusion* (DBF) to mix the

nature of the two "fusing" approaches. Moreover, in Sec. 2.2 we detail the software we developed to replicate the reconstruction pipeline of the scanner. We aim to release it in the near future. Then, after reviewing the pipeline we set the ground for a final test: in in Sec. 2.3 we talk about the experimental goals and setup, and we describe the data. Our attention is primarily focused on covering the problem of object reconstruction, although we will not overlook a direct comparison between technologies on indoor scene reconstruction scenarios. In Sec. 2.4 we report the results of our tests, where we extensively analyze our method and fairly compare it with the reference BundleFusion implementation. We conclude with a discussions about strengths and residual limitations of our proposal.

2.1 Method

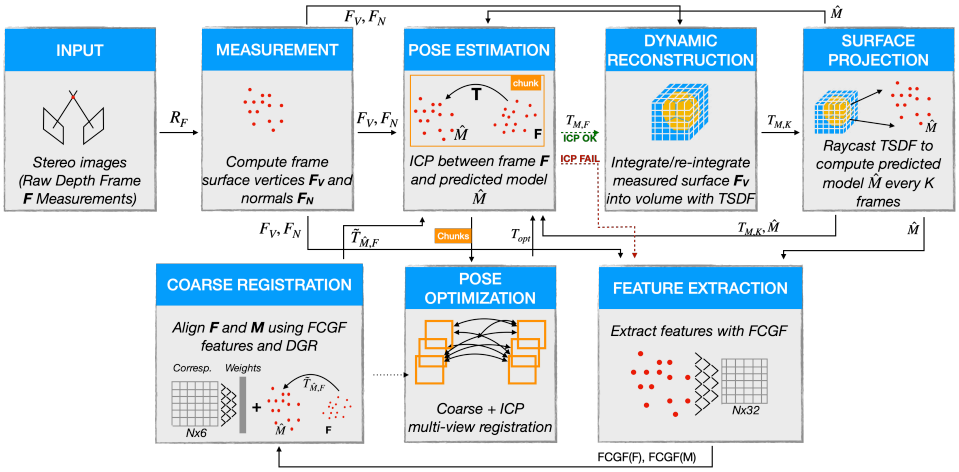


Figure 4.5: DBF pipeline. With respect to the previous proposal, containing the safeguard module for fast and robust coarse registration, we add an optimization module to adjust the poses and re-integrating them dynamically. The optimization of the poses is evaluated globally by grouping chunks of poses together.

First of all, we point out that we already tried to run the straight implementation of BF on our data. However, the results of such tests showed us the limitation of the approach in our working field. In fact, recalling the discussion we made in Chapter 2 Sec. 2.3, the nature of the data and the acquisition process we target are, in some aspects, quite far from the ones BF was developed for. In our context, relying on a repetitive sparse-then-dense alignment method for every new frame it seems not just an over-complication but, in the end, it appears to be also detrimental to the final reconstruction. During our tests, we recognized in the extensive research of 2D correspondences (based on SIFT feature matching) the main point of failure for us.

Indeed, BF targets the reconstruction of large-scale indoor scenes, where most of the frames have a wide area filled with multiple objects to hook on. Our scanner, conversely, is meant for smaller scale operation and object reconstruction, therefore most of our frames focus on small areas which are defined over the single object/subject surface we want to reconstruct. Finally, these areas are also smooth, possibly lacking of variegate texture to rely on or having repetitive patterns hard to disambiguate.

This is why our strategy is conservative with respect to the framework we designed for camera tracking. In Sec. 1 we saw that the standard alignment, complemented properly with the safeguard module based on DL architecture for sporadic robust camera relocation, produces good results already. Then we do not revise it from scratch but instead we ground on it by strengthening the baseline. Our main interest is then to extend the capabilities of our approach by adding a pose optimization module and by allowing the reconstruction to be dynamic and effective by means of a refining stage which adjusts the frames on-the-fly, similarly to what BF does. In Fig. 4.5 we depict a schematic representation of the new pipeline.

2.1.1 Camera tracking

As we said, the solution we presented at the beginning of the chapter remains a valid approach for our purposes. Indeed, from the practical standpoint, we are satisfied with an application that works at a real-time pace (thanks to an efficient ICP implementation), producing few blunders during the alignment (thanks to an handcrafted failure detector), which sporadically looses a frame. However, since the tracking loss remains critical for the application, after introducing the safeguard module (see Fig. 4.2) that we gained the possibility to recover from it without impacting the user experience in a significant way. The results we derived in Sec. 1.3 indicates that the DL models we exploit for coarse registration are valid and fast enough for re-aligning the track to the model, even in complex scenarios. Moreover, the nature of the scanner – a handheld device with live feedback during 3D reconstruction – allows to warn the user to behave accordingly to help the system whenever a problem occurs.

For these reasons, we do not intend to overturn our method here. Overall, the main concern remains detecting false alignments effectively. In fact, in Sec. 1.3 we saw at least a couple of situations in which the reconstruction moved forward by aligning all the frames, even if the model was poorly defined (especially the cases for `dummy` scene based on RANSAC only registration). These results warns us about the fact that the registration failure detector did not handle them correctly. Therefore, we try to strengthen the method by adding a couple of additional rejection criteria which are effortless to compute but can be effective to detect critical alignments.

The first new criterion is based on *fitness ratio*. In standard mode (so

without safeguard module) we evaluate the ratio between the current fitness of ICP and the previous one: if such ratio is below a threshold, we assume either that the scene changed significantly in just one frame (which is suspicious since we assume we are scanning at a high rate) or that a misalignment occurred (more likely). In any case, it is better to skip the suspicious frame and try a more robust registration by passing through the safeguard module. We formulate this rejection criterion as:

$$\begin{cases} \text{Valid} & \text{if} \quad \frac{Fitness_{curr}}{Fitness_{prev}} < \tau_{fit\_ratio}, \\ \text{Invalid} & \text{elsewhere} \end{cases} \tag{4.4}$$

Since this rejection accounts for particular cases, we set the threshold τ_{fit_ratio} equal to 70% in order to be sensitive only to extreme situations (in practice, this results in a gap from a previous fitness equal to 90% (which is roughly the value we get in good situations) to a new one equal to 60% (which is still 10% above the τ_{fit} threshold we set on the minimum valid fitness.

The second criterion is very similar but works in the metric space of the camera movement. It evaluates two thresholds: the first one is about the maximum rotation the camera is meant to have from one frame to the other, and the second is indeed for the translation. The rationale again is that if we have either a delta rotation Δ_{rot} or a delta translation Δ_{transl} out of the expected range, we do probably have a bad alignment or the camera moved too fast nonetheless, therefore it is better to skip the frame and try to use a safer registration at the next iteration. In particular, Δ_{rot} and Δ_{transl} are evaluated using equations from Chapter 3 (3.12) and (3.13) respectively and the rejection criterion is as follow:

$$\begin{cases} \text{Valid} & \text{if} \quad \Delta_{rot} < \tau_{rot} \quad \text{and} \quad \Delta_{transl} < \tau_{transl}, \\ \text{Invalid} & \text{elsewhere} \end{cases} \tag{4.5}$$

Obviously, the optimal value for these thresholds depends on the type of scanner, since we expect to be allowed to move it faster or slower according to its maximum frame rate. In Tab. 4.4 we report the values we chose based on empirical evaluations for the different datasets.

Indeed, referring one more time to Tab. 4.2 we test again the RANSAC only configuration on `dummy 01` scene with the additional checks on fitness ratio and transformation ratio. In this case the percentage of the integrated frames drops to 23.9%, meaning that the failure detector helped to recognized the non accurate registration. Conversely, the percentage remains 100% for `geom 01` and the standard ICP registration method (*i.e.* the KinectFusion configuration), meaning that the two novel criteria did non affect the reconstruction.

2.1.2 Pose optimization

BF seeks the global model consistency by proposing a hierarchical subdivision of the frames composing the reconstructing sequence. The new structure of the sequence has 2 levels in its hierarchy: the *local* and the *global* global level. In Fig. 4.6 we see the main components.

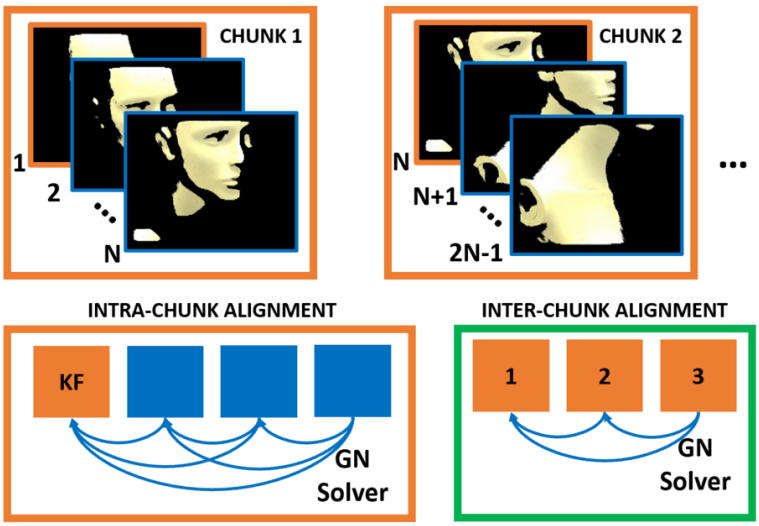


Figure 4.6: Intra-chunk and inter-chunk pose alignment for BundleFusion. Multi-view registration is computed via Gauss Newton (GN) solver.

Locally, a new frame is acquired and aligned with respect to a key frame (similarly to what we did so far). However, in BF the registration is always multi-view. When a new frame arrives, it is matched against the other frames in the bucket. Its pose is estimated from scratch by means of a Gauss Newton (GN) solver which minimizes the set of interconnected distances across the views. These distances refer to sparse correspondence matching (on top of 2D SIFT [78] feature extraction) and dense consistencies evaluation (both photometric and geometric). We refer the reader to Chapter 2 Sec. 3.2.2 for further details on the subject. In this scenario, along with the new estimated pose, also the other frames are adjusted at the end of the registration by exploiting the incremental information provided by the new view. Since the frames involved belongs to the same bucket, the local registration is called *intra-chunk*.

Then, at *global* level – *i.e.* at the top of the hierarchy – the pipeline sees only the key frames contained in the chunks representing a cluster of frames each. The same process adopted locally for registering is extended at higher level using solely the key frames. Moreover, when a pose of a chunk is optimized, all the frames associated to that chunk are rigidly transformed as well. Such an *inter-chunk* refinement is critical to the purpose of global

consistency, and this is why we wanted to adopt such a hierarchy also in our solution. Similarly to BF we split the volume into chunks but we skip the intra-chunk refinement step. Indeed, experimentally we observed that almost all the time the optimized poses were identical to the originals. In fact, since we are using a fine method as ICP for registering such a dense set of points, we end up with a set of alignments that are already sufficiently accurate. Moreover, the failure detection module ensures a higher robustness to false alignments, so that we reduce the risk of having ill conditioned chunks. Conversely, BF uses a sparse-then-dense registration as well, but in order to achieve acceptable computational performance it drastically down-samples the input data, therefore the dense alignment is less accurate by default and it is easier to have cases in which an intra-chunk refinement is necessary.

Nevertheless, these two different philosophies converge on the global refinement. As for BF, in fact, we also perform it by using the chunks as atomic elements composing the model. In our case, however, instead of using a key frame as the representative of its chunk, we create an additional small volume in which we integrate the frames in parallel to the whole volume we are constructing. Once we close a chunk then we extract the surface from this small volume and we use such a 3D point cloud as the final representative of the chunk at the pose optimization stage. Such a solution has the benefit of describing entirely the chunk, instead of using a partial representative (the key frame is just the first frame in the bucket, but the remaining frames give a marginal contribution as explained in BF [19], with the concrete risk that two keyframes find themselves too far apart). Surely, chunk-based surface extraction might not be extremely efficient (due to the repetitive integration and the additional memory consumption) but we keep its impact tolerable by cleaning the supplementary volume every time a new chunk is created (only the point cloud is preserved) and by leveraging on a fast integration via GPU implementation which keeps the time consumption below 2 ms per integration.

Finally, for the multi-view alignment, we exploit the solution proposed by Choi *et al.* [210] which is also implemented in Open3D. In brief, the method constructs a *pose graph*, the nodes of the graph being the single views, and each view i is associated with a pose matrix $\mathbf{T}_i$ while the edges connecting the nodes are associated with the transformation matrices that aligns the nodes at the ends of the edge (*e.g.* $\mathbf{T}_{i,j}$ is the transformation on the edge connecting node i and j. The set of pose matrices $\{\mathbf{T}_i\}$ are the unknown to estimate, via graph optimization [210]. In order to define the edges of our graph, we perform a set of pairwise registrations across all the chunks we collected and for each of these pairwise registration we use the safeguard module in order to align from scratch distant views in a proper way (similar to what BF does with the sparse-then-dense registration).

