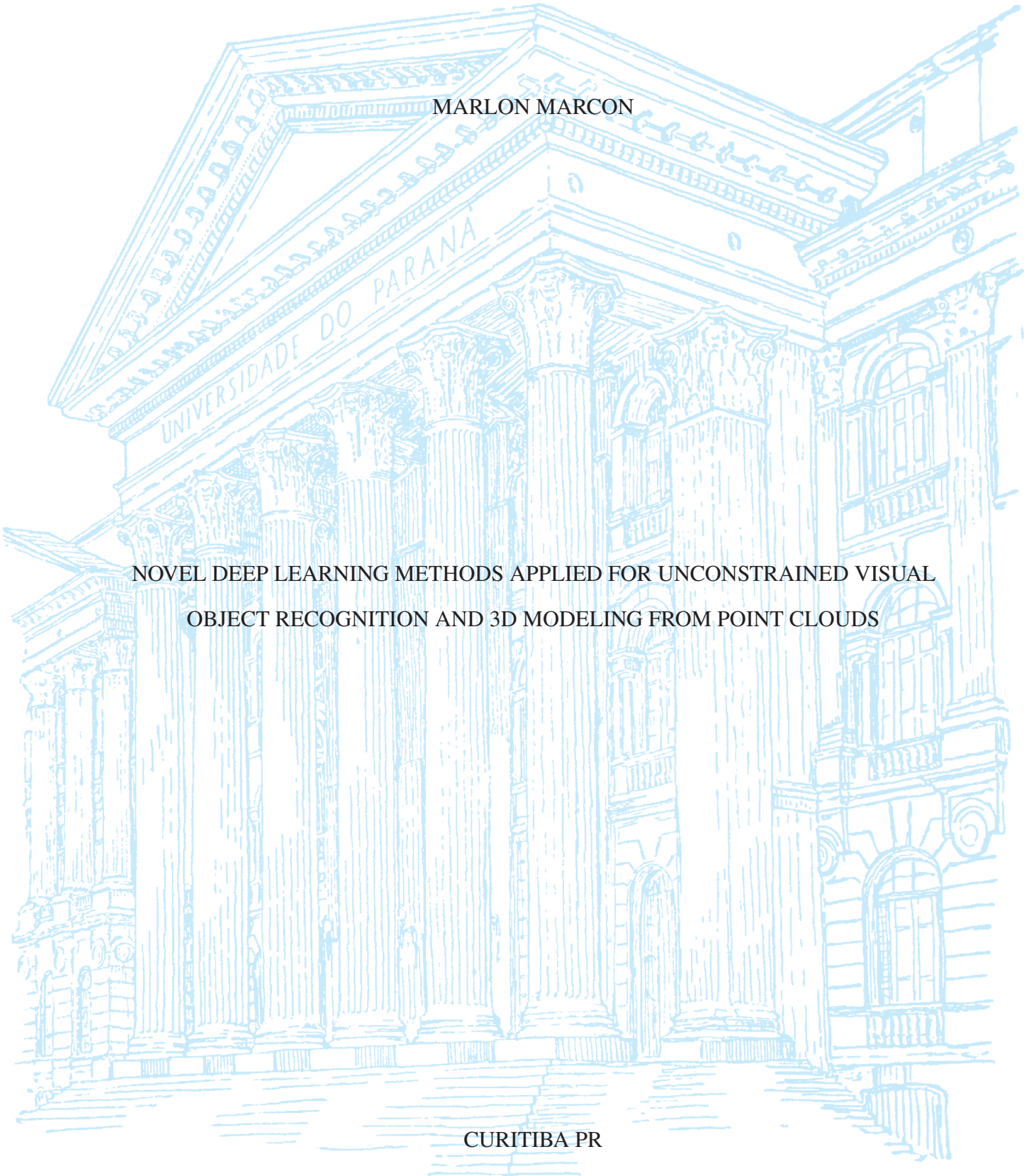
UNIVERSIDADE FEDERAL DO PARANÁ

MARLON MARCON



NOVEL DEEP LEARNING METHODS APPLIED FOR UNCONSTRAINED VISUAL OBJECT RECOGNITION AND 3D MODELING FROM POINT CLOUDS

CURITIBA PR

2020

MARLON MARCON

NOVEL DEEP LEARNING METHODS APPLIED FOR UNCONSTRAINED VISUAL OBJECT RECOGNITION AND 3D MODELING FROM POINT CLOUDS

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Luciano Silva.

CURITIBA PR

2020

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

M321n Marcon, Marlon
Novel deep learning methods applied for unconstrained visual object recognition and 3D modeling from point clouds [recurso eletrônico] / Marlon Marcon. – Curitiba, 2020.

Tese - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2020.

Orientador: Luciano Silva.

1. Visão por computador. 2. Redes neurais (Computação). I. Universidade Federal do Paraná. II. Silva, Luciano. III. Título.

CDD: 006.37

Bibliotecária: Vanusa Maciel CRB- 9/1928

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **MARLON MARCON** intitulada: **Novel deep learning methods applied for unconstrained visual object recognition and 3D modeling from point clouds**, sob orientação do Prof. Dr. LUCIANO SILVA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 09 de Novembro de 2020.

Assinatura Eletrônica
18/11/2020 07:39:36.0
LUCIANO SILVA
Presidente da Banca Examinadora

Assinatura Eletrônica
18/11/2020 13:39:09.0
JURANDY GOMES DE ALMEIDA JUNIOR
Avaliador Externo (UNIVERSIDADE FEDERAL DE SÃO PAULO)

Assinatura Eletrônica
18/11/2020 09:35:28.0
OLGA REGINA PEREIRA BELLON
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
18/11/2020 09:59:18.0
RODRIGO MINETTO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Aos meus filhos Luca e Heitor (in memoriam) e minha esposa Natalia.

ACKNOWLEDGEMENTS

Nesta seção de agradecimentos me permitirei escrever em Português porque as coisas que vêm do coração precisam ser ditas na nossa língua mãe.

Após um processo difícil da perda de um filho, nascimento de outro, viagens todas as semanas para Curitiba, mudança para Curitiba, mudança para Itália, mudança de volta para o Brasil, tudo isso com mulher e filho, enfim... estou aqui e doutor. Neste processo, muitas pessoas contribuíram muito para este momento que vivo agora, espero lembrar de todas...

Agradeço ao meu filho Luca que durante toda a sua vida não conhece o pai sem estar fazendo doutorado e ter férias. Luca, esse sacrifício todo foi para você, e agora o papai vai te dar toda atenção que você merece. Obrigado filho!

Agradeço também a minha esposa Natalia, que abdicou da sua vida profissional para me ajudar, segurando as pontas e muitas vezes foi pai e mãe do Luca. Sem você as coisas teriam sido muito mais díficeis. Obrigado Nati!

Outra pessoa fundamental durante o processo que merece destaque é a minha mãe Vera, essa pessoa maravilhosa que aceitou o filho depois de 17 anos morando fora, com todas as manias, além da nora e do neto para morarem na casa dela por um ano, e na Itália. Obrigado mãe pelo auxílio e pela criação que você me deu, você também foi fundamental neste processo! Agradeço imensamente também ao Andrea e ao *nonno Beppe*, pelo importante auxílio na adaptação à Itália.

Agradeço também, é claro, aos demais membros da minha família, que propiciaram momentos de fuga dos "problemas" do doutorado. Obrigado principalmente ao meu pai Orides, meu irmão Camillo, minha irmã Beatriz, à Sandra e aos meus sogros Noemi e Urbano.

Agradeço muito *ai ragazzi del CVLab-Unibo* pela acolhida e amizade, *grazie mille a tutti*: Alessio, Fabio, Filippo, Gianluca, Luca, Matteo, Pierluigi e Riccardo. Além disso, não posso deixar de nominar os professores Luigi Di Stefano e Samuele Salti, com os quais aprendi lições que vou usar muito na minha vida pessoal e profissional.

Agradeço também aos colegas de IMAGO e ao professor Luciano Silva pela orientação e auxílio em momentos críticos da tese.

Agradeço aos colegas e amigos de doutorado do convênio UTFPR-UFPR: André, Eliane, Evandro, Franciele, Newton, Robison, Rúbia, Soelaine e Wellton, com os quais compartilhei angústias e sucessos. Também aos professores Roberto e Eduardo que se deslocaram de suas casas em Curitiba, para ministrar aulas para nós em Pato Branco.

Agradeço a Universidade Tecnológica Federal do Paraná, pelo afastamento concedido para finalização do curso.

Agradeço aos professores Jurandy Gomes de Almeida Júnior, Olga Regina Pereira Bellon e Rodrigo Minetto por terem se disponibilizado a avaliar, e colaborado ativamente na melhoria desta tese de doutorado.

Para finalizar, agradeço a todos que de alguma forma contribuíram para o desenvolvimento deste trabalho e também a todos que torceram por mim nesta empreitada.

RESUMO

Soluções baseadas em aprendizagem profunda evoluíram rapidamente e superaram abordagens clássicas no âmbito da Visão Computacional (VC). Metodologias baseadas em processamento de textura (também chamadas de 2D), são tecnologias maduras com eficiência comprovada em diversos cenários de aplicação. Trabalhar com aprendizagem profunda em aplicações de VC e computação gráfica no contexto 3D não é trivial. Alguns fatores podem ser considerados: encontrar uma representação confiável para os dados; rotular amostras positivas e negativas para situações de aprendizagem supervisionada; e, obter invariância à rotação, induzida durante o treinamento. Processamento em tempo real para aplicações de reconhecimento de objetos e estimativa da pose também são desafiadoras, e métodos tradicionais focam principalmente na acurácia e não provêm tal propriedade. Nesta tese de doutorado, são apresentadas estratégias para lidar com tais situações, e para tal, duas partes principais são apresentadas: a primeira focada no desenvolvimento de técnicas efetivas para aplicações de VC baseadas em características de forma geral, e a segunda, que propõe estratégias para melhorar métodos de reconhecimento de objetos. O descritor LEAD é apresentado, sendo este o primeiro descritor de características local, equivariante à rotação, baseado em aprendizagem não supervisionada a partir de nuvens de pontos. Além disso, esta tese apresenta Compass, o primeiro método para definir e extrair a orientação canônica de formas tridimensionais, utilizando somente aprendizagem profunda. Com a união das duas propostas anteriores, também é apresentado o primeiro descritor local 3D invariante à rotação, baseado em aprendizagem não supervisionada, denominado SOUND. A eficácia das propostas foi avaliada experimentalmente em conjuntos de dados de referência para registro de superfícies 3D, e os resultados demonstram que as propostas deste documento superam outras técnicas não supervisionadas, além de manter-se competitivas com relação às supervisionadas. Relacionado às melhorias nos métodos de reconhecimento de objetos e estimativa da pose, neste trabalho foi proposta uma abordagem que utiliza detecção de objetos salientes, a qual provê melhorias consideráveis em relação a técnicas tradicionais. Resultados confirmam que o método impulsionado pelo uso da saliência, pode acelerar substancialmente o reconhecimento, impactando muito pouco ou até melhorando na acurácia. Também foi conduzido um extensivo estudo relacionado ao uso de arquiteturas baseadas em aprendizagem profunda, como extratores de características independentes, bem como seu desempenho no reconhecimento de objetos tridimensionais. Por fim, um método para detecção de objetos em estimativa da pose com seis graus de liberdade é apresentado. Tal proposta, identifica objetos em imagens RGB-D, extraindo características visuais e estimando a pose de objetos de forma precisa, por meio de descritores locais, e possibilita o processamento em tempo real em aplicações de estimativa da pose. Acredita-se que os avanços apresentados nesta tese, auxiliarão pesquisadores no desenvolvimento de aplicações de VC 3D, em áreas como robótica, direção autônoma e tecnologias assistivas.

Palavras-chave: Visão Computacional 3D. Aprendizagem não-supervisionada. Transferência de conhecimento. Modelos computacionais. Redes neurais convolucionais esféricas. Descritores locais. Reconhecimento de objetos em tempo real.

ABSTRACT

Deep-learning-based solutions are rapidly evolving and outperforming classical hand-crafted approaches in the Computer Vision (CV) field. Texture-based methodologies (a.k.a 2D) are mature technologies with proven efficiency in several application scenarios. To work with deep learning for 3D CV and graphics applications is not straightforward. Some factors could be considered: finding a reliable representation from data; annotating data with true and false examples in a supervised fashion training; and achieving invariance to rotation induced during training. Real-time processing for 3D object recognition (3DOR) and pose estimation applications is also untrivial, and standard pipelines focus on the accuracy and do not provide such property. In this doctoral thesis, we present some strategies to tackle these issues. We split this dissertation into two main topics: first focusing on developing reliable techniques for generic feature-based CV applications and the second which proposes strategies to improve object recognition methods. We introduce LEAD, the first unsupervised rotation-equivariant 3D local feature descriptor learned from raw point cloud data. We also realize the first end-to-end learning approach to define and extract the canonical orientation of 3D shapes, which we named Compass. With the achievements of both previous methods, we merge them and propose, the first unsupervised rotation-invariant 3D descriptor, called SOUND. We evaluate our proposal's impact experimentally, which outperform existing unsupervised methods and achieve competitive results against the supervised approaches through extensive experiments on standard surface registration datasets. To update the traditional pipeline for object recognition and pose estimation, we propose a boosted pipeline that uses saliency detection algorithms, and we found considerable improvement in such methodology. Results confirm that the boosted pipeline can substantially speed up processing time with limited impacts or even recognition accuracy benefits. We conducted a comprehensive study regarding 2D deep networks as off-the-shelf feature extractors and evaluated their 3DOR's performance. Finally, we propose a novel pipeline to detect objects and estimate their 6DoF pose. To do so, we identify objects in RGB-D images applying visual features and estimate a fine-adjusted object's pose with 3D local descriptors. Our proposal unlocks real-time processing for pose estimation applications. We trust our achievements will help researchers develop 3D CV applications in the robotics, autonomous driving, and assistive technology fields.

Keywords: 3D Computer Vision. Unsupervised learning. Transfer learning. Computational models. Spherical CNNs. Local descriptors. Real-time object recognition.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3DSN	3D Smooth Net
6DoF	Six Degrees of Freedom
ACF	Alternating Contiguous Frames
ANN	Artificial Neural Networks
AUC	Area Under the Curve
CAD	Computer-Aided design
CGF	Compact Geometric Features
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CSHOT	Colored Signatures of Histogram OrienTation
CV	Computer Vision
DL	Deep Learning
DSS	Deeply Supervised Salient
ETH	Eidgenössische Technische Hochschule
FAST	Features from Accelerated Segment Test
FCGF	Fully-Convolutional Geometric Features
FGR	Fast Global Registration
FLANN	Fast Library for Approximate Nearest Neighbor
FLARE	Fast LocAl Reference framE algorithm
FPFH	Fast Point Features Histogram
FPS	Frames Per Second
GCG	Geometric Consistency Grouping
GNB	Gaussian Naïve Bayes
GPU	Graphical Processing Unit
ICP	Iterative Closest Point
IoU	Intersection over Union
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ISS	Intrinsic Shape Signatures
KNN	k-Nearest Neighbor
LEAD	Learned EquivariAnt Descriptor
LEAD-PN	LEArned Descriptor PointNet
LR	Logistic Regression
LRF	Local Reference Frame
LSO	Leave-sequence-out
LTS	Long-Term Support

ML	Machine Learning
MLP	Multi-Layer Perceptron
PCA	Principal Component Analysis
PCL	Point Cloud Library
PFH	Point Feature Histogram
PPF	Point Pair Feature
PRC	Precision-recall curve
PRIN	Point-wise Rotation-Invariant Network
RA	Reference Axis
RANSAC	RANdom SAmple Consensus
ReLU	Rectified Linear Unit
RF	Random Forest
RGB-D	Red, Green, Blue, and Depth
RMSE	Root Mean Squared Error
RoPS	Rotational Projection Statistics
RRE	Relative Rotation Error
SfM	Structure from Motion
RTE	Relative Translation Error
SGD	Stochastic Gradient Descent
SHOT	Signatures of Histogram OrienTation
SI	Spin Images
SIFT	Scale-invariant feature transform
SLAM	Simultaneous Location And Mapping
$\mathrm{SO}(3)$	Special Orthogonal Group
SOUND	Self-Orienting UNsupervised Descriptor
SSD	Single Shot MultiBox Detector
SVC	Support Vector Classifier
SVM	Support Vector Machines
T-Net	Transformation Network
TOLDI	Triple Orthogonal Local Depth Images
US	Uniform Sampling
USC	Unique Shape Context
VTK	The Visualization Toolkit
YOLO	You Only Look Once

CONTENTS

1 INTRODUCTION

Deep learning is a well-established field regarding 2D Computer Vision (CV) problems, attracting considerable attention in the last few years. Deep methods have outperformed high engineered methods by merely learning from data. Accurate deep-learning-based methods rely on a high amount of data. When we deal with supervised learning approaches, despite their success, another hindrance is found: such data must be labeled with positive and negative examples. On the other hand, unsupervised techniques act mimicking humans and mammals, i.e., learning by observation, not necessarily with supervision.

3D data allow understanding better the surrounding environment and act as complementary information regarding 2D images (Guo et al., 2020). However, when we stand on 3D applications, including autonomous driving, robotics, remote sensing, and medical treatment (Chen et al., 2017), we realize that processing 3D data is not straightforward (Guo et al., 2020). The first barrier we find is a standard representation, including depth images, point clouds, meshes, and volumetric grids. Luckily, recent advancements encourage adopting point clouds, mainly by achievements with the PointNets (Qi et al., 2017a,b). Additionally, the point cloud representation preserves the original geometric information in 3D space without any discretization and is intimately related to RGB-D images.

Several application situations are related to point cloud processing, including surface registration, shape classification and retrieval, object recognition, and 6DoF[1] pose estimation (Su et al., 2015; Elbaz et al., 2017; Zeng et al., 2017b; Manhardt et al., 2019; Gojcic et al., 2019; Choy et al., 2020). Most of these applications rely on the local features' use to identify similarities between shapes. Effective pipelines leveraging the feature-matching paradigm hinge upon compact representations of the local geometry referred to as *descriptors*. Descriptors should be *invariant* and *robust* to the nuisances encountered in 3D CV scenarios, such as viewpoint changes, sensor noise, point density variations, occlusions and clutter.

Conceiving hand-crafted functions to extract robust and distinctive features from 3D data has a relatively long history in CV (Johnson and Hebert, 1999; Rusu et al., 2009; Tombari et al., 2010; Guo et al., 2013a; Salti et al., 2014). Nevertheless, due to the challenging settings mentioned earlier, designing an effective local descriptor turns out a rather complex effort. Taking advantage of the emerging deep networks in processing 2D visual data, and working on such challenges, the attention was shifted toward learning *deep* local descriptors from 3D data (Zeng et al., 2017a; Deng et al., 2018b,a; Khoury et al., 2017; Gojcic et al., 2019; Spezialetti et al., 2019; Choy et al., 2019b; Bai et al., 2020). Deep strategies have outperformed conventional hand-crafted techniques by far, achieving the state-of-the-art in most benchmarking scenarios.

Despite the importance of achieving invariance to viewpoint changes to 3D descriptors, learned approaches exhibit a performance drop when the training and testing sets are on different viewpoints or imposed random rotations (Zeng et al., 2017a; Deng et al., 2018b; Esteves et al., 2018). This reduction is probably because 3D data under rotations induce distinct network features, as demonstrated in 3D object classification problems (Sedaghat et al., 2016). A popular strategy to provide rotation invariance is to express the 3D coordinates of the cloud's points w.r.t. a coordinate system, defining a Local Reference Frame (LRF) (Khoury et al., 2017; Gojcic et al., 2019) or a Reference Axis (RA) (Deng et al., 2018a). Several hand-crafted proposals aim to define a reliable canonical orientation for 3D surfaces (Petrelli and Di Stefano, 2012; Salti et al.,

[1]Six degrees of freedom refers to the geometrical transformation representing a rigid body's movement in a 3D space. We use a 4×4 matrix to represent the composition between the estimated rotation $\tilde{R}$ and translation $\tilde{T}$.

2014; Yang et al., 2017; Melzi et al., 2019; Zhu et al., 2020), however, hand-crafted choices may inject imprecisions on the process. A data-driven approach could surpass such issues, but none of the previous works has yielded such estimative without some pre-assumption from the data. This thesis introduces the first end-to-end, entirely data-driven, feasible way of learning a robust canonical orientation for point clouds. Based on the observation that a canonical pose's inherent property is equivariance to 3D rotations, we propose to employ Spherical CNNs (Cohen et al., 2018; Esteves et al., 2018), which are equivariant by design.

When we deal with object recognition applications, the standardized protocol for local descriptors (Aldoma et al., 2012b) could be an obstacle to real-time processing. The term real-time may cause uncertainty and depends on the application. In this thesis's context, we consider real-time applications those with a frame-rate of at least 30 frames per second (FPS), but a lower rate could also be compliant to a wide range of systems. Some strategies, such as keypoint extraction, segmentation, and highlighting specific areas (Tombari et al., 2013; Gomes et al., 2013), are widely used to speed up the whole process, but eventually, they face accuracy underperformance. As a result of this doctoral thesis, we propose a method that boosts object recognition accuracy and time processing on the object recognition task, published in Marcon et al. (2019). One of our thesis' claim is that we can take advantage of 2D CV deep learning proposals to improve 3D object recognition.

Despite improvements in the 3D object recognition and pose estimation pipeline, real-time applications are still defying (Hodan et al., 2018). Otherwise, 2D-based proposals deal effortlessly with efficient real-time object detection (Redmon et al., 2016; Liu et al., 2016), and color feature extractors may assist on such application scenarios (He et al., 2016a; Sandler et al., 2018; Xie et al., 2017; Tan and Le, 2019). It is fundamental to apply such prior efforts allied to 3D-focused routines in application conditions. Agrawal et al. (2014) and Huh et al. (2016) have verified that models trained on the ImageNet dataset present a high transferring capacity and offer efficient solutions for different contexts.

Based on the previously disclosed, this thesis aims to provide novel deep-learning-based local descriptors, evaluate and validate them on feature-based registration benchmarks. We adopt an unsupervised procedure to capture nuisances from no labeled data. We yield rotation invariance by leveraging Spherical CNNs (Cohen et al., 2018), and develop the first end-to-end learned LRF, named Compass. We also extend an equivariant local descriptor (Spezialetti et al., 2019) and combine it with Compass, providing the first self-orienting (a.k.a. invariant) local descriptor. In the 3D object recognition task, we propose combining 2D deep techniques with traditional 3D feature-based methods. We speed-up and improve the standard local descriptors pipeline's accuracy, adding a process named saliency boost. We also combine off-the-shelf deep-based color and shape features, provide competitive object descriptors, and propose an efficient real-time pose estimation pipeline.

1.1 MOTIVATION

The first part of this thesis lies basically on exploring unsupervised approaches to learn 3D surface embeddings. Despite the availability of a high amount of data, to employ a deep-learning-based supervised approach demands a previous human-made annotation on it, becoming an expensive and laborious task. Unsupervised or even self-supervised methods are next to humans and mammals' way of learning, mimicking our instinct of learning by environmental observation and comprehension without an explicit tutor. According to LeCun et al. (2015), unsupervised approaches have the potential of attracting more attention in a few years, concerning the supervised ones.

About 3D local descriptors, Esteves et al. (2018) observed a considerable drop in performance when inducing random rotations in a deep model trained in a canonical orientation form. So, it is crucial to develop rotation invariant methods or to find a way to estimate the surface's orientation and then impose a canonical form at test time. Existent methods tend to extract the correct orientation from training data, but results of Gojcic et al. (2019) and (Li et al., 2020a) show a significant performance decrease from models trained on an indoor dataset and tested in an outdoor environment.

The availability of low-cost RGB-D sensors, which deliver in real-time color and depth information, has propitiated the emergence of datasets that simulate *real world* environments. Such sets enable a fairly benchmarking of state-of-the-art methods, providing real situations, e. g., clutter, occlusions, and a significant degree of noise inherent in their images. Robust 3D CV techniques must deal with such problems efficiently. To do so, it is required to employ state-of-the-art CV algorithms, provided by libraries such as OpenCV (Bradski and Kaehler, 2008), PCL (Rusu and Cousins, 2011), and Open3D (Zhou et al., 2018).

Compared to 3D applications, deep learning for 2D CV is a more stated field, with several successful strategies that outperform classical ones. Problems like object detection (Redmon et al., 2016; Liu et al., 2016), segmentation (Hou et al., 2017; Liu et al., 2019a), and feature extraction (Krizhevsky et al., 2012; He et al., 2016a; Xie et al., 2017; Tan and Le, 2019) are well established tasks for 2D. The efforts on the 3D methods' development focus on describing shape representation of surfaces and only a few studies, still hand-crafted, focus on exploring textured point clouds (Rusu et al., 2008; Salti et al., 2014).

Object recognition and 6DoF pose estimation in real-time is an *open* CV problem. The pipeline involved in this process demands a high computational power to execute its steps. Improving this pipeline is fundamental to speed-up the whole method and apply it in a real-time situation. The second part of this doctoral thesis explores the combination of the best of 2D and 3D and proposes a method to move a step forward to improve 3D-based applications.

A wide range of application scenarios can be addressed with 3D modeling and Object recognition and pose estimation. In robotics, applications include manipulation of household objects (Murali et al., 2020), bin-picking (Yan et al., 2020), and intelligent assembly in industrial lines (Li et al., 2020b). Another field attracting huge attention in the last few years, autonomous driving, consumes methods regarding 3D modeling and pose estimation (Chen et al., 2017; Arnold et al., 2019). For welfare and healthcare applications, assistive technology systems rely on object localization and recognition, scene understanding, and pose estimation (Leo et al., 2018).

This thesis relies on feature-based methods applied to registration, object recognition, and pose estimation. Figure 1.1 depicts a basic block diagram of such applications. Given two point clouds as input, such systems aim to estimate a transformation between them. In a registration scenario, the resulting transformation matrix will put both clouds in the same coordinate system, thus aligning them. In a pose estimation scenario, the position and orientation of an object on the scene, i.e., the 6DoF pose, is sought. Pose estimation and registration pipelines involve three typical stages: pre-processing, feature-based estimation, and post-processing. The pre-processing is responsible for prepare the ground for the description step, i.e., detecting keypoints or objects, segmenting the cloud, and applying filtering methods. The second stage is the core of such applications. It executes a sequence of steps: the description, feature-matching, filtering the correspondences, and finally estimating a coarse pose between the input clouds. In the end, the post-processing stage performs, if desired, a pose refinement and hypothesis verification. Throughout this dissertation, we explore in a certain way most of the presented steps, excepting Noise filtering and Hypothesis verification. For more details on the pipelines, please refer to Chapter 2.

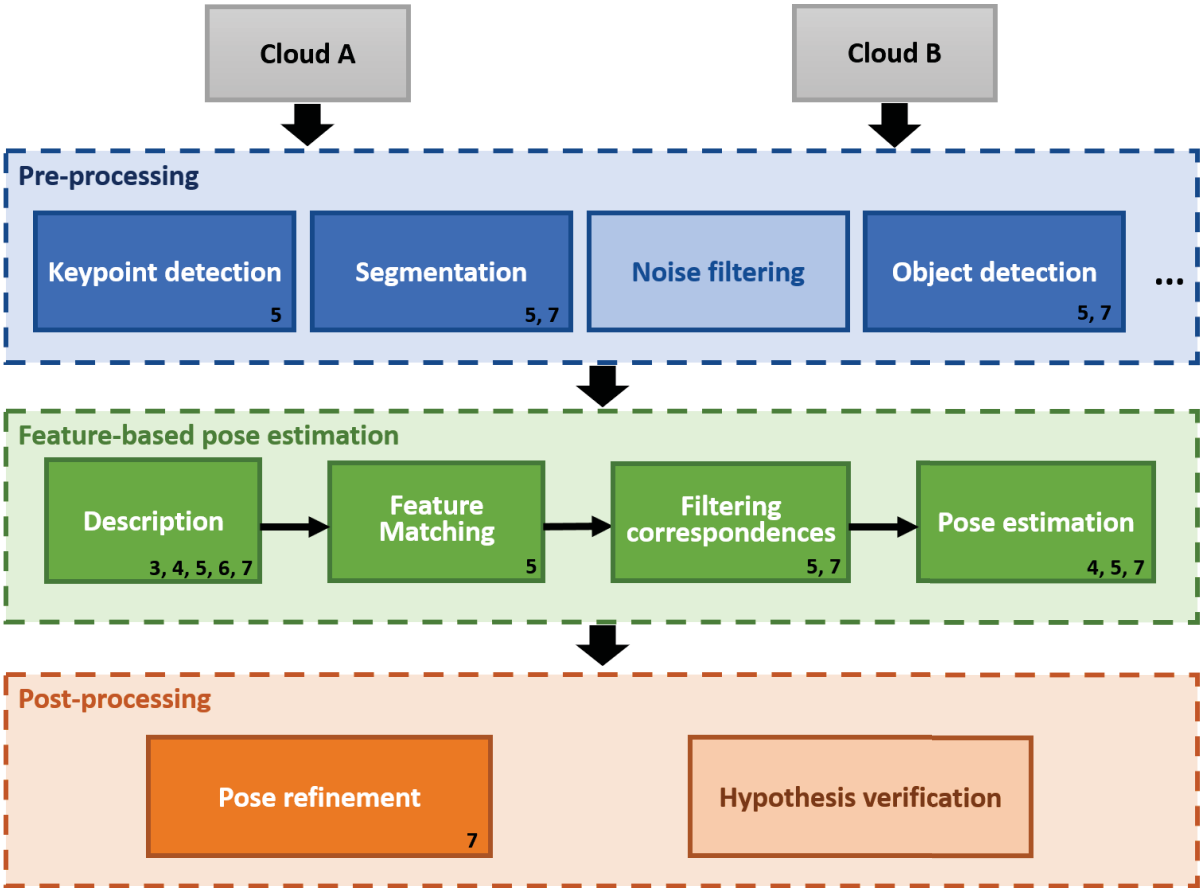


Figure 1.1: Block diagram of feature-based applications explored in this dissertation. Numbers inside each box refer to the chapters that address each step.

1.2 OBJECTIVES

This doctoral thesis foundation lies in developing unsupervised strategies to describe 3D patches, dealing with nuisances of 3D data based on the previously presented. We also propose combining 2D deep learning techniques with 3D methods, evaluating them in feature-based application tasks. To this end, this doctoral thesis's main objective is **to provide strategies to improve feature-based applications on 3D point clouds**.

Among all the possible feature-based problems addressable through 3D CV techniques, we will focus our efforts on 3D registration, 3D object detection, and 6DoF pose estimation scenarios. Due to the main objective's coverage, we refine and explicitly point which strategies will be tackled, by presenting the following specific objectives:

- To propose, evaluate, and validate an end-to-end rotation invariant local descriptor from unlabeled data for feature-based applications.
- To evaluate the proposed techniques in a feature-based registration scenario, and on a rotation-invariant full shape object recognition problem;
- To evaluate 2D visual features applied to the object recognition scenario on RGB-D images;
- To propose and evaluate improvements on the standard object recognition pipeline, based on 2D visual features from pre-trained deep networks;

- To propose and evaluate a generic pipeline of object recognition and 6DoF pose estimation in uncontrolled indoor environments.