Unfortunately, this solution has a current limitation, due to the compu-

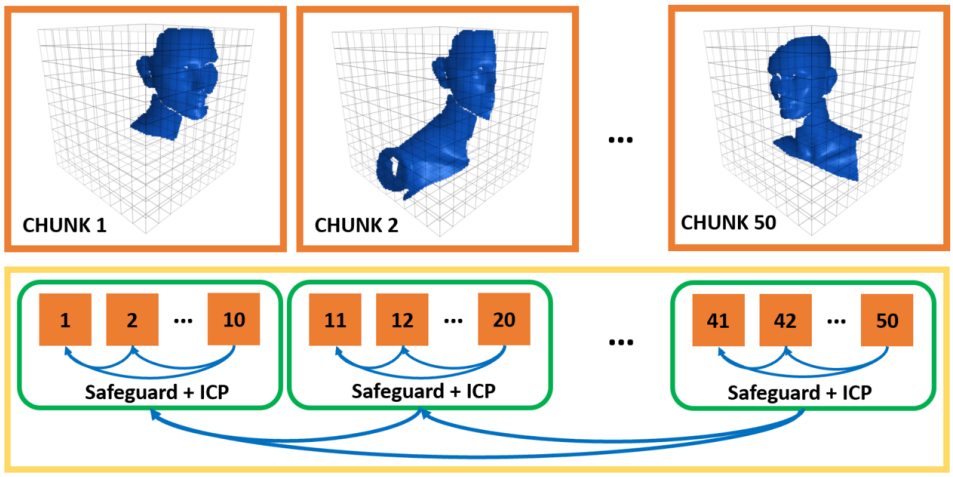


Figure 4.7: Inter-chunk refinement for Deep-BundleFusion. Pairwise registration are performed using the safeguard module and ICP. The multi-view is exploited via graph optimization [210]. A *divide et impera* solution is adopted for reducing the computational complexity.

tational burden derived from the set of multiple pairwise registrations. For instance, assuming 50 chunks (representatives of 500 frames, a more than reasonable value for our acquisitions), the combination produces $C(50) = \frac{50!}{2*(n-2)!} = 1225$ pairwise registration in total. Such an amount of coarse registrations take on average take $0.040\text{s} \times 1225 = 49$ s if computed sequentially. Obviously this is not feasible for our real-time constraint. Due to this limitation, we currently follow a *divide et impera* approach (Fig. 4.7): first, we set the maximum number of chunks to be equal to 50, then we separate them in 5 macro containers and we perform the $C(10) = \frac{10!}{2*(n-2)!} = 45$ pairwise registrations in each of them. Finally we aggregate the point clouds representing these containers and we repeat the multi-view alignment at macro level. In total we have $45 \times 5 + C(5) = 225 + 10 = 235$ registrations instead of 1225. Moreover, we exploit the nature of our networks by passing batches of 4 point clouds to extract the features in parallel, as well as batches of 2 set of correspondences to register in order to further reduce the computational impact. Overall, this can be put in a parallel thread so that after few seconds we get a refined model associated to those 50 chunks and in the meantime we can move forward with the scanning, starting a new volume. Following this approach, in principle we are able to recover a real-time compliant solution.

At the end of the reconstruction, we will then have a set of point clouds that we need to align. Once again we can leverage the graph optimization but also other methods, such as the work of Bonarrigo *et al.* [64], are valid. Actually, there is no scene in our dataset with more than 1500 frames, therefore the maximum number of macro chunks we get in the end is three. However,

usually this is not the case for other datasets, as for instance Redwood, that has scenes with almost 4000 frames. Nevertheless, we tackle this datasets by increasing the size of a regular chunk. In practice, a good value for chunk size with this kind of dataset is 20.

Overall, this approach remains effective but sub-optimal. We will discuss more about it in the results section, while we seek for future improvements to leverage on a more effective method to deal with the multi-view registration. The work of Gojcic *et al.* [18] is indeed a good candidate for the job but unfortunately the source code for the multi-view part has never been released.

2.1.3 Dynamic reconstruction

As we can see in Fig. 4.5, the output of the pose estimation now goes into a dynamic reconstruction module. Indeed, the poses which are currently estimated are temporary, and an optimized version could be found after the global evaluation. In order to refine the model after the pose optimization, we need to de-integrate and re-integrate the frames we want to adjust. As explained in BF, the volumetric integration via Truncated Signed Distance Function (TSDF) [62] is a symmetric operation which can be reversed. In fact, during the straight integration of a frame $\mathcal{F}_i$, each voxel v in the volume is updated by:

$$\begin{cases} \mathbf{D}'(v) = \frac{\mathbf{D}(v)\mathbf{W}(v) + w_i(v)d_i(v)}{\mathbf{W}(v) + w_i(v)} \\ \mathbf{W}'(v) = \mathbf{W}(v) \ + \ w_i(v) \end{cases} \tag{4.6}$$

where $\boldsymbol{D}(v)$ is the signed distance of the voxel, $\boldsymbol{W}(v)$ is the voxel weight, $d_i(v)$ is projective distance (along the z axis) between a voxel and $\mathcal{F}_i$, and $w_i(v)$ is the integration weight for a sample of $\mathcal{F}_i$. On the opposite, to update a voxel after removing such frame $\mathcal{F}_i$ it is possible to revert the sign of summation and change it to a subtraction:

$$\begin{cases} \mathbf{D}'(v) = \frac{\mathbf{D}(v)\mathbf{W}(v) - w_i(v)d_i(v)}{\mathbf{W}(v) - w_i(v)} \\ \mathbf{W}'(v) = \mathbf{W}(v) \ - \ w_i(v) \end{cases} \tag{4.7}$$

In practice, we adjust frames in the model only if the refined pose differs from the current by a significant margin, to ensure to minimize the amount of operations and to avoid burdening the user experience with unnecessary latencies.

2.2 Implementation

2.2.1 Source Code dependencies

We mentioned earlier that we exploited Open3D [55] to handle the 3D processing for all the steps in the pipeline. This is a very useful library, which is currently followed by a large community with interests in developing software for 3D data, and that is constantly improved and updated by the Intel

research group. With the release of version 0.11.0 at the end of 2020, a new engine has been added besides the legacy one, which offers CUDA capabilities to leverage GPU computation for fast parallel computing. Such an engine is based on open3D *tensor*, a data structure which is compatible with Numpy and PyTorch, two standard libraries for array processing. Once a stable version of the new engine was released in the library, I refactored the code to move most of the computation on GPU. This migration enhanced drastically the performance of the system (the gain was measured near 6× the CPU version). Such a boost offered the possibility to extend the working domain of our pipeline to an actual real-time 3D reconstruction. More details about the impact of this will be provided in the next section, when discussing about experimental timings.

Moreover, in the previous section we said that we need to de-integrate our data at runtime to refine the model. Actually, the Open3D data structure we use to define the TSDF Volume does not support the de-integration (up to the current version 14.1). Because of this, we had to implement by ourselves the method, by editing the open source code. We then wrote the function in C++ – both for the legacy[1] and for the new tensor-based data structure[2] – and then bind it to python.

Finally, we used MinkowskiEngine (ME) as backend for sparse CNNs operations, along with PyTorch as main framework for architectural implementation. Currently ME is the most unstable library of the pool, due to its very recent development. We migrated the code to the most recent version of the library 0.5 (released in December 2020) and we add a couple of fixes avoid useless data conversions from PyTorch to Numpy to speedup the processing. Unfortunately, ME does not support Windows at the moment and this is the main reason why we preferred to work under Linux.

The final product is a 99% Python code with an additional 1% of bash for scripting the tests, therefore a Docker container could be created for an easier distribution. A paper describing this last part of our work is under preparation and will include a release of our code.

2.2.2 Graphical User Interface

Earlier implementations of the pipeline were command line executables, optionally run via scripts. However, we aimed to move forward to a software design which is closer to a real 3D scanning application. At the same time, we needed a development visual tool to better understand the functioning and keep track, also visually, of the main internal state parameters and performance indicators of the method under development. Therefore we in-

[1] `http://www.open3d.org/docs/release/python_api/open3d.pipelines.integration.TSDFVolume.html`

[2] `http://www.open3d.org/docs/release/python_api/open3d.t.geometry.TSDFVoxelGrid.html`

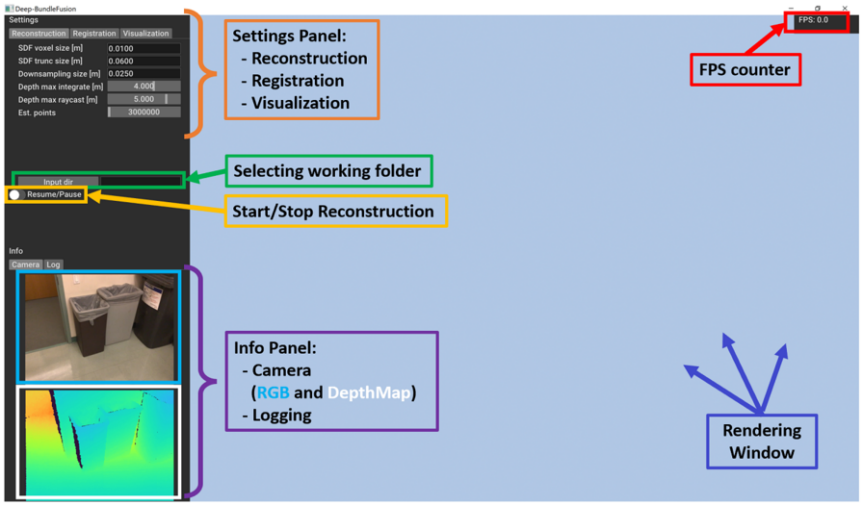


Figure 4.8: Our application GUI on startup.

troduced a Graphical User Interface (GUI) for our application. In Fig. 4.8 we show how it appears on startup. A panel on the left of the screen allows the user to modify on-the-fly some critical settings, such as the `voxel size`, the `fine sampling`, the registration method, etc. From the same panel we can specify the working directory containing the configuration files and run and stop the reconstruction using a toggle. In the lower part of the panel we have some useful information from logging and a live feedback of the camera frame, both RGB and Depth. The remainder of the windows is for rendering the 3D scene while it is reconstructed.

In Fig. 4.9 we show how a regular run looks like. The scene keeps growing while a yellow frustum representing the scanner moves in the field. We can also navigate the scene during the reconstruction in order to inspect the model from different views.

Finally, in Fig. 4.10 we show another feature for the reconstruction feedback. Each new fragment is depicted using a color scheme to indicate the *quality* of the alignment: based on the ICP fitness, the color will be closer to light green for higher fitness values and more reddish for lower values conversely. Ideally, in a real reconstruction scenario this kind of feedback can help the user understanding whether the system is struggling to reconstruct the model or not, and to adapt his/her movement consequently.

2.3 Experimental setup

DBF has been designed to simulate the working pipeline for our 3D scanner prototype, the InSight. Therefore, the main targets for the pipeline are the small and medium scale objects reconstruction. We already thoroughly discussed the key features of such an application scenario throughout the

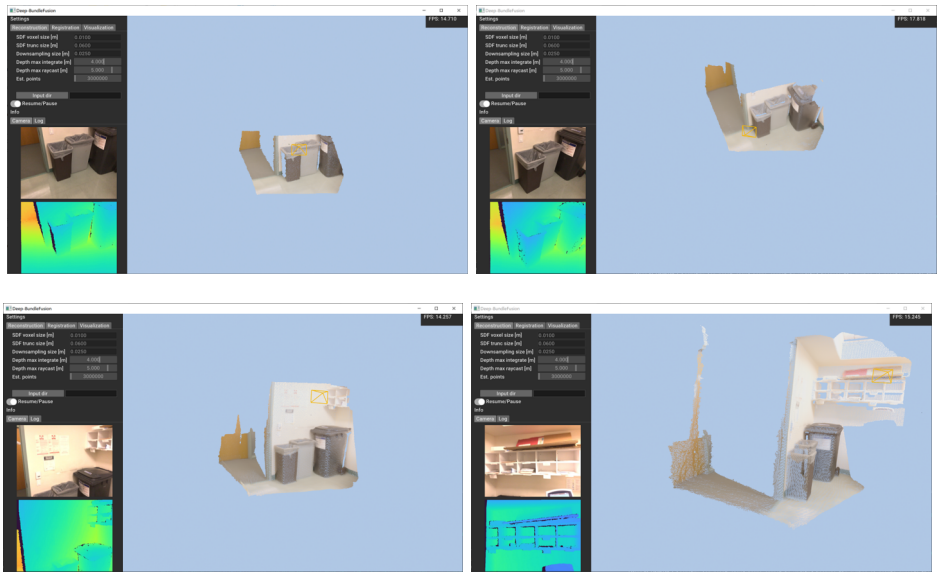


Figure 4.9: Example of reconstruction with our application. A yellow frustum shows the location of the camera in the 3D space. While the model is growing, we can move around to inspection the scene.

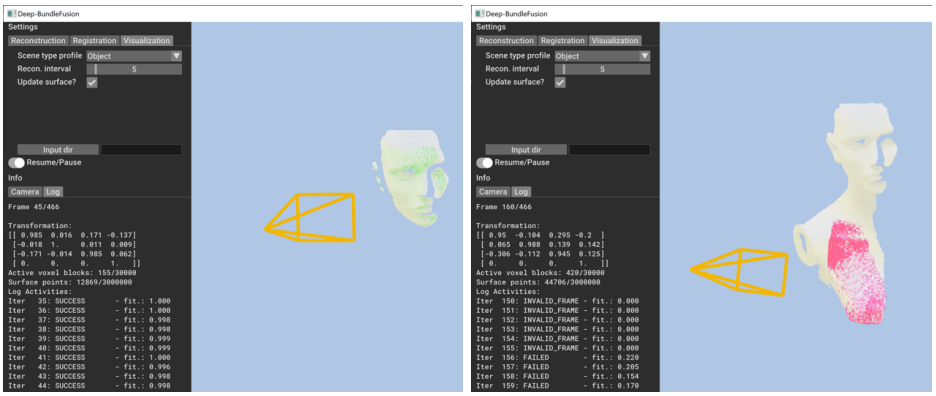


Figure 4.10: Example of reconstruction in our application. Detail of the color-coded representation of the currently framed area. When the color tends to the red, it means we are getting worse results for the registration.

entire document so we do not reiterate the discussion here. The reader can refer to Chapter 1 Sec. 2 for an introduction to the topic, to Chapter 2 Sec. 3 for more details on the type of data we produce with the InSight and finally to Chapter 3 Sec. 4.1 for an additional comparison of our dataset with 3DMatch, the indoor reconstruction dataset containing the data that was used to test BF originally.

Since we already know that BF struggles with our data, we decide to extend our tests to another publicly available 3D object dataset: the Redwood object collection [57]. Such a dataset is the closest we found to our DenseMatch, but it has been created by means of PrimeSense, a sensor which is very similar to the scanner that the author of BF used to test their own method. In the following paragraph we review very briefly the datasets we use to conduct our test. We then provide the details of the settings we chose for each dataset and we finally review the tests design for this experimental section.

2.3.1 Data

As anticipated, we use the two best datasets up to our knowledge to test an object reconstruction oriented pipeline: our DenseMatch dataset [J2] and the Redwood Object dataset [57].