1.3 CONTRIBUTIONS

To achieve our objectives, several contributions have been made during the doctoral study:

- We have significantly improved an existing efficient equivariant local descriptor, proposed by Spezialetti et al. (2019), in terms of accuracy as well as in description time, and introduced the LEAD descriptor;
- At the best of our knowledge, we have proposed the first full-data-driven LRF, named Compass. As assessed by the results, we outperform the state-of-the-art competitors, in standard registration datasets;
- The canonical orientation provided by Compass, when associated with a PointNet architecture, outperformed state-of-the-art methods in a full-shape object recognition scenario, with imposed random rotations to test data;
- Our equivariant local descriptor LEAD is the runner-up on the standard 3D registration benchmark. Additionally, LEAD presents by far the best performance in transfer learning in an extremely challenging outdoor dataset. Considering only unsupervised approaches, we beat all the competitors, and most of them by a large margin.
- This thesis presents, at the best of our knowledge, the first self-orienting local descriptor, named SOUND. By leveraging the equivariance property of Spherical CNNs, we can extract discriminant embeddings and orientation from 3D patches. Results put SOUND at the same level as the most efficient descriptors, in an indoor fashion, or even transferring to outdoor data. As an $\mathrm{SO}(3)$ manifold-living solution, the LRF or the descriptor can be unplugged and replaced by any other technique on the same conceptual basis.
- We propose an initial step on the standard object recognition pipeline based on local descriptors, named Saliency Boost. This process employs a salient object detection step that speeds up the whole process by almost five times and improves tested methods' accuracy on different datasets.
- We perform an extensive evaluation of traditional state-of-the-art networks on the RGB-D object recognition scenario. We compared pre-trained models learned from the ImageNet dataset as off-the-shelf feature extractors and performed a comprehensive evaluation regarding category and instance recognition in a standard dataset.
- We present and evaluate an efficient pipeline for object detection and 6DoF pose estimation that enables real-time processing for point clouds. Our pipeline is composed of modules that combine visual features extracted by pre-trained networks, feature-based registration methods, and fine-tuning dense registration methods. Performance results show a potential to use on real-time systems, in a scheduled operation.

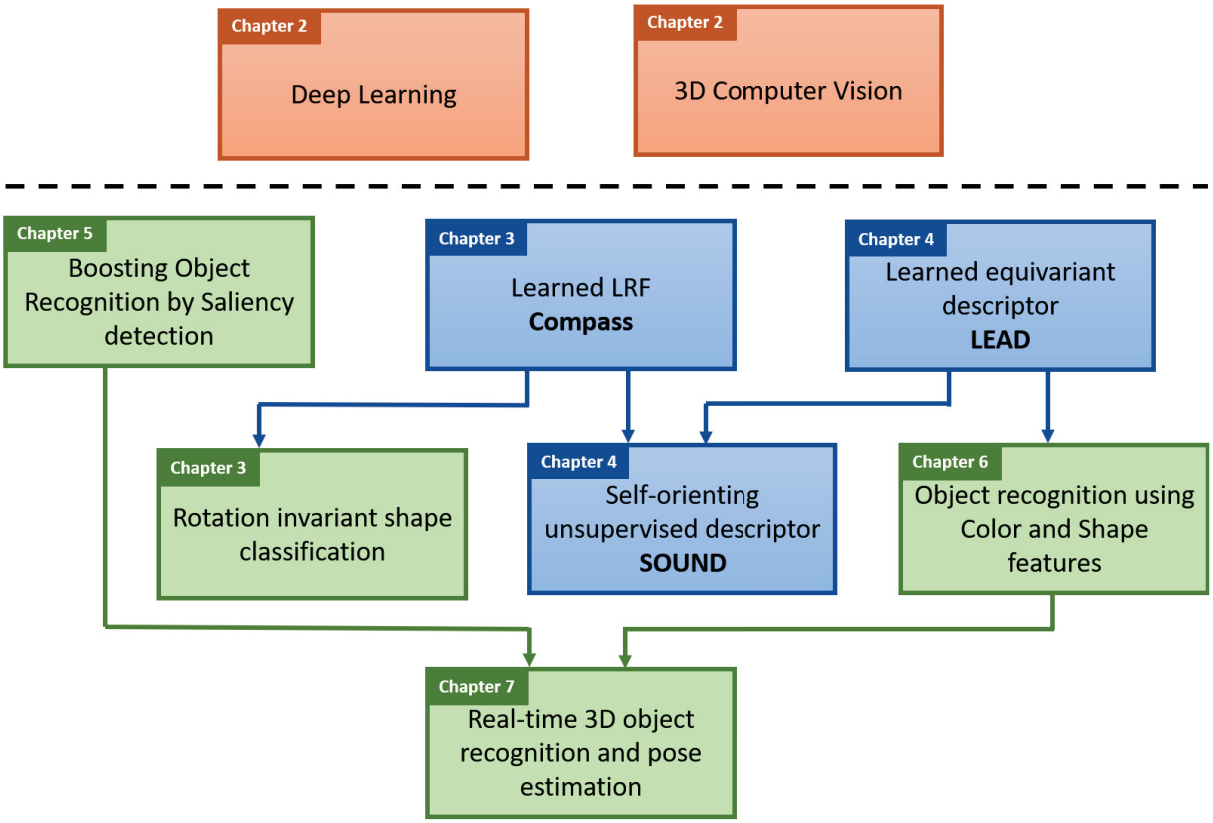


Figure 1.2: Graphical outline of this thesis. Red shaded parts refer to background concepts, Blue are related to registration and Green to object recognition developments.

1.4 OUTLINE

Part of this doctoral thesis' development includes an internship period, fulfilled under the supervision of Prof. Luigi Di Stefano from CVLab, University of Bologna, Italy. Researches regarding the local descriptors and the learned LRF were conceived during the period covering from 02-2019 to 02-2020, explicitly related to Chapters 3 and 4.

We perform an article-oriented document from Chapter 3 to 7, presenting self-content chapters with a brief related works section, methodology, and results. Figure 1.2 depicts a graphical outline and lists our main contributions per chapter and their relationship throughout this thesis. Following we present this document structure:

- **Chapter 2** presents some fundamentals regarding deep learning and 3D CV applications, presenting from the basics to state-of-the-art solutions employed in this work;
- In **Chapter 3**, we introduce our proposed LRF network, named Compass, presenting an extensive evaluation of LRF repeatability and rotation-invariant shape classification scenarios. This chapter is related to a paper accepted as poster on the *Conference on Neural Information Processing Systems* (NeurIPS), and the results were extracted from it;
- After, in **Chapter 4**, we explore the improvements made on the equivariant descriptors proposed by Spezialetti et al. (2019) and published on the main track of the *International Conference on Computer Vision* (ICCV), and present a novel proposal named as SOUND, being the first self-orienting local descriptor. Part of this chapter, regarding the LEAD

descriptor, was submitted to the *IEEE Transactions on Pattern Analysis and Machine Intelligence* (TPAMI), and the results and figures were extracted from it;

- **Chapter 5** is related to the Saliency Boost step we proposed to speed-up and improve accuracy on the standard local descriptors pipeline for object recognition. This chapter was partially published in the *International Conference on Image Analysis and Processing* (ICIAP), and the results regarding the Kinect dataset are extracted from Marcon et al. (2019);

- **Chapter 6** presents an evaluation of the visual feature extractors based on deep learning architectures. We also present and evaluate a proposal of combining color and shape features in the object recognition scenario;

- **Chapter 7** introduces our proposed pipeline for object detection and 6DoF pose estimation on point clouds. This chapter was submitted to the *International Conference on Computer Vision Theory and Applications* (VISAPP), together with some results presented in Chapter 6;

- **Chapter 8** concludes our thesis, presents our findings, future directions, and final remarks.

2 FUNDAMENTALS

This thesis is founded in two main interconnected areas of Artificial Intelligence: CV and ML. This chapter presents some basics about both, starting with a basic overview of ML and aspects regarding the construction of 2D and 3D deep learning methods (Section 2.1), followed by more applied concepts regarding CV algorithms, specifically for a 3D context (Section 2.2).

2.1 DEEP LEARNING

Deep learning models unlock to computers the ability to learn representations from data with different levels of abstraction (Murphy, 2012). Unlike former hand-crafted techniques that use a "human" comprehension of a problem to represent a set of features, deep learning models learn these features using statistical models, aiming at finding the most likable answer to the data used as input.

Deep learning methods have consistently improved the state-of-the-art in many applications, such as speech recognition and CV (Goodfellow et al., 2016). Tasks that are natural for human beings, such as recognizing a specific object and saying who its owner is, or understanding the main subject of a speech or a lecture, are still untrivial for computers. With deep learning models, we can go much faster toward solving challenging problems. These methods' remarkable results have a price: to have good models, we must have useful high-level labeled data or high-engineered methods that could learn from raw data.

To understand deep learning, one must know some basic concepts about it as a subfield of Machine Learning (ML). To make this thesis self-contained, this chapter starts with a brief introduction to ML concepts. It is essential to understand all of the deep learning methods employed in this study. After that, we explain deep networks more thoroughly, focusing on CV applications. Finally, we present some aspects regarding the usage of deep learning methods applied to 3D CV applications.

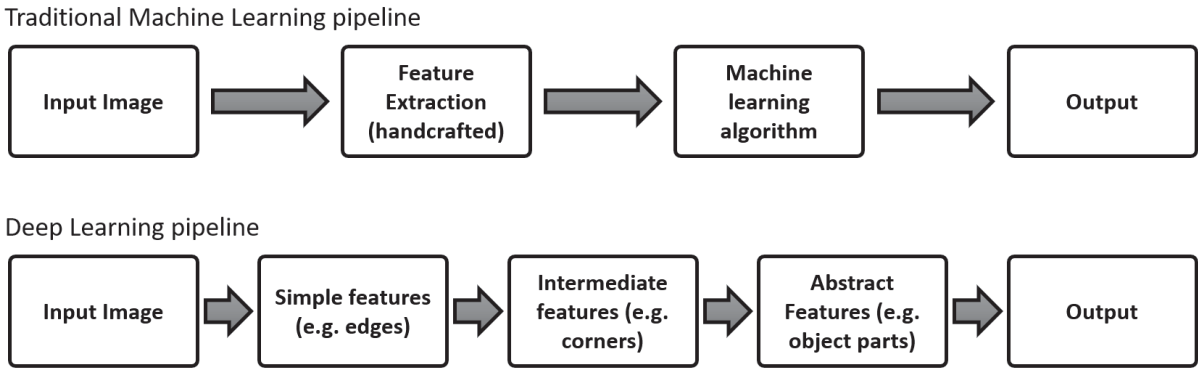


Figure 2.1: Traditional ML vs. Deep Learning pipelines. Steps of feature extraction (in both pipelines) represent generical boxes in the process that could have many sub-steps to be executed. Source: Adapted from Goodfellow et al. (2016)

2.1.1 Machine Learning Basics

Nowadays ML technology fills up many trivial tasks in a modern world: web search filtering, recommendation systems on streaming platforms or e-commerce websites, machine translation,

fraud detection on financial activities, and others (LeCun et al., 2015). According to Murphy (2012), ML is a set of methods that can automatically detect patterns in data and then use unseen patterns to predict future data or perform decisions under uncertainty.

An ML system must be fed by features to detect patterns in data, represented as sequences of information (called feature vectors). Conventional ML systems cannot correctly process raw natural data (for example, the pixel values of an image) and transform into a suitable representation. As an alternative, these systems require a hand-crafted transformation in the input data (LeCun et al., 2015).

The input dataset employed to find this representation is called the training set. To verify if this learned representation is generalizable enough, a test set, not seen in training, must be used. We could categorize ML algorithms into two main groups, the **supervised** and **unsupervised** approaches. The term supervised comes from an instructor showing a student (or a computer) what to do. There is no instructor in unsupervised learning, and the system has to learn without any guidance (Goodfellow et al., 2016).

2.1.1.1 Supervised Learning

In this trend of ML methods, we feed the algorithm with examples of input and output, and *learn* a high-level function that maps the input (x) into the output (y). Given a training set containing N samples, paired as (x_1, y_1), (x_2, y_2), ...,(x_n, y_n), where each y_i was generated from an unknown function $y = f(x)$, the objective is to find a function h that approximates the real function f (Russell and Norvig, 2013).

The function h is a hypothesis. The learning process consists in finding the more suitable solution on the whole space of possible hypotheses, the one that will have an optimal (or nearly) scenario even for unknown samples. Usually, we employ a test set with labeled samples never seen by the system to measure this generalization capacity. Based on a (good) test set, we can verify if a learned hypothesis could predict in a *real world* scenario (Russell and Norvig, 2013).

To search in the hypotheses space, we usually use the Stochastic Gradient Descent (SGD) algorithm. This procedure consists of computing the errors between the output predicted at training time and the expected (labels), compute the average gradient or commonly called **loss** and adjust the model according to it (LeCun et al., 2015). This operation continues processing small sets of training samples until the average of the objective function stops decreasing.

If the output y represents a finite set of values (classes), the learning problem is called **classification**, as an example, we have an image of a handwritten digit, and the output is the digit itself. When the output is a real number, we face a **regression** problem, and an example of it is estimate a house price based on the number of rooms, neighborhood, and building area. Following we have some examples of popular supervised machine learning techniques:

> K**-Nearest Neighbo**r (KNN) is a classifier that outputs the k nearest values from the training set in Euclidean space;
>
> **Linear** and **polynomial regression** techniques output a function that approximated the training set's behavior by using a continuous function;
>
> **Naïve Bayes** classifiers are a family of probabilistic algorithms based on applying Bayes' theorem, assuming that the features are independent;
>
> **Support-vector machines** (SVM) are machine learning models utilized for classification rather than regression analysis (Boser et al., 1992; Cortes and Vapnik, 1995). This

method is one of the most prominent in ML and can explore linear or non-linear problems efficiently.

Decision tree is another group of learning algorithm which breaks the input space into regions and has separate parameters for each region. A more accurate method based on it is the **random forest**. It combines many decision trees fitting each sub-samples of the dataset, presenting a significant improvement in predictive accuracy (Goodfellow et al., 2016).

2.1.1.2 Unsupervised Learning

In unsupervised learning algorithms, the goal is to find patterns on the input without any specific feedback. This type of algorithm divides the dataset into similar patches in a clustering process in traditional ML systems. Unsupervised approaches help to extract information from distribution without requiring human labor to generate the output targets by annotating examples. In the context of deep approaches, unsupervised algorithms play an essential role with autoencoders. We could also learn the entire probability distribution of a dataset, whether explicitly as in density estimation or implicitly for tasks like denoising (Goodfellow et al., 2016).

As the main conventional unsupervised techniques, we have the k-means algorithm that divides the training set into k different clusters near each other, and the Principal Component Analysis (PCA), which learns a representation with a lower dimension concerning the input. Dimensionality reduction is useful to bring linear independence to data, but also for visualization purposes. For a more detailed explanation of both algorithms, please refer to Goodfellow et al. (2016)

Despite the success of supervised approaches, LeCun et al. (2015) believe that unsupervised methods would become far more relevant in a few years. Humans and animals tend to learn in an unsupervised way, e.g., discovering the world surrounding and observing it, and the machines could also explore such evolutionary features.

2.1.1.3 Neural Networks

Artificial Neural Networks (ANN), or just Neural Networks (NN), are computational models that mimic the structure and functions of biological neural networks of many mammals. The main component of an ANN is the artificial neuron, first introduced by McCulloch and Pitts (1943). This artificial neuron maps the behavior and structure of its biological inspiration mathematically. Dendrites are formalized as the multiplication $w_i x_i$ between the axon x, and the synapse w, respectively, called input and weight. The dendrites carry out the signals to the body cell, that process all the inputs plus a bias b by a summation. A function called activation (φ) models the resultant output. The most common activation functions are the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU). A NN can dynamically learn by combining many neurons and changing their weights and bias to control the influence of the neural units.

The simplest type of ANN is the feed-forward NN. The information goes from input to output always forward in this type of network, i.e., without cycles. The arrangement of neurons is made in layers. The first layer's output is connected to the second one's input, going through the layers until the last one, called the output layer. The intermediate layers are named hidden and play an essential role. They can find features within the data and allow the following segments to operate on those features rather than the noisy and large raw data from the input layer. The typical layer on these networks is the fully-connected, in which every neuron is connected pairwisely

between two adjacent layers (Hornik et al., 1989). Figure 2.2 shows a graphical representation of an artificial neuron and an ANN.

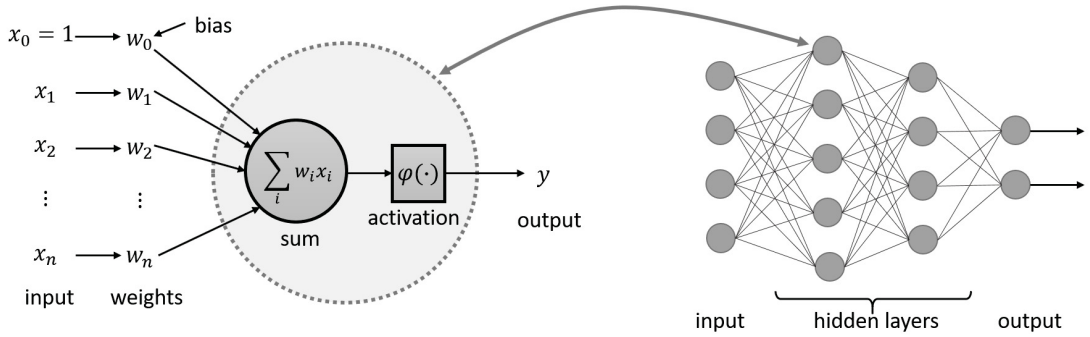


Figure 2.2: Neural Network model. **Left**: Artificial neuron with inputs $(x_i w_i)$ for $i = 0..n$, the summation, the activation units, and the output y. **Right**: A scheme of a feed-forward neural network with two hidden layers and distribution of $[4, 5, 4, 2]$ neurons.

2.1.1.4 Why use deep learning?

Most standard ML algorithms work very well on a wide variety of significant problems. However, problems such as recognizing speech or objects are not straightforward. The development of deep learning was, in part, motivated by the failure of conventional algorithms to generalize well on such tasks.

Data availability has grown considerably in the latest years, but standard techniques' performance has not accompanied this growth. Figure 2.3 shows that as the volume of training data increases, deep networks tend to improve the model's performance, and because of that, we tend to associate deep learning with large datasets (Rosebrock, 2017).

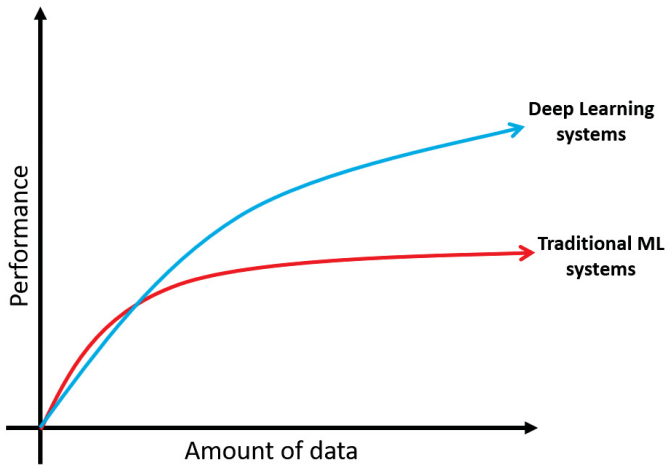


Figure 2.3: Performance vs. amount of data in ML systems. Source: Adapted from (Ng, 2020)

Another challenge lies in generalizing to new examples, that becomes exponentially more complex when working with high-dimensional data. The mechanisms used in traditional machine learning are inadequate to learn complex functions in high-dimensional spaces. Such spaces also frequently impose high computational costs, and the design of deep learning methods tends to overcome these and other obstacles (Goodfellow et al., 2016).

2.1.2 Deep Networks

The main deep-learning-based approaches lie on the Convolutional Neural Networks (CNN) (LeCun et al., 2015). A CNN is a specific type of feed-forward neural network that performs convolutions, exploits, and takes advantage of a grid-like input structure. Examples could be 1D grids, for time series, 2D for pixels of an image, or 3D for point clouds (Goodfellow et al., 2016). Fully-connected deep neural networks demand a massive number of parameters that would most likely drive to overfit or be a computational wasting. Therefore, to overtake this situation, CNNs use convolutional layers that rely on ideas like local receptive fields and shared weights.

A CNN is usually built with two main building blocks, convolutional and pooling layers. These building blocks represent data in terms of features by weighting and modifying smaller patches' identifiable characteristics within each layer's inputs. The overall idea behind using these building blocks is that the input representation is gradually increased in abstraction as it progresses through the layers. Earlier layers contain more specific structural information such as borders, while later layers contain more complex information about how specific objects look.

2.1.2.1 Convolutions

Convolutional networks are as any ANN, i.e., composed of neurons with learnable weights and biases. Each neuron in a convolutional layer receives some inputs and calculates their outputs by learning such parameters. However, there is a significant difference concerning the traditional architectures: the sharing of weights between neurons, known as filters. The convolutional operation consists of sliding a filter over the input. An element-wise multiplication is performed at every location and summed together, yielding the corresponding result in the output feature map. In Equation 2.1 the convolution operation S is expressed for a point (i, j) of a 2D Image (I) and a kernel filter (K) in a discrete domain.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n). \tag{2.1}$$

It is worth noting that the layers in a CNN share their parameters, and thus, all the neurons are capable of detecting the same feature (e.g., a corner), despite the location on input, resulting in a feature map. For example, using a filter of 3×3 and input of 32×32, each neuron on the next convolutional layer will have only 9 weights for that filter instead of 1024 in a fully connected layer.

Each layer detects many features, and this number corresponds to the **depth** of the feature maps. In general, controlling the output volume involves three hyperparameters: the depth mentioned earlier; the **stride**, that is the step of the sliding operation on the convolution; and the **zero-padding** that adds a zero-valued border on the input, to maintain the original input size, if desired. The following formula return the output feature map volume (W_{out}), based on the hyperparameters:

$$W_{out} = \frac{(W_{in} - F + 2P)}{S + 1} \tag{2.2}$$

where W_{in} is the input volume size, F represents the filter size of the convolutional layer, S, and P are the stride and the amount of zero-padding, respectively.

2.1.2.2 *Pooling*

According to Goodfellow et al. (2016), the processing of a typical convolutional layer in a network consists of three stages:

1. Performing convolutions to produce a set of linear activations;
2. Executing a non-linear activation function, such as the ReLU;
3. Applying a pooling function to modify the output of the layer.

A pooling function replaces the net's output at a particular location with a summary statistic of the nearby outputs. For example, the max-pooling (Zhou and Chellappa, 1988) operation reports the maximum output inside a rectangular neighborhood. Other pooling functions include the average, the L^2 norm of a rectangular region, and a weighted average based on the central pixel's distance. Figure 2.4 presents an example of the above-cited convolution and max-pooling operations on a convolutional layer.

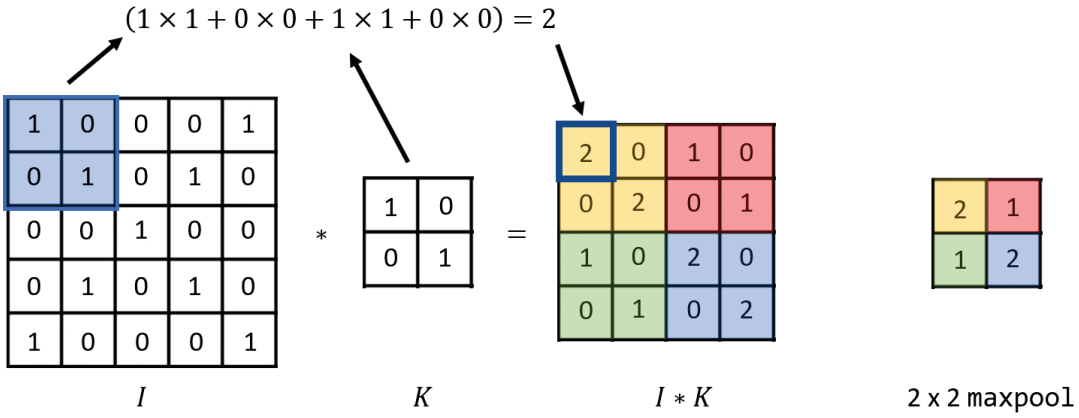


Figure 2.4: Convolution and max-pooling operations. Given an image I and a kernel filter K, the $I * K$ represents the resultant feature map of a convolution. The rightmost image presents a grid outputted by a 2×2 max-pooling operation.

2.1.2.3 *Training a Deep Network*

Training a Deep Network is not a straightforward task. Indeed, besides the data availability and architecture's setup, some factors must be considered. The loss function and optimization algorithms are the first to pick. Moreover, learning rate and decay, weight initialization and decay, and dropout layers are also significant. For more information about these hyperparameters, please refer to Goodfellow et al. (2016).

Despite the high demand for data, and eventually laborious effort in training deep learning architectures, sometimes we do not have enough data to do it. Thus, it is fundamental to find a way to use previously trained weights. We call this process **transfer learning**, which consists of extracting features from the network by forwarding examples, i.e., without training. Another essential process, known as **fine-tuning**, performs an adaptation of a pre-trained network to the context of a different (and potentially smaller) dataset.

2.1.2.4 Invariance and Equivariance

One of the main facts on CNNs success and robustness is learning invariant features from (a sufficient amount of) data. The term invariance refers to the ability of, under a transformation on input, the output remains unchanged. Another relevant property is the equivariance, which means that the object's position does not need to be fixed to be detected by CNN. To clarify, if we have a function $f(x)$ that is equivariant to a function g, so $f(g(x)) = g(f(x))$.

The translation invariance is an inherent ability of convolutional and fully-connected layers. It is provided by parameter sharing, but also by combining pooling layers and striding. It can be an advantageous property if we care more about whether some feature is present than exactly where it is.

Convolution is not naturally equivariant to other transformations, such as changes in scale or rotation. The only strategy to learn rotation and scale invariance is to augment the input dataset and provide more examples of the same objects with distinct sizes and orientations. Some recent studies have shown that a framework named Spherical CNNs unlocks the rotation equivariance "natively" on CNNs (Cohen et al., 2018, 2019; Esteves et al., 2018, 2020).

2.1.3 3D Deep Learning

This section explores some concepts regarding the most prominent technologies to work on 3D data. We start with a brief overview about data representation for deep learning systems, and the following subsections explore some architectures for dealing directly on point clouds with PointNets (Qi et al., 2017a,b), to achieve rotation invariance with Spherical CNNs (Cohen et al., 2018; Esteves et al., 2018), and to describe 3D surfaces by a folding-based strategy (Groueix et al., 2018; Yang et al., 2018b).

2.1.3.1 3D data representation

Different from images that have a dominant representation as a 2D array of pixels, 3D has no direct representation. A point cloud is a set of points in space sampled from object surfaces, usually acquired by 3D sensors such as LiDARs or depth cameras. A polygon mesh is a collection of interconnected surfaces broadly used in computer graphics in 3D modeling and rendering algorithms. Volumetric representation quantifies the space into small voxels, in a regular 3D grid. Finally, we also represent 3D as multiple projected depth views, frequently used for visualizations. Figure 2.5 depicts the above-cited representations on a 3D Model.

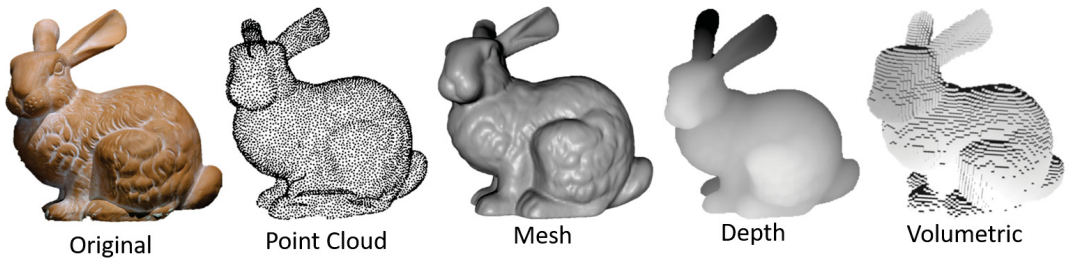


Figure 2.5: 3D data representation. Model Bunny extracted from the Stanford Views dataset (Curless and Levoy, 1996).

2.1.3.2 *PointNets*

Point clouds are irregular data structures, and feed neural networks with them is not an obvious task. Most methods tend to convert to alternative representations such as binary occupancy 3D grid, i.e., one if has a point and zero if it is empty. After that, apply this transformed data to a CNN that works on a volumetric grid, such as 3D ShapeNet (Wu et al., 2015), VoxNet (Maturana and Scherer, 2015), and Volumetric and Multi-view CNNs (Qi et al., 2016). Another characteristic of the point clouds is that they are very sparse structures, thus work with 3D grids may imply a large space and computational cost. Qi et al. (2017a) propose a framework that deals directly with point clouds, called PointNet.

PointNet is a unified architecture that consumes raw point clouds as input, i.e., each point is a 3D coordinate. The output can be either a class label for the entire input or per point segment/part labels for each input point. The authors also propose a variant called PointNet++ (Qi et al., 2017b) that works with the surface normals associated with the points. This network architecture (Figure 2.6) has two main parts: the classification network for object recognition tasks and the segmentation network for per point semantic assignment. PointNet also provides two transform networks (T-Net) to deal with the object's pose and to transform the features.

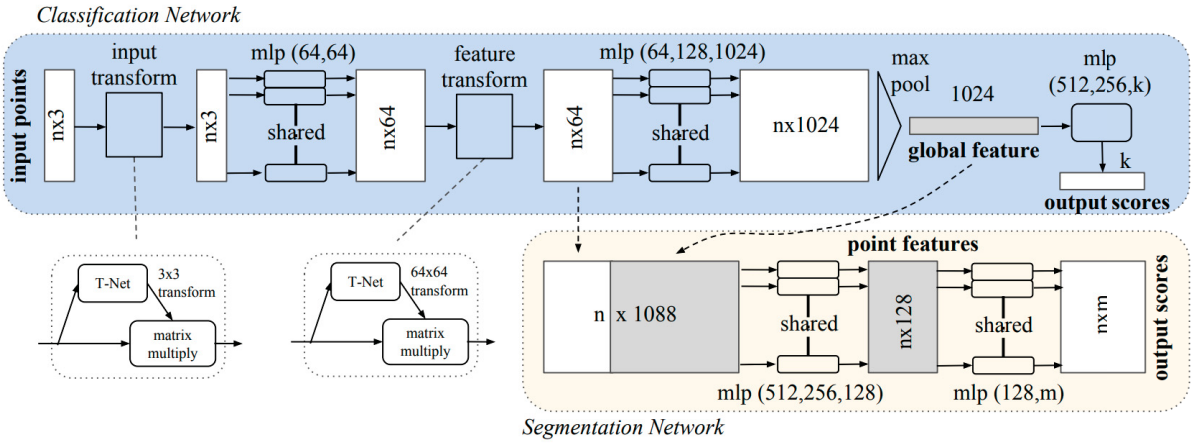


Figure 2.6: PointNet Architecture.
Source: Qi et al. (2017a)

2.1.3.3 *Spherical CNNs*

Previously, we pointed that despite achieving translation equivariance on traditional CNNs, transformations on rotation and scale are essential matters on the 2D CV. 3D information tends to solve scaling obstacles, being rotation invariance achieved by augmenting the input. However, it is hard to impose synthetic transformations on data that reflect real-world situations, ensuring that every rotation will be acknowledged on training. In this section, we present a recent technology called Spherical CNNs, that is rotation-equivariant by nature, and the basis for the developments presented in Chapters 3 and 4. For more in-depth knowledge and mathematical proofing, please refer to Cohen et al. (2018).