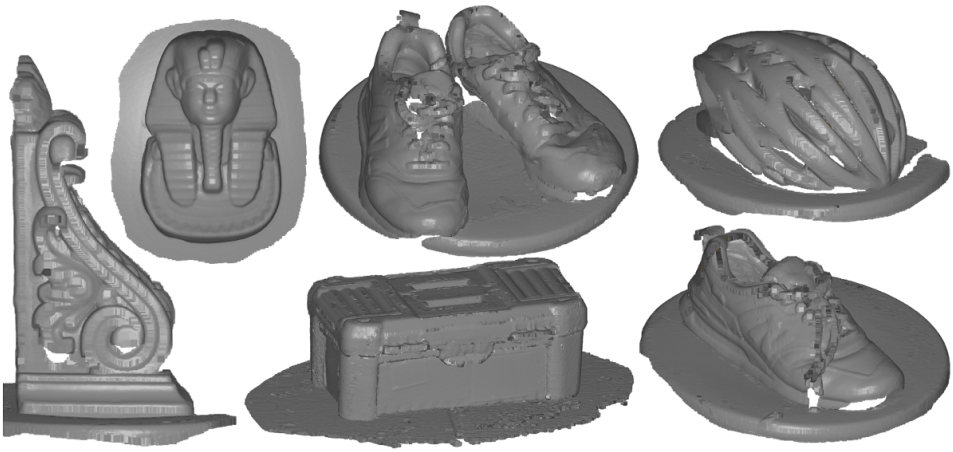


Figure 4.11: DenseMatch test scenes. From left to right: `column`, `statue`, `shoes`, `toolbox`, `helmet` and `shoes`.

Object reconstruction using the InSight: DenseMatch dataset Once again we evaluate the dataset we created using our prototype scanner. Actually, with respect to the original collection, we added few more scenes for having a set of fresh new tests. In Fig. 4.11 we show the scenes we evaluated. In particular, each scene comprises a set of independent acquisitions (from two to four) which we aggregated and passed to the pipeline as a whole. Similar to what we did in Sec. 1.2, we combine multiple sessions to automatically introduce some hard transitions to test the robustness of the method to tracking failures and to implement a real scenario in which the user interrupts the acquisition several times. In the end, a single session covers the object partially, but the whole covers (almost) the entire object

indeed. Therefore, often happens that we revisit the same spot multiple times under different vantage points and coming from different directions (loop closures)., closing loops.

Figure 4.12: DenseMatch test scenes. From left to right: `body`, `pig`, `angel`, `knight`, `gargoyle` and `lion`.

Object reconstruction using the PrimeSense (low-cost scanner): Redwood dataset Before introducing DenseMatch in Chapter 2, in Chapter 1 Sec. 2 we highlighted the lacking of a proper benchmark dataset solely focused on object reconstruction with high quality real scenes. In this context, the Redwood dataset [57] was the closest option, up to our knowledge. We already stated our concerns for this dataset when we introduced it in Chapter 1 and in fact we had a hard time to find any reliable sequence to use for our evaluation. We ended up hand-picking a bunch of scenes we considered valid for our test. In particular we explored the classes of *statues* and *toys*, which offered rigid scenes and a reasonable volume size and targets an application similar to what we addressed with the InSight. We depict such scenes in Fig. 4.12.

Unfortunately, the authors did not provide any ground truth for these models (what we reported in figure were our best results), therefore we can not leverage a reference for computing the deviation of the estimated surface. Nevertheless, we will see that a qualitative analysis is sufficient to evaluate the effectiveness of DBF and to compare it against BF.

2.3.2 Settings

In Tab. 4.4 we report the main settings we used to test the method, according to each benchmark dataset. We point out that even if application

target is similar for Redwood and DenseMatch, the parameters of the former overlap much more with 3DMatch (indoor reconstruction) instead, because of the similar scanner device configuration they share. The only meaningful differences between the two datasets are the maximum depth we use for data integration and voxel size of the volume. Indeed the scenes we picked for testing Redwood dataset do not require a depth of field higher than 2.0 m since we are scanning small and medium size objects. Such a constraint reduces the background noise and make the acquisitions closer to a standard approach for object reconstruction. Moreover, the size of the volume is subsequently much smaller for Redwood dataset. That is why we can use a smaller voxel-size and push the reconstruction accuracy to the upper limit of the original sensor. In 3DMatch scenes this value (*i.e.* 4 mm) is not feasible due to memory limitations, so we had to use a larger voxel. Finally, for all the settings that are in common with BF, such as the voxel size of the volume or the SDF truncation, we used a unique value, which is reported in Tab. 4.4 and marked with an asterisk. The remaining BF-specific parameters conversely, have been kept as we presented them in Chapter 2 2.3.

Name	Scope	Value			Note
		3DMatch	Redwood	DenseMatch	
`sdf voxel size`	Integration	1 cm	4 mm	1 mm	Resolution of the building volume
`sdf trunc`	Integration	6× `voxel size`			Truncation distance for TSDF ([62])
`z min`	Integration	0.3 m	0.4 m	0.5 m	Min. depth in RGB-D for volume integration
`z max`	Integration	4.0 m	2.0 m	0.7 m	Max. depth in RGB-D for volume integration
`z min rayc`	Ray Cast	0.3 m	0.4 m	0.5 m	Starting distance from camera for ray casting
`z max rayc`	Ray Cast	5.0 m		2.0 m	Maximum distance from camera for ray casting
`chunk size`	Ray Cast	20		10	Num. of frames to align before model update
`fine sampling`	Registration	2.5 cm	1.0 cm	2.5 mm	Sampling size to apply before registration
`coarse sampling`	Registration	5.0 cm	2.0 cm	5.0 mm	Sampling size to apply before registration
`icp dist`	Registration	1.5× `voxel size`			Max. dist. to consider two points as overlapping
τ_{fit}	Registration	30%	50%		Min. fitness for valid ICP (see Eq. 4.2)
τ_{fit_ratio}	Registration	75%			Min. fitness ratio of consec. frames (see Eq. 4.4))
τ_{rmse}	Registration	90%			Min. RMSE ratio for valid ICP (see Eq. 4.3))
τ_{rot}	Registration	15°			Max. rot. between consec. frames (see Eq. 4.5))
τ_{transl}	Registration	10 cm			Max. transl. between consec. frames (see Eq. 4.5)

Table 4.4: Settings used to run 3D reconstruction with DBF pipeline on respectively 3DMatch [6], Redwood [57] and DenseMatch [J2] datasets.

Hardware specifications All tests are performed on a Linux Ubuntu 18.04 machine with a NVIDIA Titan V video card, AMD Ryzen 1950x processor, and 64 GB of RAM.

2.4 Results

2.4.1 Tests design

We want to compare DBF with BF when employed for object reconstruction but at the same time we are interested to understand whether the renewed pipeline can offer additional advantages with respect to the first solution

solely implementing the safeguard mechanism. Therefore we first compare our method against BF by testing the two distinct collection of objects we presented above. This evaluation can only be qualitative with respect to the Redwood dataset because as we said we do not have a ground truth reference for it. In this extended evaluation we also consider the timings of our application, in order to understand whether it can fulfill the real-time requirement or not. Moreover, we add an ablation study on the pose optimization module to analyze its contribution. Finally, we review the most critical cases, together with some examples from the different working addressed by BundleFusion dataset, in order to build a discussion about the open issues and the future improvements we can seek.

2.4.2 Comparison with BundleFusion

A visual comparison of the reconstruction results obtained by using BF and DBF are reported in Figures 4.14 and 4.13, which refers to the DenseMatch and the Redwood datasets respectively. The first clear outcome from our evaluation is that DBF significantly outperformed BF on reconstructing the objects in both the test datasets.

Actually, this fact could be no surprise for DenseMatch evaluation: BF already struggled in the earlier evaluation we made in Chapter 2 Sec. 2.3 and we here we are finding confirmations on new data of the same kind. For instance, in the scenes representing one `shoe` and the pharaon `statue`, BF lost the track after few frames into the elaboration and then recovered it later in the sequence but badly, such that the reconstructed model comprises two distinct misaligned sub-portions. The `column` also was partially reconstructed, and presents evident misalignment in the middle. Marginally better is the result with two `shoes`: the additional object in the scene helped to disambiguate the matching so that no misalignments occurred. However, the tracking was lost again quite soon and the model is highly uncomplete. Another relatively good result is obtained with the `helmet`, but again clearly it is sub-par with respect to DBF reconstruction.

Then, likely even more interesting is to notice how DBF still performed better than BF also with the Redwood data. Overall, we observe that the DenseMatch dataset is more challenging for BF than the latter. Our assumption is that, differently from the InSight, the PrimeSense scanner has a relatively large field of view and it is used at a working distance of above 1 m most of the time. On average, these two aspects are together responsible for a larger area of the object surface which is framed at each shot, which favors the search of 2D correspondences across the views. For instance, `body`, `lion` and especially `angel` reconstructions are on-par with our result. Nevertheless, the method failed to reconstruct the model correctly in some of the Redwood scenes as well. In particular, the `gargoyle` and the `knight` scenes were very challenging for BF, which repeatedly lost the tracking, especially

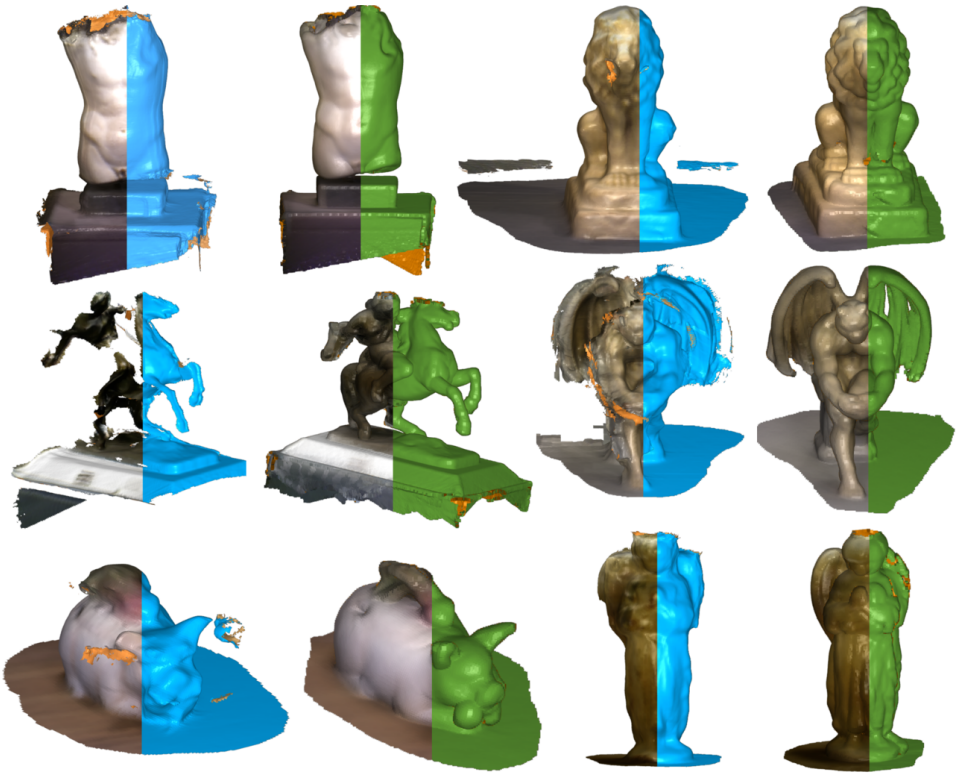


Figure 4.13: Redwood objects test scenes results. For all the reconstructed scenes both textured and shaded mesh is shown. In blue: results for BundleFusion. In green: results for Deep-BundleFusion.

when transitioning from the front of the statues to their back. In these cases the background is not helpful, since the statue stands in the middle of the room with nothing else within the working volume, whereas the surface is seen from a critical angle and does not provide sufficient information to the 2D-based tracking system.

In Fig. 4.15 we show two views of the `gargoyle` result to highlight how the back of the wings have not been reconstructed. We do the same for the right side of the `knight`, which is completely missing. This is critical for an handheld solution since it needs to be able to track the object even in a challenging situation such as the change of side. For the latter case we also show that the problem does not seem to be related to an unnecessarily strict threshold on the maximum reprojection error: indeed in Fig. 4.16 we report the result for varying this threshold and we show that there is no value we can use to solve the issues. Indeed, if a larger threshold is allowed, BF tends to align badly the frames and a lot of noise is introduced in the volume, up to the point that the reconstructed model is almost unrecognizable. This problem is due to the fact that it is easy to mistake a real corresponding

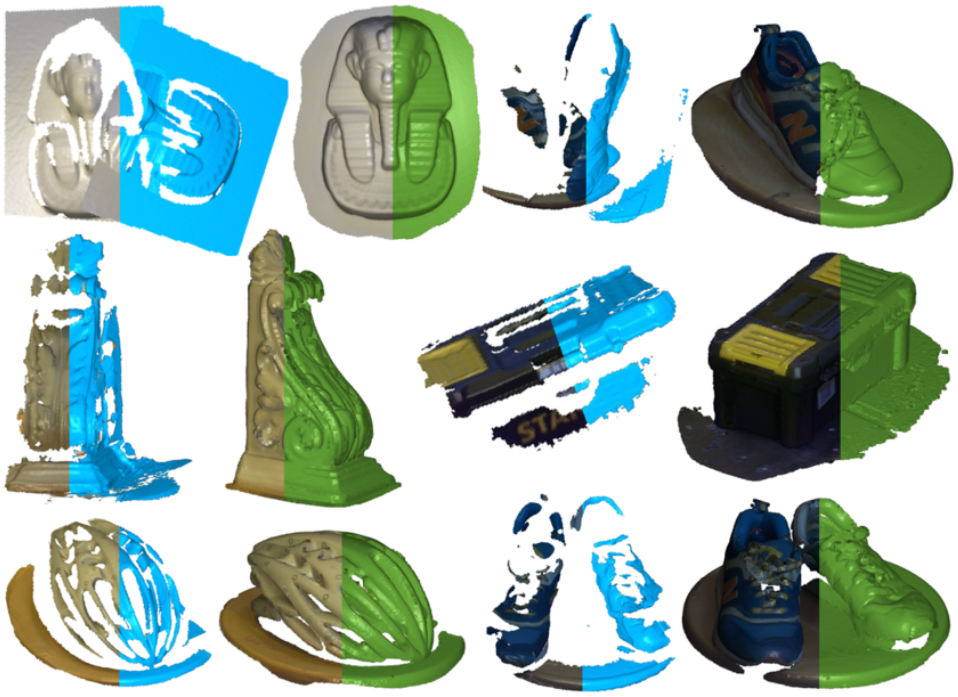


Figure 4.14: DenseMatch test scenes results. For all the reconstructed scenes both textured and shaded mesh is shown. In blue: results for BundleFusion. In green: results for Deep-BundleFusion.

3D point when matching sparse points extracted using SIFT on 2D images pair. Such a local misalignments is due to sensor noise and potential sub-optimal key point detection and in the end it can translate into significant displacement of points in 3D, with subsequent errors at registration level. On the contrary, DBF performed smoothly on these two scenes, and on the rest of the test set too (apart from a couple of noticeable cases that we will comment below).

Moreover, in Tab. 4.5 we report the number of frames integrated using the two methods, as well as the average distance of closest points between the estimated model and the ground truth for DenseMatch, as we already did in the first test we made in Sec. 1.3. Our method integrated a remarkable 98% of the total frames for DenseMatch and the 99.7% for Redwood. In contrast, BF scored 86.9% of integrated frames with Redwood and just 19.9% with DenseMatch. Finally, the average point-to-point distances between the reconstructed surfaces and by-hand aligned ground-truth ones shows a optimal result for our method, whereas in some cases for BF such result is heavily affected by the data partition due to the bad alignments.

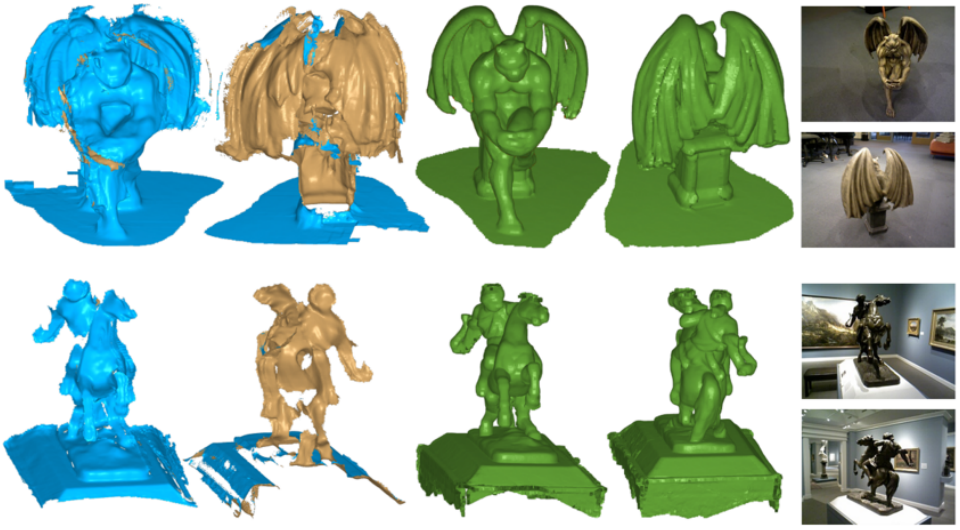


Figure 4.15: Scenes from Redwood dataset (`gargoyle` on top and `knight` at the bottom). In blue, reconstructed using BF: moving behind the statues the method lost the track. We show that the meshes are partial, and they miss one side of the model (front and back are shown). In green, reconstructed using DBF: also the back was recovered correctly, by rapidly recover the alignment using the safeguard module based on 3D geometry features.

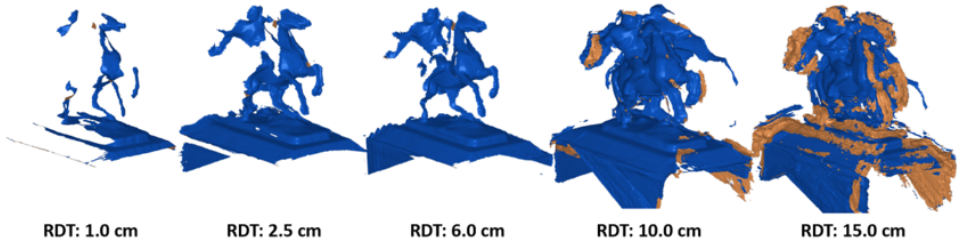


Figure 4.16: Results for 3D reconstruction of `knight` scene from Redwood dataset using BF and varying maximum reprojection distance thresholds (RDT).

2.4.3 Timings

In Tab. 4.6 we report the reconstruction time performance of the system we got on average with our data from DenseMatch. The result indicates that on average our system runs slightly below 20 fps if GPU support is on. In practice, we never got down under 10 fps with such a solution. When using CPU only, instead, the average reconstruction rate is near 7 fps, but it can also drop to 5 in worst situations. However, such cases are rare and the 7 fps average speed is a borderline yet acceptable frame rate for a real-time

DenseMatch [J2]						Redwood Objects [57]			
Scene	# Frames	Integrated		Dist [mm]		Scene	# Frames	Integrated	
		BF	DBF	BF	DBF			BF	DBF
column	1292	472	1256	1.6	0.2	angel	1542	1533	1540
helmet	460	258	452	0.2	0.1	body	863	833	863
shoe	1059	102	1041	fail	0.1	gargoyle	1856	900	1855
shoes	1007	149	976	0.8	0.2	knight	3885	3283	3875
statue	704	86	709	fail	0.0	lion	1281	1265	1271
toolbox	1504	134	1477	0.7	0.4	pig	2905	2905	2900
Tot.	6026	1201 (19.9%)	5911 (98.0%)			**Tot.**	12332	10719 (86.9%)	12304 (99.7%)

Table 4.5: Comparison between BF and our method, by evaluating DenseMatch and Redwood test scenes. We report the number of frames for each scene and the total frames integrated by each method (the percentage in brackets) We also add the average point-to-point distance between the reconstructed model and the ground-truth for DenseMatch, while we do not have a reference for the scenes we picked from Redwood.

application, even though it requires the user to move more carefully to not lose track too frequently. In contrast, the standard approach with GPU is full supported and it is reliable for a real-time application. Just for reference, BF demands two GPUs (physical or virtual) in order to start the processing. We tested it on a slightly different but yet powerful configuration using a Intel i7-7820HQ with 64Gb or RAM and a GPU NVIDIA Quadro P5000 with 16Gb of dedicated memory (split in 2 virtual GPUs with 8Gbs each) and we got a frame-rate near to 22 fps when running our data. An additional comment is for the coarse registration method: DGR remains faster than RANSAC (based on correspondences) even when running on CPU. Moreover, RANSAC has only the CPU timing reported since at the moment there is no available GPU implementation in Open3D. Although ICP has the two options, in our experiments we always had better results using legacy ICP (CPU-based), therefore, for our pipeline we preferred to keep the legacy version, which is remarkably fast anyhow. However, this probably indicates that we can still improve the GPU implementation and timings of ICP, and we consider this as a near future task.

2.4.4 Pose optimization contribution

Now we investigate the contribution of the pose optimization module in our method. We repeat the test removing this block from our pipeline, therefore we go back to the workflow we presented at the beginning of the chapter (see Fig. 4.2). In Fig. 4.17 we show a comparison of the two solutions for different examples on both datasets. It can be appreciated that in these particular cases, the optimization module helped to solve minor and major issues, depending on each scene. Surely the worst cases is the `helmet` scene that, similarly to what happened a couple of times with BF, was affected by a bad misalignment (probably due to the repetitive pattern on the surface of the helmet) which caused a partition of the data in two distinct sub-

Operation	Scope	Every Iteration?	CPU		GPU	
			T Avg [ms]	T Std [ms]	T Avg [ms]	T Std [ms]
Frame creation	Input	yes	42.1	5.6	24.4	3.2
Frame integration [62, 139]	Integration	yes	69.6	2.4	2.1	0.8
Fine reg (ICP pt2plane) [125]	Registration	yes	11.1	2.4	-	-
Ray casting	Model update	no	119.2	4.2	34.7	2.8
Feat extr (FCGF) [15]	Registration	no	183.2	5.2	35.7	4.6
Feature matching	Registration	no	37.5	1.0	9.9	1.1
Coarse reg (DGR w/o opt) [17]	Registration	no	12.8	1.1	2.4	0.7
Coarse reg (RANSAC) [114]	Registration	no	78.4	6.5	-	-
Total time for one iteration (fps in brackets {·})			150.1 {6.66}	8.5	55.2 {18.2}	6.3

Table 4.6: DBF timings with DenseMatch data. In the third column we highlight which operations are involved at each iteration. The total time additionally considers some miscellaneous which are not tracked specifically (for instance data transfer, chunks initialization, data sample and other minor computations).

models. In such a case, the optimization module saved the reconstruction by re-evaluating all the chunks globally and easily found and corrected the two partitions. The other errors are milder: the optimization module helped to refine the model by adjusting few frames that drifted throughout the alignment. Overall, the analysis shows that this module is helpful both to improve the quality and also to correct for some critical errors that can occur during the free-hand scanning.

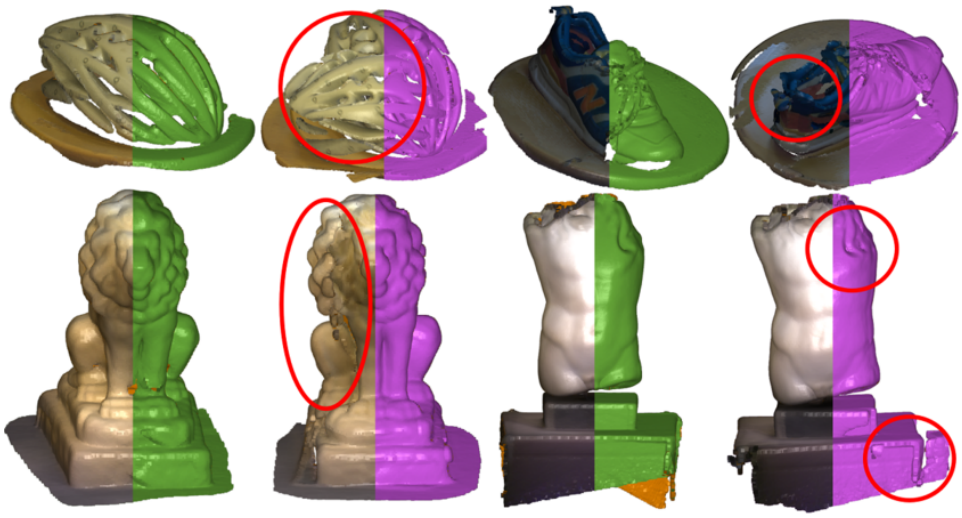


Figure 4.17: Results of DBF on different scenes from DenseMatch (top) and Redwood (bottom). Each model shows both the texture and a shaded uniform color. In green the results with pose optimization module active, in purple when it is turned off instead. In the red circles we aligned the reconstruction errors that occurred due to misalignemnt problems.

2.4.5 Critical cases

To conclude, we review the worst results we got using our method, in order to open a critical discussion about residual limitations and what it can be done better. In Fig. 4.18 we present a detail of the `angel` and the `toolbox` scenes, from Redwood and DenseMatch dataset respectively. The former estimated model shows that a misalignment affected the reconstruction. We can see how the angel has two faces, so it means that we have a partitioned set of frames that have been integrated with a drift with respect to the starting point. It is interesting to notice that this problem is almost entirely compensated if we remove the global adjustment. Indeed, this is the only case in which the optimization module worsen the reconstruction overall. We address this problem to be related to the chunks subdivision we performed to reduce the computational burden. A full exploitation of the whole set of pairwise alignments between all chunks should be more effective, however we are currently limited by the amount of computational time required, as discussed in Sec. 2.1.

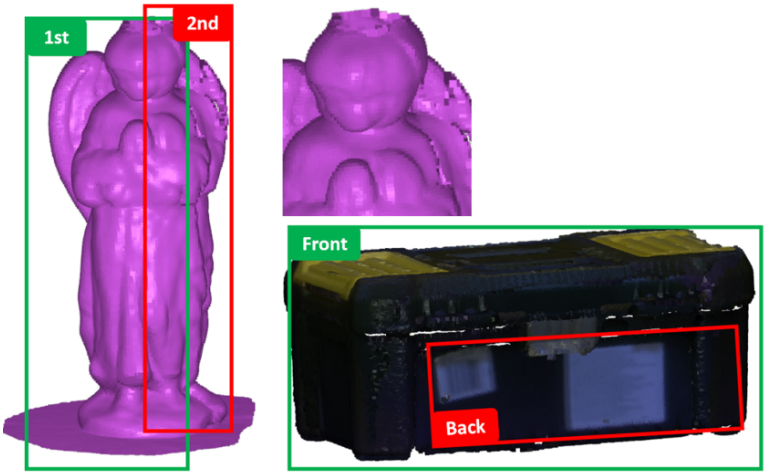


Figure 4.18: Examples of errors for DBF on `angel` and `toolbox` scenes. We highlight for the former case the two partitions of the model (two faces are visible) and the bad registration of the back of the box on the front side for the latter.

In the toolbox case instead, we have a problem of a different nature. The object has a shape close to a parallelepiped, it is almost perfectly symmetric, and we can distinguish the front from the back solely thanks to the locking system. During the scanning, we performed three distinct acquisition sessions to get the full set of view from all possible angles. In particular, we started from the front and we moved to the right, until we reached the end of the right short side, then we paused the system. After a while, we started

again scanning from a spot close to where we left, we went on the back of the box and we stopped again. We finally completed the acquisition with a third session, moving from the back to the front from the left side of the box. In Fig. 4.19 we review these steps and we highlight the problem we faced. During the transition from the first session to the second, we temporarily lose the track and we ask the safeguard module to help to retrieve a coarse estimation of the camera pose. However, the registration is misled by the symmetry of the object and then the back of the toolbox (which has never been seen before) has been aligned with the front. The difference between the two point clouds is marginal and our failure detector was deceived.