Architectures which employ spherical signals and perform spherical convolutions achieve equivariance to rotations, as demonstrated by Cohen et al. (2018). They proposed a framework that, differently from consuming data based on the cartesian space (2D or 3D), uses the Special Orthogonal Group, or $\mathrm{SO}(3)$ for short. The $\mathrm{SO}(3)$ is a 3D manifold comprising rotations about the origin of a 3D Euclidean space $\mathbb{R}^3$ under the operation of composition.

There are several distinct forms to represent rotations in the 3D space. In the context of this work, we consider the approach proposed by Euler. Since rotations in the 3D space have 3 degrees of freedom, only three parameters are necessary to characterize a rotation. These parameters are known as the Euler angles α, β, and γ. Consequently, a full rotation will be the composition of the correspondent transformations (XYZ), presented by Equation 2.3. In this context, we consider the ZYZ-Euler angles, meaning that both the first and the last angles will refer to rotations around the Z axis.

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{2.3}$$

where:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{2.4}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.5}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

The Spherical CNNs' architecture performs convolutions on the $\mathrm{SO}(3)$ space and produces rotation-equivariant feature maps also in these representations. As the resultant signal on the maps lives in $\mathrm{SO}(3)$, the Euler angles can also rotate the feature maps. This process is fundamental in developing rotation-equivariant descriptors, as described in Chapters 3 and 4.

2.1.3.4 Folding-based Networks

Folding-based strategies focus on the use of unsupervised learning algorithms for point clouds. They propose to surpass issues on the point cloud representations using a 2D grid structure and reconstruct the clouds through a plane-folding operation. Assuming that any 3D surface can be transformed into a 2D plane, by cutting, squeezing, and stretching operations, the inverse operation could also be conceivable.

Two seminal works propose similar strategies simultaneously: Fold-Net (Yang et al., 2018b) and AtlasNet (Groueix et al., 2018). Both use an auto-encoder network, producing a bottleneck layer that learns to fit the 2D surface into the 3D data. The produced bottleneck layer's feature map serves as codeword (a.k.a. descriptor) and works as a high-dimensional embedding of an input point cloud. They demonstrate that the folding operations can build an arbitrary surface when provided a correct codeword. At training time, a fully connected decoder network is responsible for updating the codeword weights and reconstruct the input cloud. The reconstructed point cloud outputted by the decoder is then confronted with the input, using the Chamfer Loss. In this thesis, we adopt the AtlasNet (Groueix et al., 2018) as the decoder of our learned descriptor.

The Chamfer distance can assess the similarity between the two sets of samples. Particularly, during the train, we minimize the average of the Euclidean distances between 3D clouds, considering for each input's point the nearest neighbor in the reconstructed one, and *vice*

versa. Let $\mathcal{S}$ be the set of 3D input points belonging to the neighborhood of $\mathbf{p}$ and $\mathcal{S}^\star$ the set of points generated by the decoder. The Chamfer distance can be formulated as

$$\begin{aligned}\mathcal{L}(\mathcal{S},\mathcal{S}^\star)_\theta = &\frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}\in\mathcal{S}}\min_{\mathbf{x}^\star\in S^\star}\|\mathbf{x}-\mathbf{x}^\star\|_2 + \\ &\frac{1}{|\mathcal{S}^\star|}\sum_{\mathbf{x}^\star\in\mathcal{S}^\star}\min_{\mathbf{x}\in\mathcal{S}}\|\mathbf{x}^\star-\mathbf{x}\|_2.\end{aligned} \tag{2.7}$$

The term $\min_{\mathbf{x}\in S}\|\mathbf{x}^\star-\mathbf{x}\|_2$ estimates the precision of the predicted point cloud by measuring the average distance between the predicted points and the closest ground truth one, while $\min_{\mathbf{x}^\star\in S^\star}\|\mathbf{x}-\mathbf{x}^\star\|_2$ quantify how well the predicted point cloud covers the ground truth by measuring the average distance between a ground truth point and the closest predicted one. To enforce that the distance from $\mathcal{S}$ to $\mathcal{S}^\star$ and the distance vice versa have to be small simultaneously, the two terms are summed together.

2.2 3D COMPUTER VISION

With the development of autonomous driving cars, virtual and augmented reality applications, 3D vision problems become more relevant since they provide much richer information than 2D. Applications such as 3D object detection and recognition, 3D pose estimation, and 3D reconstruction rely on feature descriptors. In this chapter, we explore such structures that play a fundamental role in this thesis' proposals. We start exploring acquisition aspects regarding 3D vision systems. We then discourse specifically on the descriptors and present general and specific issues regarding local and global features. This thesis focuses on object recognition and reconstruction using descriptors, addressed in Sections 2.2.3 and 2.2.4. Finally, we present datasets used throughout the thesis to test the proposed approaches.

2.2.1 Acquisition

RGB-D cameras capture color and depth information in real-time. Despite their well-known use on CV applications, they have attracted attention from the delivery of low-cost sensors, like Microsoft Kinect, Intel RealSense, Structure Sensor, and Asus Xtion.

This kind of low-cost sensor presents some limitations that may include the maximum depth capture (from 0.2 to 4.5m), low depth resolution (about 0.3 MP), or even low-quality scannings when capturing information in reflexive and transparent surfaces (Kadambi et al., 2014). In Figure 2.7, we present an example of a scene captured by the Kinect sensor, with significant nuisances of this kind of image.

In the context of this work, we consider mostly images captured by RGB-D sensors and LiDAR. However, other kinds of sensors or techniques can obtain or estimate depth information from the "*real world*". Among them, one could consider the Structure from Motion (SfM) and Stereoscopy. Both use color information to detect the disparity between images and estimate depth. As they use color-only information, they are more sensitive to illumination changes (Pan et al., 2016).

2.2.2 Descriptors

Descriptors play a fundamental role in CV applications, especially involving 3D processing. They are data structures that describe objects or scenes fully or partially. An efficient 3D descriptor

must be highly discriminative and robust under noise, occlusion, and illumination variation. Tombari et al. (2012) cite three kinds of descriptors based on the specificity when working on RGB-D data:

> **Point descriptors**: use position and color information of a 3D point as its descriptor. This approach is simplistic and sensitive to noise and thus rarely used.
>
> **Local descriptors**: describe the neighborhood of a point considering a specific radius. Deal with geometrical and textural information about a portion of a scene/object
>
> **Global descriptors**: process a patch pre-segmented, which is likely an object and generates a description based on its structure.

In the context of this thesis, we explore mostly Local Descriptors. However, in Chapter 6, we apply global features in the object recognition task and present an approach to combine Global and Local features in a hybrid pipeline that detects objects by using global features and estimate the object's pose with locals (Chapter 7). Notwithstanding, we perform a more profound review of the 3D local features, with a more focused review on the globals in Chapter 6.

2.2.2.1 3D Local descriptors

A local 3D descriptor processes a keypoint neighborhood to produce a feature vector discriminative to clutter and robust to noise. In this section, we review the literature concerning such proposals. As the local descriptors are closely related to the keypoint detection algorithms, we start scratching those techniques we considered in this dissertation. Then, we show some approaches regarding the hand-crafted descriptors, and finally, move to modern data-driven methods. To handle the viewpoint variations and attain the invariance to rotation, the descriptors mentioned above rely either on an LRF or RA. Hence, we present also a review concerning the LRF estimators considered in this thesis.

2.2.2.2 Keypoint Extraction

This step concerns selecting some surface points, either from images or point clouds. According to Tombari et al. (2013), keypoint extraction must reduce data dimensionality without losing

Figure 2.7: Example of scene captured by the Kinect sensor. We could see some nuisances present in an image captured by such a low cost sensor: self oclusion, reflexive and transparent surfaces, noisy depth map. On the left, we have the color image and on the right, the depth signal. Black parts represent failures in depth estimation . Source: Images extracted from the Washington RGB-D Scenes dataset (Lai et al., 2014)

Figure 2.8: Keypoint extraction techniques applied on point clouds. From left to right: Original 2.5D model of Mario (extracted from the Bologna Kinect dataset (Salti et al., 2014)), Uniform Sampling with a 1cm and 3cm of radius, and ISS.

discriminative capability. In this work, we explore techniques which work in 3D, as Uniform sampling and Intrinsic Shape Signatures (ISS) (Zhong, 2009), and 2D alike, as SIFT (Lowe, 1999) and FAST (Rosten and Drummond, 2006).

Uniform sampling downsamples point clouds by segmenting it in voxels based on specific leaf size, and selects as keypoint each nearest neighbor point to a voxel centroid (Rusu and Cousins, 2011). **ISS** (Zhong, 2009) selects keypoints based on a local surface saliency criterion, extracting 3D points that have a considerable surface variation in their neighborhood.

The keypoint detector proposed in **SIFT** (Lowe, 1999) is arguably the prominent proposal for RGB images. It is based on detecting blob-like and high contrast local features amenable to compute highly distinctive features and similarity invariant image descriptors. The **FAST** keypoint extractor (Rosten and Drummond, 2006) is a 2D corner detector based on a machine learning approach, widely adopted in real-time CV applications due to its remarkable computational efficiency. Figure 2.8 presents examples of 3D keypoints extractors applied on point clouds.

2.2.2.3 Hand-crafted 3D Local Descriptors

Before the deep learning revolution, scholars have designed hand-crafted functions to abstract the structural information of the 3D keypoints neighborhood, i. e. features, into high-dimensional representations. To this end, the distribution of the features is discretized according to a quantization domain and approximated through histograms. The main proposals differ for the geometric or topological measurements employed (Guo et al., 2016). A local 3D descriptor processes a keypoint neighborhood to produce a feature vector discriminative concerning clutter and robustness to noise. Many descriptors have been proposed in recent years and several works, e.g., Guo et al. (2014b), have investigated their relative merits and limitations. In this thesis, we explore both descriptors which process only depth, or depth and color information, and some of them are briefly explained as follows.

Introduced by Salti et al. (2014), Signatures of Histogram OrienTation (**SHOT**) describes a keypoint based on spatial and geometric information. To calculate the descriptor, we establish an LRF around the keypoint. Then, a canonical spherical grid is divided into 32 segments. Each segment produces a histogram that describes the angles between normals at the keypoint and

normals at the neighboring points. The authors also proposed a variation to work with color at the points, called **CSHOT**. The color value is encoded according to the CIELab color space and added to SHOT's angular information. This descriptor is known to yield better results than SHOT when applied to colored point clouds.

PFHRGB (Rusu et al., 2008) is based on the Point Feature Histogram (PFH) and stores geometrical information by analyzing the angular variation of the normal between each pair of combinations in a set composed by the keypoint and all its k-neighbors. PFHRGB works on RGB and stores the color ratio between the keypoint and its neighbors, increasing its efficiency on RGB-D data (Alexandre, 2012). To speed-up, the descriptor calculation, Rusu et al. (2009) proposed a simplified solution, called **FPFH** (Fast PFH), which considers only the differences between the keypoint and its k-neighbors. An influence weight is also stored, resulting in a descriptor that can be calculated faster while maintaining its discriminative capacity.

2.2.2.4 Learned 3D Local Descriptors

The deep learning paradigm has proven to be the holy grail to elaborate 2D visual data. This success has shifted more attention in designing learned 3D local descriptors (Zeng et al., 2017a; Deng et al., 2018b,a; Khoury et al., 2017; Spezialetti et al., 2019). The typical supervised workflow foresees the adoption of a siamese-style convolutional network (Chopra et al., 2005) and a metric loss (Schultz and Joachims, 2004; Weinberger and Saul, 2009) to teach the network how to pull similar features together, i. e., descriptors for the same 3D point acquired under different viewpoints, while pushing unique features apart. Some studies based on this paradigm are 3DMatch (Zeng et al., 2017a), CGF (Khoury et al., 2017), 3DSmoothNet (Gojcic et al., 2019), and Li et al. (2020a). Unsupervised approaches, instead, propose to employ the latent codeword of an encoder-decoder architecture as a 3D feature descriptor. As examples we have, PPF-FoldNet (Deng et al., 2018a), 3D-PointCapsNet (Zhao et al., 2019), and Spezialetti et al. (2019).

On the other hand, the most recent proposals employ fully convolutional networks (Long et al., 2015) and purely data augmentation to learn a rotation-invariant local feature descriptor for point cloud with a supervised approach. The work of Choy et al. (2019b), named Fully convolutional geometric feature (FCGF), represents the first work in this direction. It adopts sparse convolutions (Choy et al., 2019a) to manage the unorganized structure of point clouds and densely extract a compact local feature embedding. Similarly, D3Feat (Bai et al., 2020) leverages KPConv (Thomas et al., 2019) to perform convolution on raw 3D coordinates and proposes a strategy to predict both a detection score and a feature descriptor at each 3D location in the input point cloud. These methods can extract dense features in just one forward pass. Despite being highly efficient in terms of computation time, they poorly generalize when trained and tested on data containing geometries of different natures, as we show in Chapter 4.

In this thesis, we present three new unsupervised 3D Local descriptors. Two of them, LEAD and SOUND, rely on the Spherical CNNs and provide rotation equivariant approaches. The third proposal, named LEAD-PN, is based on PointNet architecture trained in an invariant fashion. Despite the lower performance concerning the previous, it outperforms most unsupervised approaches on registration applications. Chapter 4 states more details regarding these descriptors. Table 2.1 summarizes the above-cited methods, showing their main characteristics.

Table 2.1: Comparison between 3D local descriptors. *Means that depends on an external hand-crafted LRF

Descriptor	hand-crafted	learned	supervised	unsupervised
SI (Johnson and Hebert, 1999)	✓			
PFH (Rusu et al., 2008)	✓			
FPFH (Rusu et al., 2009)	✓			
USC (Tombari et al., 2010)	✓			
RoPs (Guo et al., 2013b)	✓			
SHOT (Salti et al., 2014)	✓			
CGF (Khoury et al., 2017)		✓	✓	
3DMatch (Zeng et al., 2017a)		✓	✓	
PPFNet (Deng et al., 2018b)		✓		✓
PPF-FoldNet (Deng et al., 2018a)		✓		✓
PointCaps3D (Zhao et al., 2019)		✓		✓
3DSmoothNet (Gojcic et al., 2019)		✓	✓	
Spezialetti et al. (2019)		✓		✓*
FCGF (Choy et al., 2019b)		✓	✓	
D3Feat (Bai et al., 2020)		✓	✓	
Li et al. (2020a)		✓	✓	
LEAD (Ours)		✓		✓*
LEAD-PN (Ours)		✓		✓*
SOUND (Ours)		✓		✓

2.2.2.5 Local Reference Frame

The definition of a canonical pose of a point cloud has been studied mainly in the field of local feature descriptors (Johnson and Hebert, 1999; Rusu et al., 2009; Guo et al., 2013b; Salti et al., 2014). Indeed, the definition of a robust LRF is:

$$\mathcal{R}(p) = \{\hat{\mathbf{x}}(p), \hat{\mathbf{y}}(p), \hat{\mathbf{z}}(p) \mid \hat{\mathbf{y}} = \hat{\mathbf{z}} \times \hat{\mathbf{x}}\} \tag{2.8}$$

The LRF estimation of the local neighborhood of a keypoint p is crucial to create rotation-invariant local features. An LRF works efficiently if, and only if, is perfectly equivariant to rotations of a keypoint. Hence, repeatability refers to the ability of an LRF in aligning the same keypoint under different viewpoints. Figure 2.9 presents an example of an LRF being extracted from different views of the same object. Note that we may face different nuisances in both views, such as self-occlusion.

Several works define the local canonical system's axes as eigenvectors of the 3D covariance matrix between points within a spherical region of radius r centered at p. As the signs of the eigenvectors are not repeatable, some works focus on the disambiguation of the axes: Mian et al. (2010), RoPS (Guo et al., 2013b), and SHOT-lrf (Salti et al., 2014). Another family of methods leverages the normal to the surface at p, i.e., $\hat{n}(p)$, fix the $\hat{\mathbf{z}}$ axis, and then exploit geometric attributes of the shape to identify a reference direction on the tangent plane to define the $\hat{\mathbf{x}}$ axis: Point signatures (Chua and Jarvis, 1997), Board (Petrelli and Di Stefano, 2011), FLARE (Petrelli and Di Stefano, 2012), TOLDI (Yang et al., 2017), and GFrames (Melzi et al., 2019).