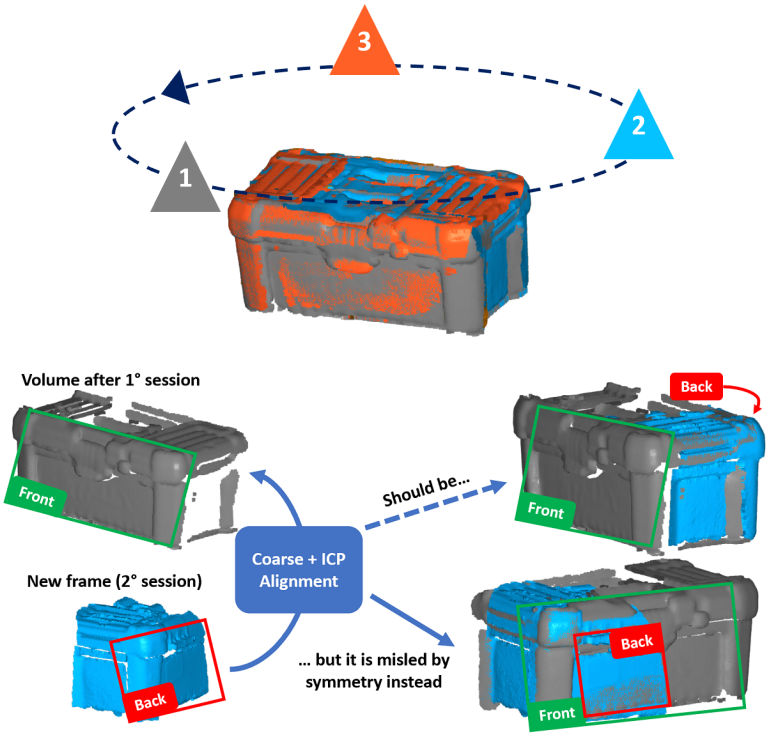


Figure 4.19: Detail on the error occurred with the `toolbox` scene. (top) we show the path we follow to acquire the model (split in 3 distinct sessions). (bottom) the new point cloud from the second session is framing the back of the box (not reconstructed yet), which is wrongly aligned with the front side that we previously reconstructed.

2.4.6 Evaluation on indoor reconstruction dataset

Finally, we add a last comparison by evaluating how our method performs in a different application context. We then try to reconstruct a scene from BF dataset [19]. This is an indoor 3D reconstruction task, which is mainly

characterized by a longer sequence of frames and spans a much larger volume with respect to our previous tests. In Fig. 4.20 we show our result compared

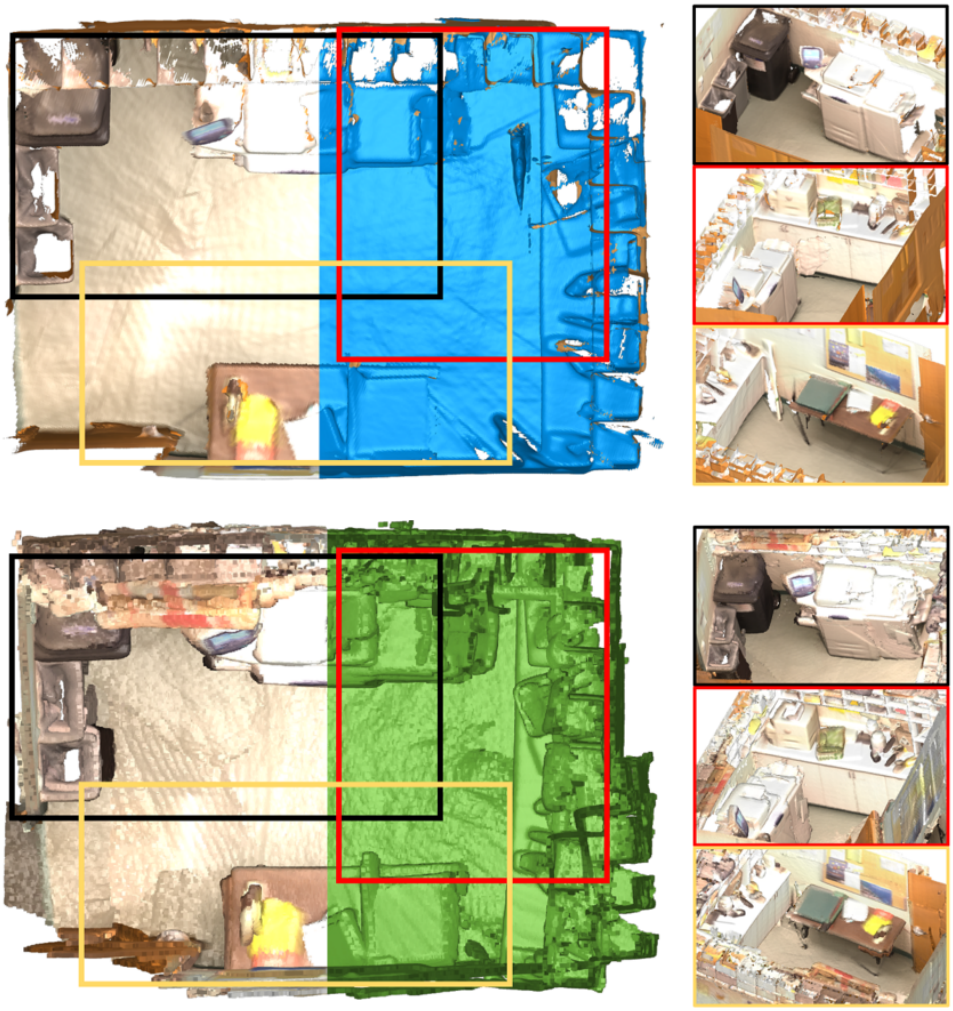


Figure 4.20: Reconstruction result for `copyroom` scene in BundleFusion dataset [19]. (top): reconstructed with original method by BF. (bottom): reconstructed with DBF.

with the original model. From the figure, it is clear that BF is particularly well suited for this kind of task: being able to compare a large set of frames all together and simultaneously, using a lightweight 2D feature-based solution allows to achieve a very robust result. The accuracy of the reconstruction, conversely is sacrificed a little, since we had to use 1 cm voxel size in order to run the code which crashed otherwise. Indeed, such a voxel size does not fully exploit the spatial resolution that can be offered by the scanner

they used, namely the Structure by Occipital (in Chapter 1 Sec. 1.3 we reported the main specifications). Nevertheless, the result is remarkably good for several application fields. On the contrary, in this case, our method under-performed. The main issue we encountered was due to the lack of a full extensive evaluation of the whole set of chunks, which conversely we need to split in order to keep the performance of the system within the requirement of a real-time application. Indeed, our system is on par with BF at local level and it actually allows to use even a higher resolution for the final volume since it does not need to keep all the frames stored in memory. In Fig. 4.21 we report some sub-parts of few scenes we reconstructed using DBF and a voxel size of 4 mm. The result of these reconstruction is good and in theory we can use multiple sub-parts to re-align the final model in post processing. However, the real-time processing for this kind of scene

Figure 4.21: Examples of indoor reconstruction. Partial reconstruction (both textured and shaded mesh) of scenes `copyroom`, `apt0` and `office3` from BundleFusion dataset using DBF.

is still under analysis for our method, which is currently limited by the computational time it requires to process such a high amount of chunks simultaneously. Many ideas are in place, starting with the creation of a feature table to leverage a configuration which is similar to what Bonarrigo *et al.* proposed in [64]. Another interesting research topic is certainly the investigation of a more efficient way to compute the multi-view registration. In particular, in the future we would like to evaluate whether we can extend the DL-based pairwise registration to a multi-view application, in order to leverage once again the robustness of a data-driven method along with the high level of parallel computation it can offer on modern GPUs.

3 Conclusion

In this chapter we presented a novel framework to address the problem of real-time 3D reconstruction by focusing on the application of a high resolution handheld optical scanner (the InSight). We started from the considera-

tion that the pipeline that was originally designed for the InSight was affected by a series of issues: in particular it was lacking both a robust method for fast and accurate camera tracking recovery and a solution to refine the model on-the-fly, in order to deal with drift and sporadic misalignments. Therefore we tried to exploit the novel DL-based solutions that have been objective of a deep analysis in the Chapter 3. The first attempt we made was focused on designing a safeguard module to address the problem of tracking loss. We then developed a solution which uses data-driven methods to register a new frame to the model. The method starts with FCGF to extract the geometric features from the two set of points and then it feeds DGR model with the putative correspondences we get from the feature matching, in order to infer the coarse registration. We tested the method with DenseMatch dataset and we show how the new framework is robust to several cases of tracking. Moreover we found that the DL based solution maintains sustainable the computational overhead of the safeguard module such that we are able to use it in a real-time workflow. Finally, we focused on the possibility to refine dynamically our model in real-time. We proposed a solution that we called Deep-BundleFusion which offers the same powerful camera relocation module based on DL and, in addition, it allows to globally optimize over the frames in the sequence by exploiting a hierarchical data structure and by performing a multi-view registration over a collection of frames aggregated into chunks. The resulting pipeline showed remarkable results for an object reconstruction pipeline and it outperformed the previous methods, including the state-of-the-art in real-time reconstruction BundleFusion, when evaluated on the DenseMatch and Redwood 3D object datasets.

Conclusion

At the beginning of the work, our goal was to revise the 3D reconstruction workflow of the InSight, an handheld optical scanner which suffered from the irreversible loss of tracking and it was somehow difficult to use due to the suboptimal user experience it offered overall.

Although the target was very specific, we soon realized that the issues affecting the considered instrument are not limited to this specific device but they extend to the entire category of handheld optical scanners, which is a large and growing family that is getting more relevant every day because of the advantages it can offer in such a digital-centric world. Moreover, despite the increasing interest this kind of solution is getting, we found the literature to be still lacking of a well centered analysis regarding typical data and requirements of handheld object scanning, being more focused on a related but different set of applications and low-cost devices. The 3D reconstruction requirements we target are increasingly relevant nowadays, they refer to common needs shared across several applications (such as reverse engineering, quality inspection and accurate 3D modeling) declined into several domains (such as entertainment, industrial, cultural, and bio-medical). However, they remain challenging with respect to the available solutions. For instance, the medical sector is seeking for an accurate, comfortable to use and affordable device to address the task of human body scanning for orthotics related applications and the prototype of scanner we are designing wants to embrace such needs to fill some gaps existing in the literature and in the market.

Our early work involved the analysis of the state-of-the-art in different ways to approach real-time 3D reconstructions, in order to leverage on standard handcrafted solutions to improve critical parts of the original pipeline, such as the key frame update policy. However, while these initial attempts were non conclusive to solve all the issues, they helped us to better understand the nature of the challenges we had. Then, at the end of this preliminary evaluation, we restricted the search of solutions around two key concepts: improving the robustness of the local registration and offering an effective global refinement to adjust the model on-the-fly.

At this stage of the research, we moved our attention toward novel solutions, which are based on the most recent and appealing technology in the field of Computer Vision, *i.e.* Deep Learning. Indeed, as my PhD

work progressed, the research community was proposing novel techniques to build complex and effective models to deal with 3D processing-related tasksng some interesting works on 3D viwe registration. A pool of similar and promising solutions was presented at the same conference (CVPR 2020) during the second year of my PhD and since then we started analyzing, implementing, revising, comparing them and, ultimately, we tried to adapt them to our needs.

In order to analyze and to assess these new methods, we created and shared our own 3D object dataset, which is meant to provide a reference to the researchers that need to work with real and high-quality data in the field of object reconstruction. Then, we made the first cross-domain assessment of these contemporary methods and we finally proposed a first configuration that was suitable for real-time applications.

Grounding the result of our experiments, we finally conceived a new design of a 3D reconstruction pipeline suitable to the InSight, as well as to scanners producing similar data flows. At its core, a DL-based module guarantees a quick and robust 3D registration thanks to its capability to achieve a fast and effective camera relocation, even in complex scenarios. Moreover, the DL-based module is also fundamental to provide reliable registrations into a pose optimization block, which builds a pose graph to adjust the poses and to refine the models by means of a dynamic reconstruction approach.

Our final product is a software with a graphical user interface which wants to reproduce a real 3D scanning application. We will soon be working on a final scientific contribution whereby releasing an open repository of the software, along with an extended version of our dataset – DenseMatch – in order to provide an easy-to-use framework for fast and robust 3D reconstruction to the research community. Our plan is to initially make it possible the use of the reconstruction software on post-acquisition sequenced of common RGB-D frames, but later we aim to leverage the Open3D library to extend the compatibility to the drivers of Intel scanners. In this way, we intend to provide an alternative solution for real-time scanning which can exploit DL-based methods for robust reconstruction.

In conclusion, although the InSight is still a prototype and it is not sure if it will be released or not, the study of this device have offered us many *insights* to understand real and complex problems that we tried to address, hoping that our solutions can be helpful to the largest possible research community.

List of publications

Peer-reviewed journals

[J1] **M. Lombardi**, M. Savardi, A. Signoroni; "Cross-domain assessment of deep learning-based alignment solutions for real-time 3d reconstructions", *Computer & Graphics Journal*, vol. 39, 2021 doi: `https://doi.org/10.1016/j.cag.2021.06.011`.

[J2] **M. Lombardi**, M. Savardi, A. Signoroni; "Densematch: a dataset for real-time 3d reconstruction", *Computer & Graphics Data in Brief*, vol. 99, 2021 doi: `https://doi.org/10.1016/j.dib.2021.107476`.

Peer reviewed conferences with proceedings

[C1] **M. Lombardi**, A. Riccardi, A. Signoroni; "Increasing tracking robustness for low-complexity real-time reconstruction with handheld optical scanners", *ISPRS Optical 3D Metrology workshop*, 2020 doi: `https://doi.org/10.5194/isprs-archives-XLII-2-W18-115-2019`.

[C2] **M. Lombardi**, M. Savardi, A. Riccardi, A. Signoroni; "3DReg-i-Net: an improved 3D deep learning registration network for robust real-time scan alignment", *STAG*, 2019 doi: `https://doi.org/10.2312/stag.20191363`.

[C3] **M. Lombardi**, M. Savardi, A. Signoroni; "Deep-learning Alignment for Handheld 3D Acquisitions: A new Dense-match Dataset for an Extended Comparison", *STAG*, 2020 doi: `https://doi.org/10.2312/stag.20201244`.

Appendices

A Stereo Vision

Here, we briefly review the elements of 3D reconstruction from images in order to provide context to discussion we made throughout the entire work. Such a review does not mean to be exhaustive at all, therefore we suggest the reader to refer to the books [30, 211] for a more complete reference.