All the methods previously pointed extract hand-crafted information to estimate the LRF. At the best of our knowledge, just one method, named LRf-net (Zhu et al., 2020), proposes to extract orientation from data, but by using hand-crafted preprocessing of the input. In Chapter

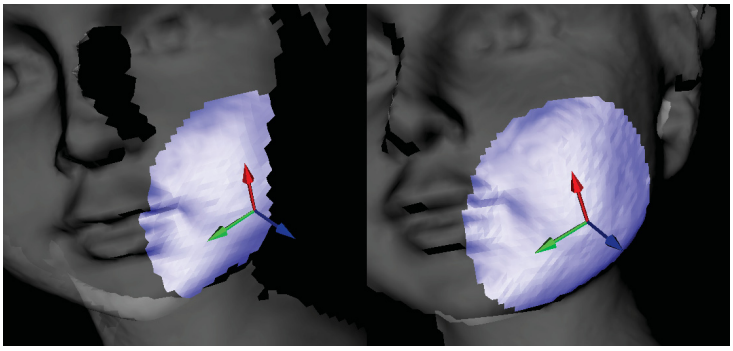


Figure 2.9: LRF repeatability example. Both images show a LRF estimated for a keypoint with a fixed search radius (lighter points). Green, blue, and red arrows represent respectively the x, y and z axis. Source: Petrelli and Di Stefano (2012)

3, we present **Compass**, a method that differs sharply from previous methods because it learns the cues necessary to canonically orient a surface without making a priori assumptions on which details of the underlying geometry may be effective to define a repeatable canonical pose. Table 2.2 present state-of-the-art methods and the respective classification.

Table 2.2: Comparison between LRF estimation methods. CA refers to *Covariance Analysis*-based approaches, and GA refers to *Geometric Attributes*. *Uses hand-crafted preprocessing

Method	CA	GA	hand-crafted	learned
Point signatures (Chua and Jarvis, 1997)	✓		✓	
EM (Novatnack and Nishino, 2008)	✓		✓	
Mian (Mian et al., 2010)	✓		✓	
Board (Petrelli and Di Stefano, 2011)		✓	✓	
FLARE (Petrelli and Di Stefano, 2012)		✓	✓	
RoPS (Guo et al., 2013b)		✓	✓	
SHOT-lrf (Salti et al., 2014)	✓		✓	
TOLDI (Yang et al., 2017)		✓	✓	
GFrames (Melzi et al., 2019)		✓	✓	
LRF-net (Zhu et al., 2020)				✓*
Compass (Ours)				✓

2.2.3 3D Object Recognition

Recognition systems work with objects, which are digital representations of tangible real-world items that exist physically in a scene. This kind of system is unavoidably an ML-based approach. Thus there are two main big stages to be fulfilled: training and testing. We can see this division in Figure 2.10, which represents a generic scheme of such systems.

In the training phase, we build a database of objects representing our system's knowledge base, which will serve as a reference to the classification process. Models are stored as complete structures of objects (Full-3D) or partial views (2.5D) and described following their geometric and texture features. In the test phase, we extract scene elements and search for corresponding objects that we have previously got in training. The recognition of an object occurs when the database features are compatible with those extracted on the scene, called feature matching. It may be necessary to perform some post-processing and verification steps to refine the results.

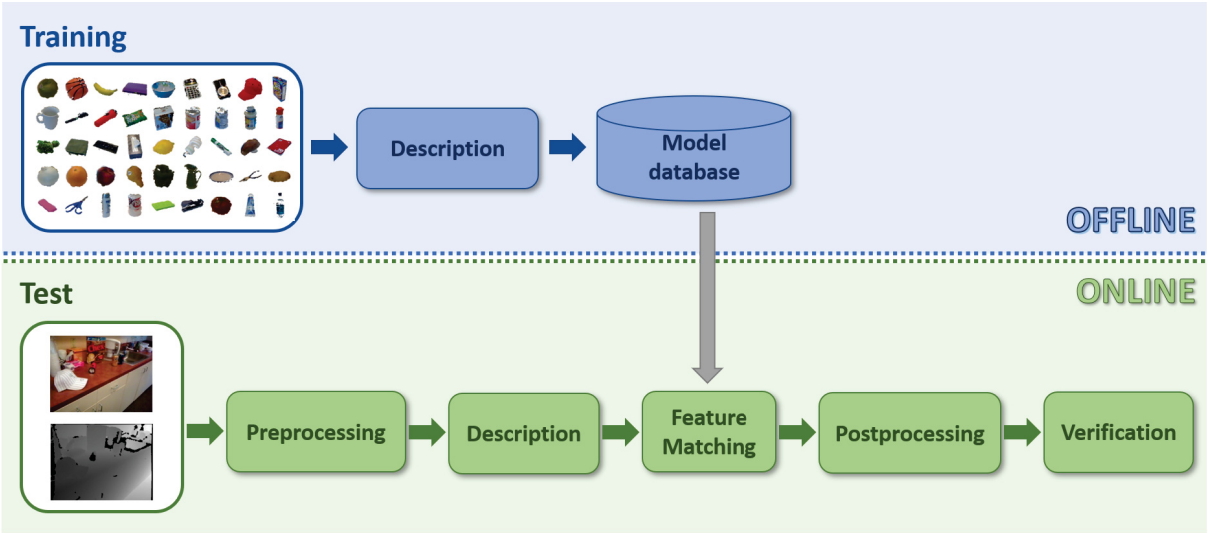


Figure 2.10: Block diagram of a generic pipeline for object recognition systems. The blue-shaded part refers to the training stage (offline), and the green refers to the test stage (online)

Real-time RGB-D based applications can face an enormous amount of processing data. These sensors capture 30 fps[1] RGB and Depth information at 640×480 pixels, generating a bandwidth of more than 30 MB/s[2]. Reducing this data volume is crucial in such applications, and strategies attempt to mitigate it. Sections 2.2.3.1 and 2.2.3.2 report more deeply such strategies that could work with local or global descriptors.

The employment of local or global descriptors in object recognition systems requires different steps on the respective pipeline. Figure 2.11 depicts each of them, and Sections 2.2.3.1 and 2.2.3.2 describe their steps in detail. These pipelines correspond to instances of the test phase on the generic pipeline presented in Figure 2.10

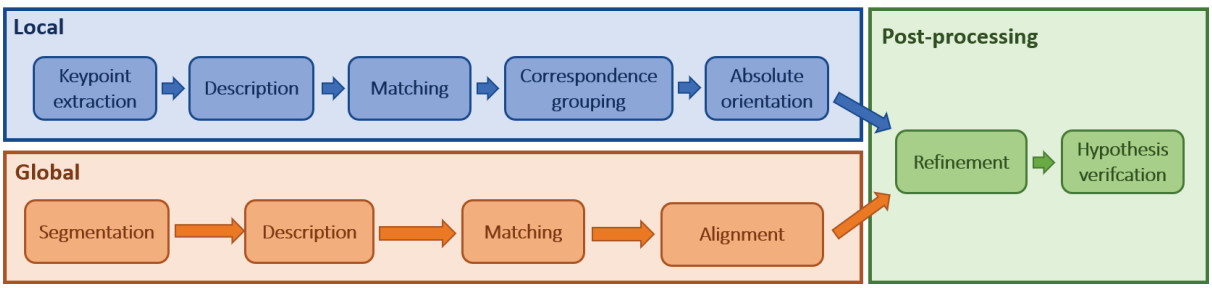


Figure 2.11: Object recognition pipeline based on 3D Local (blue-shaded parts) and Global (orange-shaded parts) features. Source: Adapted from Aldoma et al. (2012a)

2.2.3.1 Local descriptors pipeline

Local descriptors are representations of the neighborhood of a keypoint, producing a feature vector associated. These descriptors are well-known by the capability of dealing excellently with some nuisances, such as noise and occlusions. An object recognition system demands a sufficient number of described keypoints to be effective. Thus, one main drawback of this approach lies in the high computational power and memory consumption required. The object

[1]Frames per second

[2]Considering 30fps and 300k points of RGB-D information (4 bytes). Point clouds may use 16 bytes per point, increasing this number considerably, to more than 140MB/s.

recognition pipeline, proposed by Aldoma et al. (2012a) and depicted in Figure 2.11, shows a series of execution steps, starting from an input point cloud, and described as follows.

The **Keypoint extraction** step selects sample points of an object or scene, intending a dimensionality reduction. The detected keypoints must be robust to the viewpoint, noise, and scale variations. The main characteristics of a keypoint detector are repeatability and distinctiveness. The former can detect the same keypoints under pose/viewpoint variation, and the latter concerns the capacity to discriminate and group objects (Tombari et al., 2013). The most prominent algorithm is the ISS (Zhong, 2009), but more straightforward approaches, as Uniform and Random Sampling, are often used in object recognition applications.

The **Description** stage consists of generating a feature vector (also called descriptor) representing the neighborhood of a keypoint. A good descriptor must extract topological information, considering different point densities, noise levels, and occlusion (Tombari et al., 2013).

After calculating the descriptors in the scene and object's points, the **Matching** is responsible for finding the corresponding object's points in the scene. Commonly this process is done by similarity, selecting the NN in a high-dimensional feature space. This task is very computationally costly, and optimized procedures like FLANN (Fast Library for Approximate Nearest Neighbor), proposed by Muja and Lowe (2009), provide excellent results (Aldoma et al., 2012a).

The matching process results in a set of all point-to-point correspondences between objects and scenes. Assuming that a rigid transformation exists, the **Correspondence grouping** rejects every correspondence that is not geometrically consistent between the object and the scene (Aldoma et al., 2012a). The main approaches are the Geometric consistency grouping (CGF), in Chen and Bhanu (2007) and Hough Voting (Tombari and Di Stefano, 2010).

Despite the correspondence grouping stage's efficiency, some groups are eventually not consistent with a unique 6DoF pose. Therefore, in the **Absolute orientation**, we perform an additional step, based on the RANSAC (RANdom SAmple Consensus) algorithm, to eliminate those correspondences not consistent with the same pose (Aldoma et al., 2012a).

In short, the whole process (Figure 2.11) consists of giving an object and a scene. The pipeline will return the most likely 6DoF transformation that localizes and aligns the referred object on the scene.

2.2.3.2 Global descriptors pipeline

Analogously to the local, the global descriptors also describe the topography of objects to recognize them in a scene. However, instead of keypoints, they represent a whole object or cluster of points that are likely to be objects. In terms of the amount of memory and processing, they are very friendly. Simultaneously, this kind of descriptor is more sensitive to occluded objects and more dependent on a reliable segmentation.

The pipeline concerning global descriptors is slightly different from the local alternative. In the following, we present the steps, according to the classification proposed by Aldoma et al. (2012a), and depicted in Figure 2.11.

As explained first, global descriptors demand the use of complete objects. Hence, the scene **Segmentation** process is a critical step in it. Simple but sometimes efficient algorithms segment a scene into clusters, which then will be classified as objects. According to Aldoma et al. (2012a), the grouping of points based on a specific distance threshold is a useful approach but lies on parameter choice. Other initiatives include the use of sliding window (Redmon et al., 2016; Liu et al., 2016) for images or sliding shapes (Song and Xiao, 2014, 2016) for point clouds.

Due to the advances of these methods, we can segment objects efficiently in real-time using these technologies.

The **Description** stage is similar to the local alternative and returns a feature vector that describes the geometry and texture of point clusters. After getting the descriptor, differently from local descriptors, which lies on similarity approaches, the **Matching** phase for global descriptors can be done using more robust machine learning techniques, such as those presented in subsubsection 2.1.1.1.

Most of the global methods are invariant to rotation, so an additional process is required. The **Alignment** step consists of adding a process called camera roll histogram (Aldoma et al., 2012a). Despite this approach's efficiency, the pose estimation presents a lower performance concerning the local descriptors pipeline.

2.2.4 3D Scene Registration

Registration is a fundamental building block in 3D point cloud-based applications. The use of point cloud registration includes computer graphics, robotics, cultural heritage modeling, digital archaeology, architecture, and several CV applications. Such applications include object modeling, tracking, and simultaneous localization and mapping (SLAM).

3D registration consists in aligning two or more point clouds, and it is fundamental for 3D modeling. The main task is to find the relative pose between views, acquired from different viewpoints. After the alignment, the objective is to fuse them into a single point cloud so that subsequent processing steps, such as segmentation and object reconstruction, can be applied (Holz et al., 2015).

In this thesis, we exploit pairwise registration, which consists of finding a global transformation between two overlapped views, i.e., with common areas. Pairwise registration relies on approaches based on the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992; Chen and Medioni, 1992).

As an optimization algorithm, the effectivity of the ICP depends on its initialization. The ICP delivers a reliable alignment between the views, and a well-selected initial hypothesis transformation speeds its convergence and considerably improves its effectivity. There are two main approaches to obtaining a transformation between two point clouds: feature-based registration or dense registration. Figure 2.12 presents a pipeline that coverage both situations.

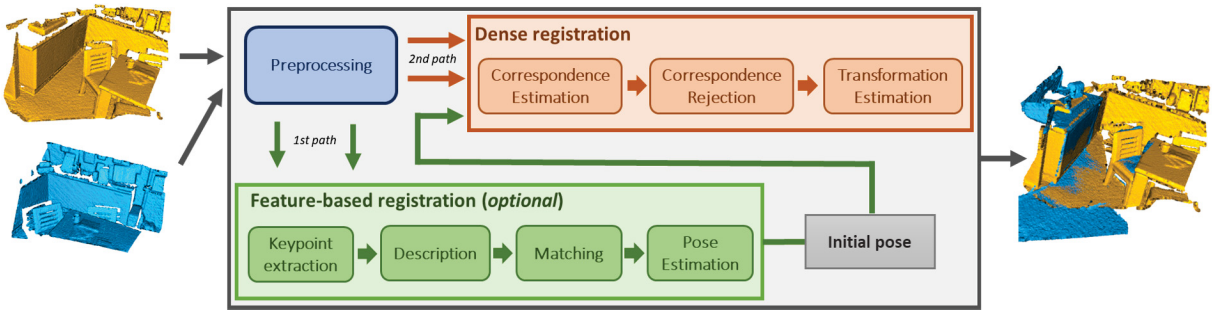


Figure 2.12: Registration pipeline. First path involves an optional pre-alignment step (the green-shaded area), and second path (orange-shaded steps) corresponds to the dense registration phase., which could input a coarse registration from the 1st step or execute independently. Source: Adapted from Holz et al. (2015)

The pipeline execution starts from a pair of views, yet in memory, and the **Preprocessing** stage seeks to enhance the input clouds, remove acquisition noise, or extract information about the surface, such as normals. The first path (colored in green) is optional and involves processing local descriptors to estimate a coarse initial transformation. In this pathway, we face similar steps

to the object recognition pipeline. Processing steps like **Keypoint extraction**, **Description**, and **Matching** are indistinguishable in both pipelines. Geometric feature descriptors are computed and matched in a high-dimensional space. The more descriptive, unique, and persistent these descriptors are, the higher the likelihood that the resultant **Pose estimation** closely aligns both views.

The second path is responsible for the dense registration via ICP and considers the closest corresponding points in the Cartesian space. Initial pose information, yielded by the first path, could be used as input to ICP, or it can estimate the transformation at its own. However, executing only the second path could be computationally expensive, depending on the input clouds' size.

The **Correspondence estimation** consists of determining corresponding points in both clouds, and computing the transformation that aligns them, in a process iteratively repeated following a convergence criterion. Following the pipeline, the next level is responsible for **Correspondence rejection**. This step consists of filtering the point pairs matched in the previous stage to help the **Transformation estimation** algorithm toward convergence to a global minimum. Several approaches are suitable for this step, and the most prominent are those based on distances (Rusinkiewicz and Levoy, 2001) and RANSAC (Fischler and Bolles, 1981). Finally, the last stage aims to find a transformation estimation that minimizes the Euclidean distance between found pairs of closest points using least-squares error (Besl and McKay, 1992). Another approach, proposed by Chen and Medioni (1992) uses a similar intuition, but, minimizing the distance between points using a point-to-plane metric.