A.1 Camera calibration

Our starting point is an optical sensor (*e.g.* a digital camera) which can be used to the describe the geometry of the world in a relative coordinate system by means of a *projective* model. Given a 3D point $\boldsymbol{X}_{\mathcal{C}} = \{x_{\mathcal{C}}, y_{\mathcal{C}}, z_{\mathcal{C}}\} \in \mathbb{R}^3$, where $\mathcal{C}$ indicates that $\boldsymbol{X}$ is in the camera coordinate system, this can be mapped to a 2D point $\boldsymbol{x}$ onto a plane, *i.e.* a pixel into an image. The 2D point can be represented using the homogeneous coordinate system, such that $\boldsymbol{x} = \{u, v, 1\} \in \mathbb{Z}^2$. Then we can define $\boldsymbol{K} \in \mathbb{R}^{4\times3}$ as the matrix that maps $\boldsymbol{X}_{\mathcal{C}}$ to $\boldsymbol{x}$:

$$\boldsymbol{x} = \boldsymbol{K}\boldsymbol{X}_{\mathcal{C}} \tag{1}$$

In the ideal condition of having a *pinehole* camera model (*i.e.* the light rays travel in straight lines and the lenses are either distortion-free or compensated), $\boldsymbol{K}$ can be defined as:

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & p_u \\ 0 & f_y & p_v \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where f, s and p are the focal length, a scale factor and the principal point of the camera respectively. In general, estimating these parameters is referred to the process of *internal calibration*. Moreover, in order to map a point from the global reference system of the world $\mathcal{W}$ into the local reference system of the camera $\mathcal{C} \in \mathbb{R}^{3\times4}$, it is necessary also to perform an *external calibration*. The extrinsic parameters composing the external calibration matrix $\boldsymbol{C}$ basically describe the pose of the camera in 3D world. The model

is defined by means of a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$, such that

$$\boldsymbol{C} = [\boldsymbol{R}|\boldsymbol{t}] \tag{3}$$

where $\boldsymbol{C}$ maps a homogeneous 3D point in the world $\boldsymbol{X}_{\mathcal{W}}$ into a homogeneous 3D point in the camera system $\boldsymbol{X}_{\mathcal{C}}$:

$$\boldsymbol{X}_{\mathcal{C}} = \boldsymbol{C}\boldsymbol{X}_{\mathcal{W}} \tag{4}$$

By combining the internal and the external matrices we finally get the *projection matrix* $\boldsymbol{P}$:

$$\boldsymbol{x} = \boldsymbol{K}\boldsymbol{C}\boldsymbol{X}_{\mathcal{W}} = \boldsymbol{P}\boldsymbol{X}_{\mathcal{W}} \tag{5}$$

We started by assuming to have the ideal configuration of a pinhole camera, which is not affected by distortions. In practice, the lens distortion is always critical and so we need to estimate it through camera calibration. In literature we can find many proposal for camera calibration, either by computer vision and photogrammetry researchers. Some popular examples are the plane reference calibration by Zhang [212], the 3D reference object-based calibration by Faugeras [213], the self-calibration from Fraser and from Gruen [214, 215] and the calibration using vanishing point for orthogonal direction bu Liebowitz *et al.* [216].

A.2 Range image generation

A rigid pair of sensors can be used to define a stereoscopic system [30]. At its core, we have two cameras looking at the same point in 3D space. If we know how to calibrate these cameras, then by exploiting the positions in 2D on the two image planes of such a point we can retrieve its distance from the system. In general, mapping each pixel of the image plane with the additional information of depth returns a so-called Depth Map. In practice, many algorithms have been developed through the years in order to compute depth maps from stereo images [217]. Actually, most of them do not compute depth directly but rather via a preliminary dense *disparity* field estimation. With the term disparity we mean here the horizontal displacement of the pixels from the two images. Indeed, depth z and disparity d are strictly related in such a stereo configuration. Specifically, the depth is inversely proportional to disparity:

$$z = \frac{bf}{d} \tag{6}$$

Here b and f represents the baseline (*i.e.* the distance between the two cameras) and the focal length (assuming they are the same for both the cameras). We refer for such a configuration to Fig. 1.

However, computing the disparity it is not straightforward since it means to solve the correspondence points problem first. This is a fundamental problem in Computer Vision, which for stereo matching has been tackled by

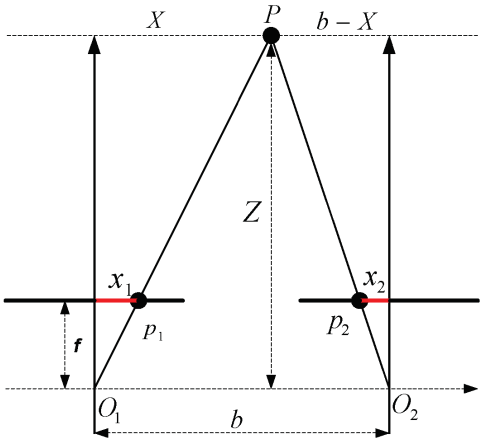


Figure 1: Stereo vision of a point P which is seen from a pair of cameras centered in O_1 and O_2 on a baseline b, having focal length f. Image taken from [218].

using the *epipolar geometry*. Referring to Fig. 2 we see that the 3D point $\boldsymbol{M}$ is mapped on the two image planes in $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ for the left and the right camera respectively. The geometric relationship between these two point is given by the Fundamental matrix [30] $\boldsymbol{F} \in \mathbb{R}^{3\times3}$:

$$\boldsymbol{m}_2^{\mathrm{T}} \boldsymbol{F} \boldsymbol{m}_1 = 0 \tag{7}$$

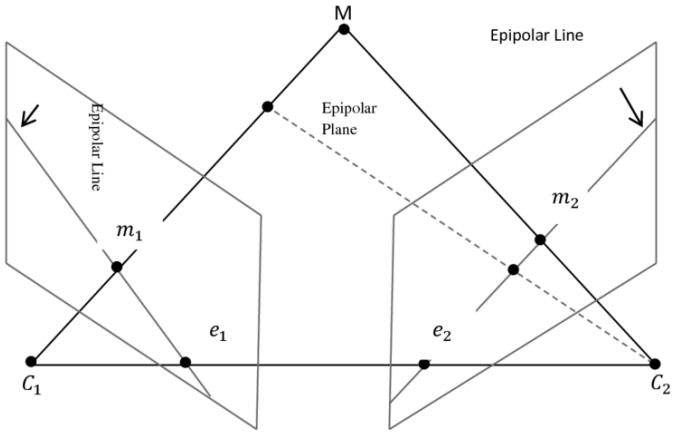


Figure 2: Epipolar geometry elements for a stereo system. The two camera centers $\mathcal{C}_1$ and $\mathcal{C}_2$ and the world point $\boldsymbol{M}$ form the epipolar plane. The epipolar line $l_2 = \overline{\boldsymbol{e}_2\boldsymbol{m}_2}$ defined by the image point $\boldsymbol{m}_1$ is given by $l_2 = \boldsymbol{F}\boldsymbol{m}_1$ while the epipole $\boldsymbol{e}$ is the projection of the camera center $\mathcal{C}_2$ into the first image. The epipolar constraint is then $\boldsymbol{m}_2^{\mathrm{T}}\boldsymbol{F}\boldsymbol{m}_1 = 0$. Image from [219].

Therefore, in a calibrated stereo configuration it is possible to apply the

Fundamental matrix to reduce the search space for a corresponding point $\boldsymbol{m}_2$ in right image with respect to $\boldsymbol{m}_1$ in left image to lie on the epipolar line.

B Deep Learning

In this section we want to give a gentle introduction to Deep Learning (DL). Again, this is meant just to give context to what we discussed in the manuscript, but for extender references we refer the reader to [220] and to the book [221].

In general, a neural network (NN) can be seen as a complex structure comprising a multitude of layers that receive input data at an entry point and produce the expected outcome at the output. The relationship between input and output is in general complex and nonlinear by construction. The rationale is that most of the problems we want to solve have an inherently complex nature, therefore it is necessary to exploit a complex solution.

In order to understand how to solve complex problems, the network has to *learn* how to do it. Learning is a very complex task itself, that is fulfilled by means of training.

B.1 Training a model

In order to react to a new information and then, in another way, *to learn* from a new example, each layer in the architecture has to be differentiable. Usually the learning policy most used in DL is *gradient descent with back-propagation*. Such a policy aim to tune the parameters of the network such that the objective of a *loss function* is minimized. Such a loss function is a key element of DL to measure the quality of the proposed output in a differentiable manner. In order to minimize its objective, an optimizer is then leveraged by the system. It is clear then that the choice of a proper cost function is a fundamental step in the process of network design. In this manuscript, we mentioned for instance the cross-entropy function as designed loss to learn how to classify an input data. In practice, in order to optimize the loss it can be used the principle of maximum likelihood. The cross-entropy function can then be described as:

$$\mathcal{L}(\theta) = -\mathbb{E}[\log p_{model}(y|x)] \tag{8}$$

Where, θ are the parameters we want to estimate (such that the cost function is as lower as possible), x is the input, y is the expected output, and p_{model} is the model distribution. Overall, the design process needs to account for several aspect, regarding the reliability of the source input, the level of difficulty of the task to learn, the putative contamination of outliers, etc..

As we said, the loss minimization goes through the gradient descent which exploits the first derivative of the function to find the best direction towards

a minimum. The size of the steps used to descend are expressed by the learning rate. In general there is no guarantee of reach a global minimum. Nevertheless, the use of specific layers as *batch normalization* and *dropout* in conjunction with gradient descent, produces a fairly robust training procedure.

B.2 Elements

Finally we review the elements of DL by introducing the most important type of data structure to us, namely the feed-forward NN. Such a network is so-called because of the method it uses to learn, *i.e.* to feed the layers and adjust the values of the internal parameters throughout training. Many of these type of NN have been proposed through the years [221]. We now review the two key ones, which are also recurrently adopted in the solutions we discussed in the thesis.

B.2.1 Fully-connected

In a *fully-connected* (FC) network (or a set of layers of a network) each node of a layer is connected to all the nodes in the following one. In this, each neuron sums the all the inputs it receives from the previous layer and eventually adds a non-linear activation function – typically a Rectified Linear Unit, (ReLu) – to produce its output (see Fig. 3). FC is the most trivial form of NN and it is known to be sensitive to the burden of complexity, due to the explosion of parameters that grows exponentially with network increasing depth.

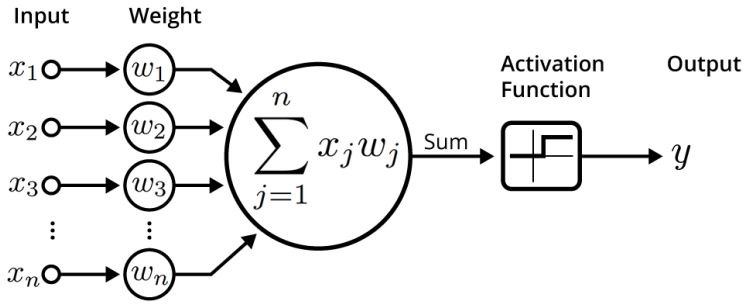


Figure 3: Example of feed forward NN with fully connected layers. Source: `https://becominghuman.ai/`.

B.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) [168] are particular types of deep feed-forward networks which gained a lot of success since its introduction due to the effectiveness, the ease of training and the lower memory consumption with respect to FC layers. Such properties are derived by the number of parameters reduction that occurs due to the introduction of spatial constraints in the hypothesis space. The constraints is produced by local connections, shared weights and the use of pooling layers. The combination of these elements enforces a structure which creates features that are spatially invariant and robust to rotation and deformations.

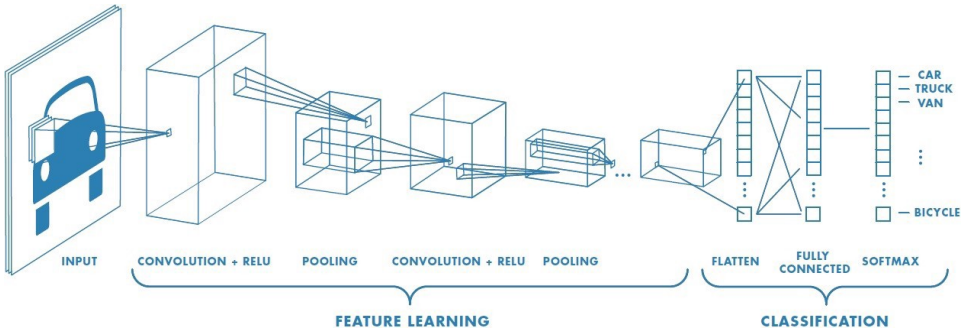


Figure 4: Example of 2D CNN on image for object classification. Source: `https://towardsdatascience.com/`.

In origin, CNNs were designed to process 2D matrices or tensors, especially referring to color images (Fig. 4). However, many other data sources are available to be processed with CNNs. The only requirement is that the input data needs to have a regular structure. In 3D, we can see this as a volumetric CNN with 3D kernels spanning over a grid of voxels (Fig. 5).

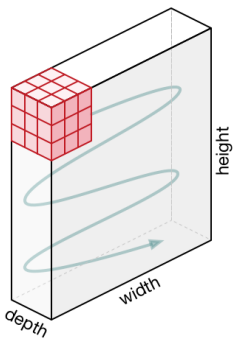


Figure 5: Example of 3D CNN on volumetric grid. The 3D kernel strides in 3 direction. The remainder of processing and elements is identical to 2D. Source: `https://towardsdatascience.com/`.

C Glossary and Acronyms

C.1 General

AI Artificial intelligence.

DL Deep Learning.

NN Neural Network.

CNN a Convolutional Neural Network is a DL model which uses convolution layers in it.

MLP a Multi Layer Perceptron is a DL model which uses full connected layers in it.

t-SNE is a non-linear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space. It models each high-dimensional object so that objects are projected trying to keep their distance with high probability.

C.2 3D Reconstruction

RI Range Image. Is a 2D image with additional 3D coordinate information associated to each pixel.

PC Point Cloud.

Voxel A 3D pixel defined over a structured volumetric grid.

TSDF Truncated Signed Distance Function is used for a parametric definition of the volume, in which a surface is derived from a structured voxel grid by extracting the zero-crossing of the TSDF from each voxel.

KF KinectFusion.

BF BundleFusion.

C.3 Metrics

False Negative (FN) the number of wrongly predicted real negative cases in the data.

False Positive (FP) the number of wrongly predicted real positive cases in the data.

True Positive (TP) the number of correctly predicted real positive cases in the data.

True Negative (TN) the number of correctly predicted real negative cases in the data.