In short, the whole process consists of given two point clouds with a degree of overlap between them, estimate a global transformation that put both views in the same global reference system, and thus align them. In a perfect overlap scenario, the presented pipeline converges to the global minimum. However, partially overlapped clouds or a weak initialization may imply in getting a local minimum. Hence, we can see the importance of developing effective local descriptors.

2.2.5 Datasets

To measure the accuracy of distinct techniques is fundamental to have standard sets of data and metrics. The scientific community provides datasets for this purpose with ground-truth annotations. In this section, we present the datasets used in this thesis to evaluate our proposals for Object Recognition and Registration applications using 3D data.

2.2.5.1 Object Recognition

In this section, we present the datasets employed in this thesis in object recognition evaluations. We consider datasets of real data, with the Washington RGB-D and Bologna Kinect and hand-made CAD models, with ModelNet and ShapeNet.

Washington RGB-D Object and Scenes: proposed by Lai et al. (2011a) from the University of Washington, the RGB-D Object contains a collection of 300 instances of household objects, grouped in 51 distinct categories. Each object includes a set of views, captured from different viewpoints with a Kinect sensor. A collection of 3 images, including RGB, depth, and mask, is presented for each view. In total, this dataset has about 250 thousand distinct images. The authors also provide a dataset of scenes, named RGB-D Scenes. This evaluation dataset has eight video sequences of every-day environments. A Kinect sensor positioned at a human eye-level height acquires all the images at a 640×480 resolution. This dataset is related to the first one,

composed of 13 of the 51 object categories on the Object dataset. These objects are positioned over tables, desks, and kitchen surfaces, in a cluttered fashion with viewpoints and occlusion variation, and have annotation at category and instance levels. A bidimensional bounding box represents the ground-truth of each object's position. Figure 2.13 presents examples of both datasets.

Figure 2.13: Examples of models and scenes from the Washington RGB-D Scenes dataset (top row), and objects from the RGB-D Object dataset (bottom row). Source: Adapted from Lai et al. (2011a).

Bologna: another object recognition dataset employed in our tests is the Kinect dataset from the University of Bologna, proposed by Tombari et al. (2010). This dataset has sixteen scenes and six models with pose annotation. Each model is represented as a set of 2.5D views from different angles and has from thirteen to twenty samples. Figure 2.14 depicts some examples of models and scenes in this dataset.

Figure 2.14: Examples of models and scenes from the Kinect dataset (Salti et al., 2014). Models of Rabbit, Frog, Mario, and Doll (center), and scenes.

ModelNet e ShapeNet: the ModelNet40 (Wu et al., 2015) is a shape classification benchmark. This dataset has 12,311 CAD models from 40 human-made object categories, and it is splitted into 9,843 for training and 2,468 for testing. In our trials, we use point clouds sampled from the original CAD models provided by the authors of PointNet (Qi et al., 2017a). ShapeNet (Chang et al., 2015) is also a collection of CAD models. The dataset presented in Figure 2.15, has 55 annotated categories, but in this thesis, we use only to provide qualitative results of three of them by orienting clouds with our LRF proposal Compass, in Chapter 3.

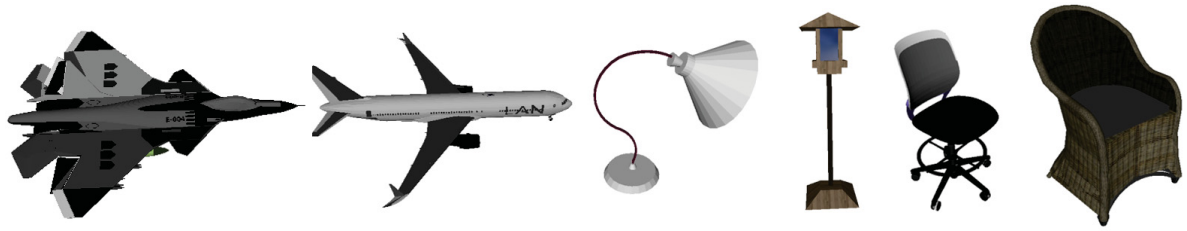


Figure 2.15: Examples of models from the ShapeNet dataset. From left to right: Models of airplane, lamp, and chair.

2.2.5.2 Registration

In this section, we present the datasets that we utilize to test our proposals for registration. We consider, for indoor scenarios, the 3DMatch, and outdoor, the ETH datasets. Both datasets provide a collection of *scenes*, composed by a set of 2.5D scans where a single scan depicts a small area of the whole environment and therefore is called *fragment*. Figures 2.16 and 2.17 present some examples of fragments of these datasets.

3DMatch: this dataset (Zeng et al., 2017a) is a collection of a large part of the publicly available datasets such as Analysis-by-Synthesis (Valentin et al., 2016), 7-Scenes (Shotton et al., 2013), SUN3D (Xiao et al., 2013), RGB-D Scenes v.2 (Lai et al., 2014) and BundleFusion (Dai et al., 2017), which contains 62 scenes. Following the standard protocol for learned descriptors we train and validate on 54 scenes while running comparison at test time only on the remaining 8. The fragments used for training and testing, are derived from the fusion of 50 consecutive depth frames of an RGB-D sensor (Deng et al., 2018a). Similarly, the *Rotated* 3DMatch benchmark is generated by rotating all the fragments of the test split around randomly sampled axes and angles over the entire space of rotations (Deng et al., 2018a). Both datasets provide accurate ground-truth transformations for performance evaluation purposes. This dataset is the *de-facto* standard for the evaluation of learned 3D descriptors, together with its *Rotated* variation, to assess the invariance to the rotation of learned methods.

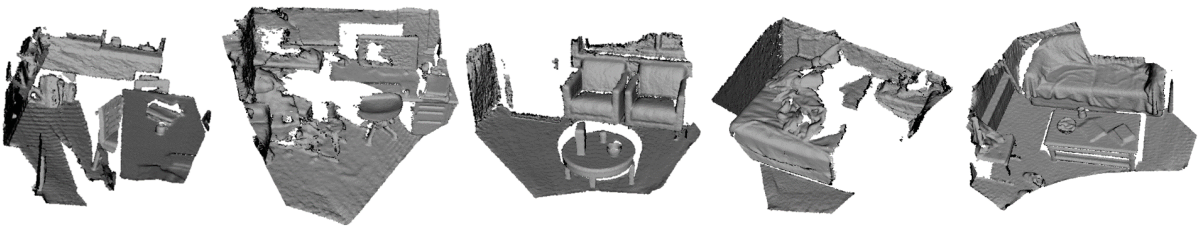


Figure 2.16: Examples of fragments from the 3DMatch Benchmark dataset. From left to right: Scenes extracted from the 7-scenes, BundleFusion, RGB-D Scenes, SUN3D, and Analysis-by-synthesis.

ETH: this dataset (Pomerleau et al., 2012) is a challenging outdoor dataset, which presents 8 sequences of sparse and dense vegetation (e. g., trees and bushes) acquired with a *laser scanner* sensor under different seasonal changes. This dataset does not have an explicit train/test splits. In this thesis we compare our methods likewise Gojcic et al. (2019), i.e., considering only on a subset of 4 scenes named: *Gazebo-Summer*, *Gazebo-Winter*, *Wood-Autumn*, and *Wood-Summer* in a transfer learning approach. This dataset, presented in Figure 2.17, was introduced by Gojcic et al. (2019) to evaluate the robustness of the generalization of learned descriptors on outdoor scenarios.

Figure 2.17: Examples of fragments from the ETH dataset. The two leftmost images represent the *wood_summer* scenes, and the rightmost the *gazebo_winter*. Reading the point clouds is not an easy task, but we can see in the wood scene the path at the center, and in the gazebo scene, there are the gazebo pillars and the background trees.

Stanford Views: This dataset contains 252 real scans of 4 objects (Armadillo, Bunny, Buddha, Dragon), from the Stanford 3D Scanning Repository (Curless and Levoy, 1996), acquired with a Cyberware 3030 MS scanner. It is a variation on the original dataset, created by the CV Lab at the University of Bologna (Petrelli and Di Stefano, 2012). This dataset is composed of meshed models, so in our experiments, we preserve only the original model's vertices, obtaining a point cloud representation of them. Figure 2.18 shows the models of this dataset. Since there is no training-test split, we could neither proceed to unsupervised learning nor weakly supervised learning. Ground-truth transformations are available for all the scans.

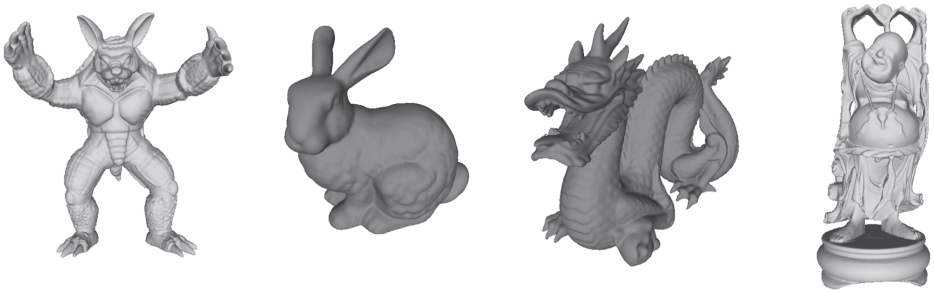


Figure 2.18: Examples of models from the Stanford Views dataset. From left to right: Armadillo, Bunny, Dragon, and Buddha.

2.3 FINAL REMARKS AND OVERVIEW

We start this chapter, presenting a basic overview of the deep learning aspects employed in this thesis. We started scratching on the ML concepts, such as supervised and unsupervised approaches, and why these techniques are in a continuous update for deep learning alternatives. Secondly, we have explored ideas about deep learning methods and the hyperparameters involved in the training process. Finally, we have shown more specifically, strategies on processing 3D point clouds in deep architectures, particularly on the PointNets, Spherical CNNs, and Folding-based approaches.

After, we explored more focused aspects of CV related to 3D applications, starting from the acquisition process of RGB-D images. This work will focus on object recognition and reconstruction applications, intimately related to feature matching. Therefore, we present concepts and relevant techniques on the local and global descriptors and the generic and specific pipelines regarding their use on real applications. Furthermore, regarding the descriptors, we presented the registration pipeline we use in this thesis. Finally, we have given the datasets we will use to demonstrate our proposed methods' effectiveness in this thesis.

3 LEARNING TO ORIENT SURFACES

Humans naturally develop the ability to mentally portray and reason about objects in what we perceive as their *neutral*, *canonical* pose, and this ability is critical for correctly recognizing and manipulating objects as well as reasoning about the environment. Indeed, mental rotation abilities have been extensively studied and linked with motor and spatial visualization abilities since the 70s in the experimental psychology literature (Shepard and Metzler, 1971; Vandenberg and Kuse, 1978; Jansen and Kellner, 2015).

Robotic and CV systems similarly require neutralizing pose variations when processing 3D data and images in many vital applications such as grasping, navigation, surface matching, augmented reality, shape classification, and detection. In these domains, two main approaches have been pursued to define pose-invariant methods to process 3D data: pose-invariant operators and canonical pose estimation. Pioneering work applying deep learning to point clouds, such as PointNet (Qi et al., 2017a,b), achieved invariance by sampling the range of all possible poses at training time through data augmentation. This approach, however, does not generalize to poses not seen during training. Hence, invariant operators like rotation-invariant convolutions were introduced, allowing to train on a reduced set of poses (ideally one, the real data) and test on the full spectrum of rotations (Masci et al., 2015; Esteves et al., 2018; You et al., 2018; Zhang et al., 2019b; Rao et al., 2019; Zhang et al., 2019a).

Instead, canonical pose estimation follows the human path to invariance more closely and exploits the geometry of the surface to estimate an intrinsic 3D reference frame that rotates with the surface.

Transforming the input data by the inverse of the 3D orientation of such a reference frame brings the surface in a pose-neutral, canonical coordinate system wherein pose invariant processing and reasoning can happen. While humans have a preference for a canonical pose matching one of the usual poses in which they encounter an object in everyday life, in machines this paradigm does not need to favor any actual reference pose over others: as illustrated in Figure 3.1, an arbitrary one is fine as long as it can be repeatably estimated from the input data.

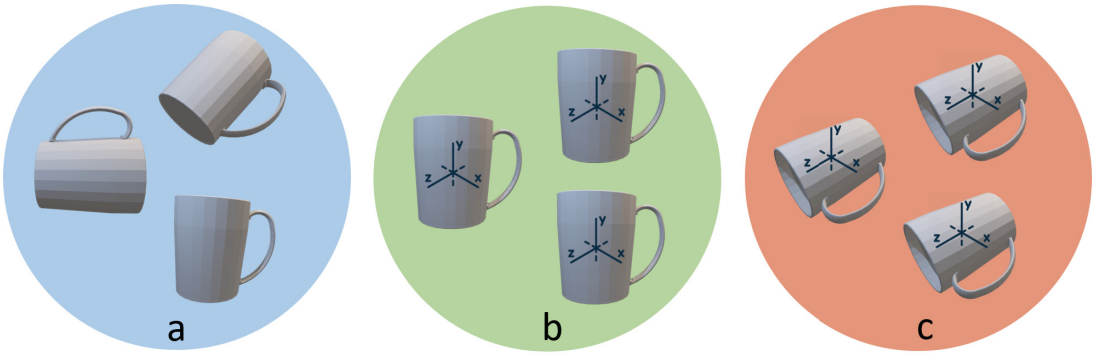


Figure 3.1: Canonical poses in humans and machines. Randomly rotated mugs are depicted in (a). To achieve rotation-invariant processing, e. g.to check if they are the same mug, humans mentally neutralize pose variations preferring an *upright* canonical pose, as illustrated in (b). A machine may instead use any canonical reference pose, even unnatural to humans, e. g.like in (c).

Despite mental rotation tasks being solved by a set of unconscious abilities that humans learn through experience, and the huge successes achieved by deep neural networks in addressing analogous unconscious tasks in vision and robotics, the problem of estimating a canonical pose

is still solved solely by hand-crafted proposals (Salti et al., 2014; Petrelli and Di Stefano, 2011; Melzi et al., 2019; Guo et al., 2013b; Yang et al., 2017; Gojcic et al., 2019; Aldoma et al., 2012c).

This may be due to convolutional networks, the standard architectures for vision applications, reliance on the convolution operator in Euclidean domains, which possesses only the property of *equivariance* to translations of the input signal. However, the essential property of a canonical pose estimation algorithm is equivariance with respect to 3D rotations because, upon a 3D rotation, the 3D reference frame, which establishes the canonical pose of an object, should undergo the same rotation as the object. We also point out that, although, in principle, estimation of a canonical reference frame is suitable to pursue pose neutralization for whole shapes, in past literature it has been intensively studied mainly to achieve a rotation-invariant description of local surface patches.

In this chapter, we explore the feasibility of using deep neural networks to learn to pursue rotation-invariance by estimating the canonical pose of a 3D surface, be it either a whole shape or a local patch.

Purposely, we propose to leverage Spherical CNNs (Cohen et al., 2018; Esteves et al., 2018), a recently introduced variant of convolutional networks that possess the property of equivariance w.r.t. 3D rotations by design, in order to build Compass. This self-supervised methodology learns to orient 3D shapes.

As the proposed method computes feature maps living in $\mathrm{SO}(3)$, i. e., feature map coordinates define 3D rotations, and does so by rotation-equivariant operators, any salient element in a feature map, e. g., its arg-max, may readily be used to bring the input point cloud into a canonical reference frame.

However, due to discretization artifacts, Spherical CNNs turn out to be not perfectly rotation-equivariant (Cohen et al., 2018). Moreover, the input data may be noisy and, in the case of 2.5D views sensed from 3D scenes, affected by self-occlusion and missing parts. To overcome these issues, we propose a robust end-to-end training pipeline that mimics sensor nuisances by data augmentation and allows gradients' calculation with respect to feature maps coordinates.