Accuracy measures the proportion of correctly classified elements.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{9}$$

Precision measures the proportion of actual positives with respect to the

predicted positives.

$$Precision = \frac{tp}{tp + fp} \tag{10}$$

Recall or Sensitivity measures the proportion of actual positives that are correctly identified as such.

$$Recall = \frac{tp}{tp + fn} \tag{11}$$

Feature Match Recall (FMR) measures the proportion of actual positives results, evaluated as the correct identification of a minimum percentage τ_2 of matches that are correctly detected as such in a set of Ω_s correspondences. A match is defined positive if the distance between the associated points is below a threshold τ_1.

$$\text{FMR} = \frac{1}{M} \sum_{s=1}^{M} \left(\mathbb{1} \left[\left(\frac{1}{|\Omega_s|} \sum_{(i,j) \in \Omega_s} \mathbb{1}(\quad \hat{\mathbf{T}}\mathbf{x}_i - \mathbf{y}_j \quad < \tau_1) \right] > \tau_2 \right) \right(\tag{12}$$

Rotation Error (RE) measures the distance in degrees between a ground truth rotation matrix and an estimated version of it.

$$\text{RE} = \cos^{-1} \quad \frac{\text{trace}(\hat{\mathbf{R}}^{-1}\mathbf{R}_{gt}) - 1}{2} \right) \left(\tag{13}$$

Translation Error (TE) measures the distance in meters between a ground truth translation vector and an estimated version of it.

$$\text{TE} = \quad \mathbf{t}_{gt} - \hat{\mathbf{t}} \tag{14}$$

Bibliography

[1] M. Zollhöfer, P. Stotko, A. Gorlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the art on 3d reconstruction with rgb-d cameras," *Computer Graphics Forum*, vol. 37, pp. 625–652, 05 2018.

[2] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 945–953.

[3] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3d shape segmentation with projective convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6630–6639.

[4] H. Huang, E. Kalogerakis, S. Chaudhuri, D. Ceylan, V. G. Kim, and E. Yumer, "Learning local shape descriptors from part correspondences with multiview convolutional networks," *ACM Trans. Graph.*, vol. 37, no. 1, nov 2017. [Online]. Available: https://doi.org/10.1145/3137609

[5] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji, and R. Wang, "3d shape reconstruction from sketches via multi-view convolutional networks," in *2017 International Conference on 3D Vision (3DV)*, 2017, pp. 67–77.

[6] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[7] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[8] G. Riegler, A. O. Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6620–6629.

[9] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017, pp. 77–85.

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

[11] H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6410–6419.

[12] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 3070–3079.

[13] S. Soltanpour, B. Boufama, and Q. Jonathan Wu, "A survey of local feature methods for 3d face recognition," *Pattern Recognition*, vol. 72, pp. 391–406, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317303072

[14] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C. L. Tai, "D3feat: Joint learning of dense detection and description of 3d local features," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6358–6366.

[15] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, nov 2019, pp. 8957–8965.

[16] G. D. Pais, S. Ramalingam, V. M. Govindu, J. C. Nascimento, R. Chellappa, and P. Miraldo, "3dregnet: A deep neural network for 3d point registration," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7191–7201.

[17] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2511–2520.

[18] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal, "Learning multiview 3d point cloud registration," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1756–1766.

[19] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Trans. Graph.*, vol. 36, no. 3, May 2017.

[20] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[21] R. Gherardi, M. Farenzena, and A. Fusiello, "Improving the efficiency of hierarchical structure-and-motion," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1594–1600.

[22] E. Trucco, A. Fusiello, and V. Roberto, "Robust motion and correspondence of noisy 3-d point sets with missing data," *Pattern Recognition Letters*, vol. 20, no. 9, pp. 889–898, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865599000550

[23] D. Vangi, F. Begani, R. Toldo, and A. Fusiello, "Photogrammetry 3d vehicle reconstruction for energy loss evaluation," in *23rd Annual Congress of the European Association for Accident Research and Analysis*, Copenhagen - Denmark, 2014. [Online]. Available: https://www.evuonline.org/index.php/publikationen?lang=da&option=com_content&view=article&id=1167

[24] M. Javaid, A. Haleem, R. Pratap Singh, and R. Suman, "Industrial perspectives of 3d scanning: Features, roles and it's analytical applications," *Sensors International*, vol. 2, p. 100114, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666351121000358

[25] A. Yao, "Applications of 3d scanning and reverse engineering techniques for quality control of quick response products," *The International Journal of Advanced Manufacturing Technology*, vol. 26, pp. 1284–1288, 11 2005.

[26] A. Negrente, U. Castellani, A. Fusiello, and V. Murino, "Model acquisition with a free-hand laser scanner," 2003, poster presented at the IEEE International Workshop on Projector-Camera Systems.

[27] G. F. Marshall and G. E. Stutz, *Handbook of optical and laser scanning*. Taylor & Francis, 2012.

[28] N. Pfeifer and C. Briese, "Laser scanning - principles and applications," in *Methods Mol Biol.*, 04 2007.

[29] R. T. H. Collis, "Lidar," *Appl. Opt.*, vol. 9, no. 8, pp. 1782–1788, Aug 1970. [Online]. Available: http://www.osapublishing.org/ao/abstract.cfm?URI=ao-9-8-1782

[30] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003.

[31] E. M. Mikhail, J. S. Bethel, and J. C. McGlone, "Introduction to modern photogrammetry," *New York*, vol. 19, 2001.

[32] W. Linder, *Digital photogrammetry*. Springer, 2009, vol. 1.

[33] T. Schenk, "Introduction to photogrammetry," *The Ohio State University, Columbus*, vol. 106, 2005.

[34] S. Battiato, S. Curti, M. La Cascia, M. Tortora, and E. Scordato, "Depth map generation by image classification," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5302, pp. 95–104, 04 2004.

[35] F. Cozman and E. Krotkov, "Depth from scattering," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 801–806.

[36] A. Loh and R. Hartley, "Shape from non-homogeneous, non-stationary, anisotropic, perspective texture." in *BMVC*, 01 2005.

[37] A. Francois, G. Medioni, and R. Waupotitsch, "Reconstructing mirror symmetric scenes from a single view using 2-view stereo geometry," in *2002 International Conference on Pattern Recognition*, vol. 4, 2002, pp. 12–16 vol.4.

[38] I. Shimshoni, Y. Moses, and M. Lindenbaum, "Shape reconstruction of 3d bilaterally symmetric surfaces," *International Journal of Computer Vision*, vol. 39, 06 1998.

[39] A. Torralba and A. Oliva, "Depth estimation from image structure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1226–1238, 2002.

[40] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Proceedings of the 18th International Conference on Neural Information Processing Systems*, ser. NIPS'05. Cambridge, MA, USA: MIT Press, 2005, pp. 1161–1168.

[41] A. Tommaselli and M. Reiss, "A photogrammetric method for single image orientation and measurement," *Photogrammetric Engineering and Remote Sensing*, vol. 71, pp. 727–732, 06 2005.

[42] K. Pradeep and A. Rajagopalan, "Improving shape from focus using defocus information," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 1, 2006, pp. 731–734.

[43] G. Haro, "Shape from silhouette consensus," *Pattern Recogn.*, vol. 45, no. 9, pp. 3231–3244, sep 2012. [Online]. Available: https://doi.org/10.1016/j.patcog.2012.02.029

[44] K. Kensek and D. Noble, *Building Information Modeling: BIM in Current and Future Practice*. Wiley, 01 2014.

[45] P. Zanuttigh, L. Minto, G. Marin, F. Dominio, and G. Cortelazzo, *Time-of-flight and structured light depth cameras: Technology and applications*. Springer, 01 2016.

[46] G. Marin, P. Zanuttigh, and S. Mattoccia, "Reliable fusion of tof and stereo depth driven by confidence measures," in *European Conference on Computer Vision*, vol. 9911, 10 2016, pp. 386–401.

[47] L. Li, "Time-of-flight camera - an introduction," in *Springer Brief in Computer Science*, 2014.

[48] Y. Cui, S. Schuon, D. Chan, S. Thrun, and C. Theobalt, "3d shape scanning with a time-of-flight camera," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1173–1180.

[49] E. Kollorz, J. Penne, J. Hornegger, and A. Barke, "Gesture recognition with a time-of-flight camera," *IJISTA*, vol. 5, pp. 334–343, 01 2008.

[50] T. Ringbeck, "A 3d time of flight camera for object detection," *Measurement*, vol. 9, 01 2007.

[51] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 755–762.

[52] G. Sansoni, M. Carocci, and R. Rodella, "Three-dimensional vision based on a combination of gray-code and phase-shift light projection: analysis and compensation of the systematic errors." *Applied optics*, vol. 38 31, pp. 6565–73, 1999.

[53] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. New York,

NY, USA: Association for Computing Machinery, 2011, pp. 559–568. [Online]. Available: https://doi.org/10.1145/2047196.2047270

[54] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. J. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time dynamic 3d surface reconstruction and interaction," in *ACM SIGGRAPH 2011 Talks*, ser. SIGGRAPH '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2037826.2037857

[55] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[56] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 09 2013.

[57] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun, "A large dataset of object scans," *arXiv:1602.02481*, 2016.

[58] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2930–2937.

[59] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3d scene labeling," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3050–3057.

[60] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1625–1632.

[61] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 323–332.

[62] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 303–312.

[63] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2100–2106.

[64] F. Bonarrigo, A. Signoroni, and R. Leonardi, "Multi-view alignment with database of features for an improved usage of high-end 3d scanners," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, p. 148, 2012.

[65] F. Bonarrigo and A. Signoroni, "An enhanced 'optimization-on-a-manifold' framework for global registration of 3d range data," in *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011, pp. 350–357.

[66] F. Bonarrigo and A. Signoroni, "Global registration of large collections of range images with an improved optimization-on-a-manifold approach," *Image and Vision Computing*, vol. 32, no. 6, pp. 437 – 451, 2014.

[67] M. Zollhöfer, M. Nießner, S. Izadi, C. Rhemann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger, "Real-time non-rigid reconstruction using an rgb-d camera," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, 2014.

[68] G. Tam, Z.-Q. Cheng, Y.-K. Lai, F. Langbein, Y. Liu, D. Marshall, R. Martin, X. Sun, and P. Rosin, "Registration of 3d point clouds and meshes: A survey from rigid to nonrigid," *IEEE transactions on visualization and computer graphics*, vol. 19, pp. 1199–217, 07 2013.

[69] S. Salti, F. Tombari, and L. D. Stefano, "A performance evaluation of 3d keypoint detectors," in *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011, pp. 236–243.

[70] S. Filipe. and L. A. Alexandre., "A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset," in *Proceedings of the 9th International Conference on Computer Vision Theory and Applications - Volume 2: VISAPP, (VISIGRAPP 2014)*, INSTICC. SciTePress, 2014, pp. 476–483.

[71] H. Chen and B. Bhanu, "3d free-form object recognition in range images using local surface patches," *Pattern Recognition Letters*, vol. 28, pp. 1252–1262, 01 2004.

[72] J. J. Koenderink and A. J. van Doorn, "Surface shape and curvature scales," *Image Vision Comput.*, vol. 10, no. 8, oct 1992. [Online]. Available: https://doi.org/10.1016/0262-8856(92)90076-F

[73] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 2009, pp. 689–696.

[74] J. Novatnack and K. Nishino, "Scale-dependent 3d geometric features," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[75] ——, "Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images," in *Computer Vision - ECCV 2008*, 01 2008, pp. 440–453.

[76] E. Akagunduz and I. Ulusoy, "3d object representation using transform and scale invariant 3d features," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[77] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud, "Surface feature detection and description with applications to mesh matching," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 373–380.

[78] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, Nov. 2004.

[79] U. Castellani, M. Cristani, S. Fantoni, and V. Murino, "Sparse points matching by combining 3d mesh saliency with statistical descriptors," *Computer Graphics Forum*, vol. 27, 2008.

[80] C. H. Lee, A. Varshney, and D. W. Jacobs, "Mesh saliency," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, 2005, pp. 65–666. [Online]. Available: https://doi.org/10.1145/1186822.1073244

[81] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3d surf for robust three dimensional classification," in *ECCV*, vol. 6316, 09 2010, pp. 589–602.

[82] T. Birdal and S. Ilic, "A point sampling algorithm for 3d matching of irregular geometries," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 6871–6878.

[83] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 914–924, 2012.

[84] L. Zhang, M. J. da Fonseca, A. Ferreira, and C. R. A. e Recuperaçao, "Survey on 3d shape descriptors," *FundaÃgao para a Cincia ea Tecnologia, Lisboa, Portugal, Tech. Rep. Technical Report, DecorAR (FCT POSC/EIA/59938/2004)*, vol. 3, 2007.

[85] I. K. Kazmi, L. You, and J. J. Zhang, "A survey of 2d and 3d shape descriptors," in *2013 10th International Conference Computer Graphics, Imaging and Visualization*, 2013, pp. 1–10.

[86] K. Himri, P. Ridao, and N. Gracias, "3d object recognition based on point clouds in underwater environment with global descriptors: A survey," *Sensors*, vol. 19, no. 20, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/20/4451

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

[88] P. Heider, A. Pierre-Pierre, R. Li, and C. Grimm, "Local shape descriptors, a survey and evaluation," in *Proceedings of the 4th Eurographics Conference on 3D Object Retrieval*, ser. 3DOR '11. Goslar, DEU: Eurographics Association, 2011, pp. 4–56.

[89] J. Yang, Q. Zhang, Y. Xiao, and Z. Cao, "Toldi: An effective and robust approach for 3d local shape description," *Pattern Recognition*, vol. 65, pp. 175–187, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320316303776

[90] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *ECCV*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 356–369.

[91] N. Pezzotti, F. Bonarrigo, and A. Signoroni, "Boosting the computational performance of feature-based multiple 3d scan alignment by iat-k-means clustering," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, 2012, pp. 89–96.

[92] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Computer Vision - ECCV 2004*, vol. 3, 05 2004, pp. 224–237.

[93] F. Tombari, S. Salti, and L. Di Stefano, "Unique shape context for 3d data description," in *Proceedings of the ACM Workshop on 3D Object Retrieval*. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1877808.1877821

[94] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, May 1999.

[95] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Trisi: A distinctive local surface descriptor for 3d modeling and object recognition," in *GRAPP/IVAPP*, 02 2013.

[96] Y. Guo, M. Bennamoun, F. A. Sohel, J. Wan, and M. Lu, "3d free form object recognition using rotational projection statistics," in *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 2013, pp. 1–8.

[97] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 3384–3391.

[98] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3212–3217.

[99] Y. Guo, F. Sohel, M. Bennamoun, J. Wan, and M. Lu, "A novel local surface feature for 3d object recognition under clutter and occlusion," *Information Sciences*, vol. 293, 01 2014.

[100] H. Dinh and S. Kropac, "Multi-resolution spin-images," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, 2006, pp. 863–870.

[101] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3d local surface description and object recognition," *International Journal of Computer Vision*, vol. 105, no. 1, Apr 2013. [Online]. Available: http://dx.doi.org/10.1007/s11263-013-0627-y

[102] T.-W. R. Lo and J. P. Siebert, "Local feature extraction and matching on range images: 2.5d sift," *Comput. Vis. Image Underst.*, vol. 113, no. 12, dec 2009. [Online]. Available: https://doi.org/10.1016/j.cviu.2009.06.005

[103] N. Bayramoglu and A. A. Alatan, "Shape index sift: Range image recognition using local features," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 352–355.

[104] H. Chen and B. Bhanu, "Human ear recognition in 3d," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 718–737, apr 2007. [Online]. Available: https://doi.org/10.1109/TPAMI.2007.1005

[105] A. Flint, A. Dick, and A. v. d. Hengel, "Thrift: Local 3d structure recognition," in *9th Biennial Conference of the Australian Pattern*

Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007), 2007, pp. 182–188.

[106] A. Flint, A. Dick, and A. Hengel, "Local 3d structure recognition in range images," *Computer Vision, IET*, vol. 2, pp. 208 – 217, 01 2009.

[107] R. Rusu, Z. Marton, N. Blodow, and M. Beetz, "Persistent point feature histograms for 3D point clouds," *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, vol. 16, 01 2008.

[108] S. Melzi, R. Spezialetti, F. Tombari, M. M. Bronstein, L. D. Stefano, and E. Rodolà, "Gframes: Gradient-based local reference frame for 3d shape matching," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4624–4633.

[109] S. Melzi, E. Rodolà, U. Castellani, and M. M. Bronstein, "Shape analysis with anisotropic windowed fourier transform," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 470–478.

[110] P. Pingi, M. Corsini, F. Ganovelli, and R. Scopigno, "Fast and simple automatic alignment of large sets of range maps," *Computers & Graphics*, vol. 47, pp. 78–88, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849314001447

[111] A. Petrelli and L. Di Stefano, "Pairwise registration by local orientation cues," *Computer Graphics Forum*, vol. 35, no. 6, pp. 59–72, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12732

[112] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, sep 1975. [Online]. Available: https://doi.org/10.1145/361002.361007

[113] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0146664X82901046

[114] 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.

[115] B. Horn, H. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, pp. 1127–1135, 07 1988.

[116] J.-M. Frahm and M. Pollefeys, "Ransac for (quasi-)degenerate data (qdegsac)," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, 2006, pp. 453–460.

[117] O. Chum, J. Matas, and J. Kittler, "Locally optimized RANSAC," in *Pattern Recognition*, B. Michaelis and G. Krell, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 236–243.

[118] O. Chum and J. Matas, "Optimal randomized RANSAC," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1472–1482, Aug 2008.

[119] D. Aiger, N. J. Mitra, and D. Cohen-Or, "4-points congruent sets for robust pairwise surface registration," *ACM Trans. Graph.*, vol. 27, no. 3, Aug. 2008.

[120] N. Mellado, N. Mitra, and D. Aiger, "Super 4pcs fast global pointcloud registration via smart indexing," *Computer Graphics Forum*, vol. 33, 08 2014.

[121] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314299908329

[122] J. Yang, H. Li, and Y. Jia, "Go-icp: Solving 3d registration efficiently and globally optimally," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1457–1464.

[123] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman, "Point registration via efficient convex relaxation," *ACM Trans. Graph.*, vol. 35, no. 4, Jul. 2016.

[124] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, April 1991, pp. 2724–2729 vol.3.

[125] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[126] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, May 2001, pp. 145–152.

[127] J. Serafin and G. Grisetti, "Nicp: Dense normal based point cloud registration," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 742–749.

[128] X. Huang, G. Mei, and J. Zhang, "Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 366–11 374.

[129] R. Benjemaa and F. Schmitt, "Fast global registration of 3d sampled surfaces using a multi-z-buffer technique," in *Proceedings. International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 1997, pp. 113–120.

[130] A. Makadia, A. Patterson, and K. Daniilidis, "Fully automatic registration of 3d point clouds," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, 2006, pp. 1297–1304.

[131] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[132] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[133] S. Thrun, *Simultaneous Localization and Mapping*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 13–41. [Online]. Available: https://doi.org/10.1007/978-3-540-75388-9_3

[134] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The slam problem: A survey," in *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*. NLD: IOS Press, 2008, pp. 363–371.

[135] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, 12 2017.

[136] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3d slam: A survey on rotation estimation and its use in pose graph optimization," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4597–4604.

[137] J. Fuentes-Pacheco, J. Ascencio, and J. Rendon-Mancha, "Visual simultaneous localization and mapping: A survey," *Artificial Intelligence Review*, vol. 43, 11 2015.

[138] B. T. Phong, "Illumination for computer generated pictures," *Commun. ACM*, vol. 18, no. 6, pp. 311–317, jun 1975. [Online]. Available: https://doi.org/10.1145/360825.360839

[139] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, Nov. 2013.

[140] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-time rgb-d camera relocalization," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013, pp. 173–179.

[141] G. Klein, I. Reid, and B. Williams, "Automatic relocalization and loop closing for real-time monocular slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 09, pp. 1699–1712, sep 2011.

[142] X. Bai, Z. Luo, L. Zhou, H. Chen, L. Li, Z. Hu, H. Fu, and C.-L. Tai, "Pointdsc: Robust point cloud registration using deep spatial consistency," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 15 859–15 869.

[143] V. Couture, N. Martin, and S. Roy, "Unstructured light scanning to overcome interreflections," in *2011 International Conference on Computer Vision*, 2011, pp. 1895–1902.

[144] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proceedings of the Third European Conference on Computer Vision (Vol. II)*, ser. ECCV '94. Berlin, Heidelberg: Springer-Verlag, 1994, pp. 151–158.

[145] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, aug 1987. [Online]. Available: https://doi.org/10.1145/37402.37422

[146] NVIDIA, P. Vingelmann, and F. H. Fitzek, "Cuda, release: 10.2.89," 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[147] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[148] J. Stückler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras," in *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Sep. 2012, pp. 162–167.

[149] M. Meilland, A. I. Comport, and P. Rives, "Dense visual mapping of large scale environments for real-time localisation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 4242–4248.

[150] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3748–3754.

[151] K. L. Lange, R. J. A. Little, and J. M. G. Taylor, "Robust statistical modeling using the t distribution," *Journal of the American Statistical Association*, vol. 84, no. 408, pp. 881–896, 1989.

[152] C. Liu and D. B. Rubin, "ML estimation of the t-distribution using EM and its exstension, ECM and ECME," *Statistica Sinica*, vol. 5, no. 1, pp. 19–39, 1995.

[153] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[154] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[155] M. Meyer and G. Kuschk, "Deep learning based 3d object detection for automotive radar and camera," in *2019 16th European Radar Conference (EuRAD)*, 2019, pp. 133–136.

[156] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3dÂ object detection and 6d pose estimation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 205–220.

[157] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep projective 3d semantic segmentation," in *Computer Analysis of Images and Patterns*, M. Felsberg, A. Heyden, and N. Krüger, Eds. Cham: Springer International Publishing, 2017, pp. 95–107.

[158] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[159] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe, "Know what your neighbors do: 3d semantic segmentation of point clouds," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[160] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai, "Deep learning representation using autoencoder for 3d shape retrieval," *Neurocomputing*, vol. 204, pp. 41–50, 2016, big Learning in Social Media Analytics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216301047

[161] L. Luciano and A. Ben Hamza, "A global geometric framework for 3d shape retrieval using deep learning," *Computers & Graphics*, vol. 79, pp. 14–23, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009784931830195X

[162] D. Stutz and A. Geiger, "Learning 3d shape completion from laser scan data with weak supervision," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1955–1964.

[163] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 07 2017, pp. 6545–6554.

[164] D. Stutz and A. Geiger, "Learning 3d shape completion under weak supervision," *International Journal of Computer Vision*, vol. 128, 05 2020.

[165] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust amp; efficient point cloud registration using pointnet," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7156–7165.

[166] J. Masci, E. Rodolà, D. Boscaini, M. M. Bronstein, and H. Li, "Geometric deep learning," in *SIGGRAPH ASIA 2016 Courses*, ser. SA '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2988458.2988485

[167] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[168] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[169] Baoyuan Liu, Min Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 806–814.

[170] C. Choy, J. Lee, R. Ranftl, J. Park, and V. Koltun, "High-dimensional convolutional networks for geometric pattern recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 224–11 233.

[171] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 337–33 712.

[172] J. Revaud, P. Weinzaepfel, C. R. de Souza, and M. Humenberger, "R2D2: repeatable and reliable detector and descriptor," in *NeurIPS*, 2019.

[173] J. Li and G. H. Lee, "Usip: Unsupervised stable interest point detection from 3d point clouds," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 361–370.

[174] Z. J. Yew and G. H. Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *ECCV*, 2018.

[175] T. Georgiou, Y. Liu, W. Chen, and M. Lew, "A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision," *International Journal of Multimedia Information Retrieval*, vol. 9, no. 3, pp. 135–170, 2020.

[176] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5540–5549.

[177] M. Khoury, Q. Zhou, and V. Koltun, "Learning compact geometric features," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 153–161.

[178] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global context aware local features for robust 3D point matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 195–205.

[179] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 998–1005.

[180] H. Deng, T. Birdal, and S. Ilic, "PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors," in *ECCV*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 620–638.

[181] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Interpretable unsupervised learning on 3d point clouds," *ArXiv*, vol. abs/1712.07262, 2017.

[182] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3d point capsule networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1009–1018.

[183] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo, "Spinnet: Learning a general surface descriptor for 3d point cloud registration," *arXiv preprint arXiv:2011.12149*, 2020.

[184] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "Predator: Registration of 3d point clouds with low overlap," *arXiv preprint arXiv:2011.13005*, 2020.

[185] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv*, 03 2016.

[186] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," *CoRR*, vol. abs/1705.10872, 2017. [Online]. Available: http://arxiv.org/abs/1705.10872

[187] L. van der Maaten and G. Hinton, "Viualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 11 2008.

[188] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, "Learning to find good correspondences," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 2666–2674.

[189] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, H. Liao, and L. Quan, "Learning two-view correspondences and geometry using order-aware network," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 5844–5853.

[190] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *2020*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 4937–4946.

[191] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[192] X. Li, J. K. Pontes, and S. Lucey, "Pointnetlk revisited," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 758–12 767.

[193] H. Wang, X. Liu, W. Kang, Z. Yan, B. wen Wang, and Q. Ning, "Multi-features guidance network for partial-to-partial point cloud registration," *ArXiv*, vol. abs/2011.12079, 2021.

[194] Z. J. Yew and G. H. Lee, "Rpm-net: Robust point matching using learned features," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 821–11 830.

[195] R. Sinkhorn, "A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices," *The Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 876–879, 1964. [Online]. Available: https://doi.org/10.1214/aoms/1177703591

[196] Y. Wang and J. M. Solomon, "Prnet: Self-supervised learning for partial-to-partial registration," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/ebad33b3c9fa1d10327bb55f9e79e2f3-Paper.pdf

[197] H. Deng, T. Birdal, and S. Ilic, "3d local features for direct pairwise registration," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 3239–3248.

[198] Y. Wang and J. Solomon, "Deep closest point: Learning representations for point cloud registration," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3522–3531.

[199] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, oct 2019. [Online]. Available: https://doi.org/10.1145/3326362

[200] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[201] J. C. Gower, "Generalized procrustes analysis," *Psychometrika*, vol. 40, no. 1, pp. 33–51, 1975.

[202] F. Crosilla, A. Beinat, A. Fusiello, E. Maset, and D. Visintini, *Advanced Procrustes Analysis Models in Photogrammetric Computer Vision*, ser. CISM International Centre for Mechanical Sciences. Springer, Cham, 2019, vol. 590.

[203] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[204] F. Arrigoni, A. Fusiello, and B. Rossi, "Camera motion from group synchronization," in *Proceedings of the International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 546–555, (Acceptance rate: 44.4%).

[205] F. Arrigoni and A. Fusiello, "Synchronization problems in computer vision with closed-form solutions," *International Journal of Computer Vision*, vol. 128, no. 1, pp. 26–52, Jan 2020. [Online]. Available: https://doi.org/10.1007/s11263-019-01240-x

[206] X. Huang, Z. Liang, X. Zhou, Y. Xie, L. J. Guibas, and Q. Huang, "Learning transformation synchronization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[207] L. Ding and C. Feng, "Deepmapping: Unsupervised map estimation from multiple point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8642–8651.

[208] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[209] P. Volonghi, G. Baronio, and A. Signoroni, "3d scanning and geometry processing techniques for customised hand orthotics: an experimental assessment," *Virtual and Physical Prototyping*, vol. 13, no. 2, pp. 105–116, 2018.

[210] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5556–5565.

[211] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[212] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[213] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA, USA: MIT Press, 1993.

[214] C. S. Fraser, "Digital camera self-calibration," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 52, no. 4, pp. 149–159, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271697000051

[215] A. Gruen, *Calibration and Orientation of Cameras in Computer Vision*. Springer-Verlag Berlin Heidelberg, 01 2001, vol. 34.

[216] D. Liebowitz and A. Zisserman, "Metric rectification for perspective images of planes," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, 1998, pp. 482–488.

[217] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithm," *Int. J. Comput. Vision*, vol. 47, pp. 131–140, 02 2001.

[218] Y.-C. Du, M. Muslikhin, T.-H. Hsieh, and M.-S. Wang, "Stereo vision-based object recognition and manipulation by regions with convolutional neural network," *Electronics*, vol. 9, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/2/210

[219] D. Bappy and H. Rahman, "A study in 3d structure detection implementing forward camera motion," Ph.D. dissertation, Mälardalen University Sweden, 05 2012.

[220] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[221] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.