We evaluate Compass on two challenging tasks. The first one is estimating a canonical pose of local surface patches, a key step in creating rotation-invariant local 3D descriptors (Salti et al., 2014; Gojcic et al., 2019; Yang et al., 2017). In the second task, the canonical pose provided by Compass is instead used to perform highly effective rotation-invariant shape classification by leveraging a simple PointNet classifier.

3.1 PROPOSED APPROACH

Our problem can be formalized as follows. Given the set of 3D point clouds, $\mathcal{P}$, and two point clouds $\mathcal{V}, \mathcal{T} \in \mathcal{P}$, with $\mathcal{V} = \{p_{\mathcal{V}_i} \in \mathbb{R}^3 \mid p_{\mathcal{V}_i} = (x, y, z)^T\}$ and $\mathcal{T} = \{p_{\mathcal{T}_i} \in \mathbb{R}^3 \mid p_{\mathcal{T}_i} = (x, y, z)^T\}$, we indicate by $\mathcal{T} = R\mathcal{V}$ the application of the 3D rotation matrix $R \in \mathrm{SO}(3)$ to all the points of $\mathcal{V}$. We then aim at learning a function, $g : \mathcal{P} \rightarrow \mathrm{SO}(3)$, such that:

$$\mathcal{V}_c = g(\mathcal{V})^{-1} \cdot \mathcal{V} \tag{3.1}$$

$$g(\mathcal{T}) = R \cdot g(\mathcal{V}). \tag{3.2}$$

We define the rotated cloud, $\mathcal{V}_c$, in (3.1) to be the canonical, pose-neutral version of $\mathcal{V}$, i. e., the function g outputs the inverse of the 3D rotation matrix that brings the points in $\mathcal{V}$ into their canonical reference frame. (3.2) states the equivariance property of g: if the input cloud is

rotated, the output of the function should undergo the same rotation. As a result, two rotated versions of the same cloud are brought into the same canonical reference frame by (3.1).

Due to the equivariance property of Spherical CNNs layers, upon rotation of the input signal, each feature map does rotate accordingly. This means that one could track any distinctive feature map value to establish a canonical orientation satisfying (3.1) and (3.2). Indeed, defining as Φ the composition of S^2 and $\mathrm{SO}(3)$ correlation layers in our network, if the last layer produces the feature map $[\Phi(f_{\mathcal{V}})]$ when processing the spherical signal $f_{\mathcal{V}}$ for the cloud $\mathcal{V}$, the same network will compute the feature map $[L_R\Phi(f_{\mathcal{V}})] = [\Phi(L_R f_{\mathcal{V}})] = [\Phi(f_{\mathcal{T}})]$ when processing the rotated cloud $\mathcal{T} = R\mathcal{V}$, with spherical signal $f_{\mathcal{T}} = L_R f_{\mathcal{V}}$.

Hence, if for instance we select the maximum value of the feature map as the distinctive value to track, and the location of the maximum is at $Q_{\mathcal{V}}^{max} \in \mathrm{SO}(3)$ in $\Phi(f_{\mathcal{V}})$, the maximum will be found at $Q_{\mathcal{T}}^{max} = RQ_{\mathcal{V}}^{max}$ in the rotated feature map. Then, by letting $g(\mathcal{V}) = Q_{\mathcal{V}}^{max}$, we get $g(\mathcal{T}) = RQ_{\mathcal{V}}^{max}$, which satisfies (3.1) and (3.2).

Therefore, we realize function g by a Spherical CNN and we utilize the $\arg\max$ operator on the feature map computed by the last correlation layer to define its output. In principle, equivariance alone would guarantee to satisfy (3.1) and (3.2). Unfortunately, while for continuous functions the network is exactly equivariant, this does not hold for its discretized version, mainly due to feature map rotation, which is exact only for bandlimited functions (Cohen et al., 2018). Moreover, equivariance to rotations does not hold for altered versions of the same cloud, e. g., when a part of it is occluded due to viewpoint changes. We tackle these issues using a self-supervised loss computed on the extracted rotations when aligning a pair of point clouds to guide the learning, and an ad-hoc augmentation to increase the robustness to occlusions. Using a soft-argmax layer, we can back-propagate the loss gradient from the estimated rotations to the positions of the maxima we extract from the feature maps and to the filters, which overall lets the network learn a robust g function.

3.1.1 Learning from Spherical Signals

The spherical correlation operator is defined for signals living on the unit sphere, S^2. Thus, before sending it through the network, the geometry around a feature point must be converted into a spherical representation. A possible solution adopted in Cohen et al. (2018) and Esteves et al. (2018) is to project a 3D mesh onto an enclosing discretized sphere using a raycasting scheme. However, in our particular case, the input data is not a regular watertight mesh, but a point cloud that corresponds to the neighborhood of the point we wish to describe, i. e. a *keypoint*. Similarly to You et al. (2018), we first convert the 3D points into a spherical coordinate system and then construct a quantized grid in this new coordinate system. The i-th cell in the resulting grid is identified with three spherical coordinates $(\alpha[i], \beta[i], d[i]) \in S^2 \times D$ where $\alpha[i]$ and $\beta[i]$ represent the azimuth and inclination angles of its center and $d[i]$ is the distance from the sphere center. The K-valued spherical signal $f : S^2 \to \mathbb{R}^K$ is then composed by K concentric spheres corresponding to the number of subdivisions along the distance axis, each sphere encodes the density of the points within each cell $(\alpha[i], \beta[i])$ at a distance $d[k]$ from the feature point. To consider the non-uniform spacing in the spherical space, cells near the south or north pole are wider in spherical coordinates, as discussed in You et al. (2018). The above-described process is applied to every keypoint of the surface we choose to describe. Starting from the spherical signal, an equivariant bottleneck is learned by forwarding the signal to the network to encode the information about the local geometry around each keypoint.

3.1.2 Training pipeline

An illustration of the Compass training pipeline is shown in Figure 3.2. Our objective is to strengthen the equivariance property of the Spherical CNN during training, such that the locations selected on the feature maps by the $\arg\max$ function vary consistently between rotated versions of the same point cloud. To this end, we train our network with two streams in a Siamese fashion (Chopra et al., 2005). In particular, given $\mathcal{V}$, $\mathcal{T} \in \mathcal{P}$, with $\mathcal{T} = R\mathcal{V}$ and R a known random rotation matrix, the first branch of the network computes the aligning rotation matrix for $\mathcal{V}$, $R_{\mathcal{V}} = g^{-1}(\mathcal{V})$, while the second branch the aligning rotation matrix for $\mathcal{T}$, $R_{\mathcal{T}} = g^{-1}(\mathcal{T})$. As the feature maps on which the two maxima are extracted should be perfectly equivariant, it would follow that $R_{\mathcal{T}} = RR_{\mathcal{V}} = R^{\star}_{\mathcal{T}}$.

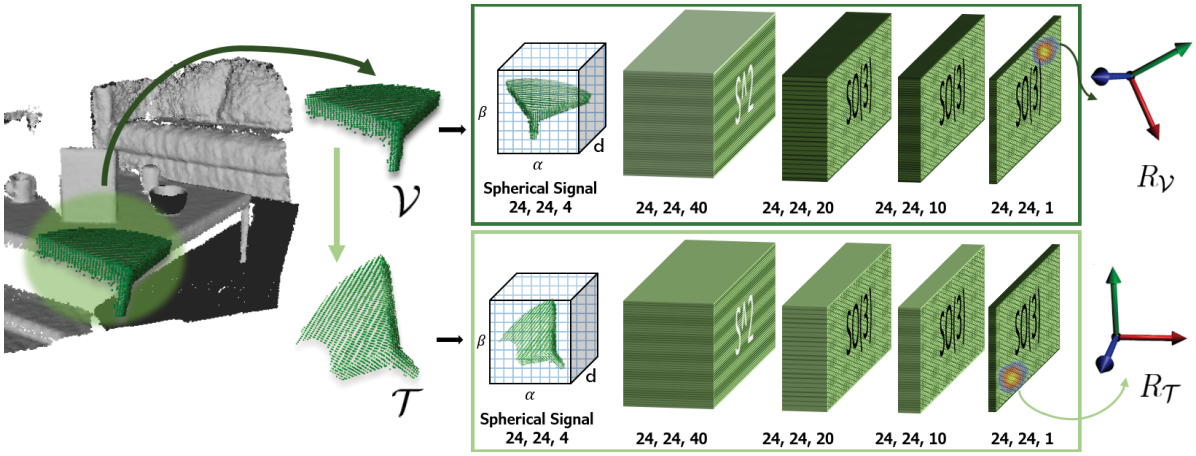


Figure 3.2: Training pipeline of Compass. We illustrate the pipeline for local patches, but the same apply for point clouds representing full shapes. During training we apply the network on a randomly extracted 3D patch, $\mathcal{V}$, and on its augmented version, $\mathcal{T}$, in order to extract the aligning rotation $R_{\mathcal{V}}$ and $R_{\mathcal{T}}$, respectively. At test time only one branch is involved. The numbers below the spherical signal indicate the number of cells along α, β and d, while the triplets under the layers indicate input bandwidth, output bandwidth and number of channels.

For that reason, the degree of misalignment of the maxima locations can be assessed by comparing the actual rotation matrix predicted by the second branch, $R_{\mathcal{T}}$, to the ideal rotation matrix that should be predicted, $R^{\star}_{\mathcal{T}}$. Thus, we can cast our learning objective to minimize a loss measuring the distance between these two rotations.

A natural geodesic metric on the $\mathrm{SO}(3)$ manifold is given by the angular distance between two rotations (Hartley et al., 2013). Indeed, any element in $\mathrm{SO}(3)$ can be parametrized as a rotation angle around an axis. The angular distance between two rotations parametrized as rotation matrices R and S is defined as the angle that parametrizes the rotation SR^T and corresponds to the length along the shortest path from R to S on the $\mathrm{SO}(3)$ manifold (Hartley et al., 2013; Huynh, 2009; Mahendran et al., 2017; Zhou et al., 2019). Thus, our loss is given by the angular distance between $R_{\mathcal{T}}$ and $R^{\star}_{\mathcal{T}}$:

$$\mathcal{L}(R_{\mathcal{T}}, R^{\star}_{\mathcal{T}}) := \cos^{-1}\left(\frac{tr({R_{\mathcal{T}}}^T R^{\star}_{\mathcal{T}}) - 1}{2}\right). \tag{3.3}$$

where tr refers to **trace**, i. e., the sum of the diagonal elements of the rotation matrix.

3.1.3 Network Architecture

The network architecture comprises an S^2 layer followed by three $\mathrm{SO}(3)$ layers, with bandwidth $B = 24$, and the respective number of output channels are set to 40, 20, 10, 1. The spherical input signal is computed with $K = 4$ channels.

3.1.4 Soft-argmax

The result of the $\arg\max$ operation on a discrete $\mathrm{SO}(3)$ feature map returns the location i, j, k along the α, β, γ dimensions corresponding to the ZYZ Euler angles, where the maximum correlation value occurs. To optimize the loss in (3.3), the gradients w.r.t. the i, j, k locations of the feature map where the maxima are detected have to be computed. To render the $\arg\max x$ operation differentiable we add a soft-argmax operator (Honari et al., 2018; Chapelle and Wu, 2010) following the last $\mathrm{SO}(3)$ layer of the network. Let us denote as $\Phi(f_{\mathcal{V}})$ the last $\mathrm{SO}(3)$ feature map computed by the network for a given input point cloud $\mathcal{V}$. A straightforward implementation of a soft-argmax layer to get the coordinates $C_R = (i, j, k)$ of the maximum in $\Phi(f_{\mathcal{V}})$ is given by

$$C_R(\mathcal{V}) = \text{soft-argmax}(\tau\Phi(f_{\mathcal{V}})) = \sum_{i,j,k} \mathrm{softmax}(\tau\Phi(f_{\mathcal{V}}))_{i,j,k}(i, j, k), \tag{3.4}$$

where $\mathrm{softmax}(\cdot)$ is a 3D spatial softmax. The parameter τ controls the temperature of the resulting probability map and (i, j, k) iterates over the $\mathrm{SO}(3)$ coordinates. A soft-argmax operator computes the location $C_R = (i, j, k)$ as a weighted sum of all the coordinates (i, j, k) where the weights are given by a softmax of a $\mathrm{SO}(3)$ map Φ. Experimentally, this proved not to be useful. As a more robust solution, we scale the softmax output according to the distance of each (i, j, k) bin from the feature map $\arg\max$. To let the bins near the $\arg\max$ contribute more in the final result, we smooth the distances by a Parzen function (Parzen, 1962) yielding a maximum value in the bin corresponding to the $\arg\max$ and decreasing monotonically to 0 .

3.1.5 Learning to handle occlusions

In real-world settings, rotation of an object or scene (i. e., a viewpoint change) naturally produces occlusions to the viewer. Recalling that the second branch of the network operates on $\mathcal{T}$, a randomly rotated version of $\mathcal{V}$, it is possible to improve the robustness of the network to real-world occlusions and missing parts by augmenting $\mathcal{T}$. A simple way to handle this problem is to randomly select a point from $\mathcal{T}$ and delete some of its surrounding points. In our implementation, this augmentation happens with an assigned probability. $\mathcal{T}$ is divided in concentric spherical shells, with the probability for the random point to be selected in a shell increasing with its distance from the center of $\mathcal{T}$. Additionally, the number of removed points around the selected point is a bounded random percentage of the cloud's total points. An example can be seen in Figure 3.3.

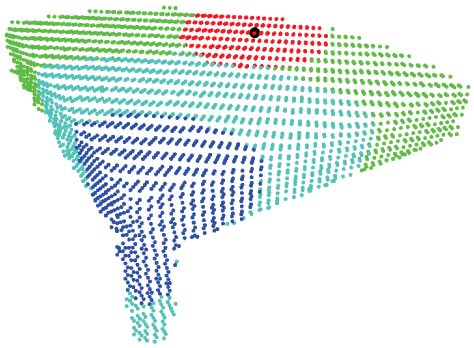


Figure 3.3: Example of the augmentation proposed to handle occlusions. Local support of a keypoint depicting the corner of a table, divided in 3 shells. Randomly selected point in black; removed points in red.

3.2 CANONICAL POSE OF LOCAL SURFACE PATCHES

On local surface patches, we evaluate Compass through the LRF's repeatability (Petrelli and Di Stefano, 2011; Melzi et al., 2019), by estimating at corresponding keypoints in different views of the same scene. All the datasets provide several 2.5D scans, i. e., fragments, representing the same *model*, i. e., an object or a scene depending on the dataset, acquired from different viewpoints.

All N fragments belonging to a test model can be grouped into pairs, where each pair $(\mathcal{F}_s, \mathcal{F}_t)$, $\mathcal{F}_s = \{p_{s_i} \in \mathbb{R}^3\}$ and $\mathcal{F}_t = \{p_{t_i} \in \mathbb{R}^3\}$, has an area of overlap. A set of correspondences, $\mathcal{C}_{s,t}$, can be computed for each pair $(\mathcal{F}_s, \mathcal{F}_t)$ by applying the known rigid ground-truth transformation, $G_{t,s} = \left[R_{t,s}|t_{t,s}\right] \in SE(3)$, which aligns $\mathcal{F}_t$ to $\mathcal{F}_s$ into a common reference frame. $\mathcal{C}_{s,t}$ is obtained by uniformly sampling points in the overlapping area between $\mathcal{F}_s$ and $\mathcal{F}_t$. Finally, the percentage of repeatable LRFs, $\mathit{Rep}_{s,t}$, for $(\mathcal{F}_s, \mathcal{F}_t)$, can be calculated as follows:

$$\mathit{Rep}_{s,t} = \frac{1}{|\mathcal{C}_{s,t}|} \sum_{k=1}^{|\mathcal{C}_{s,t}|} \mathrm{I}\Big(\big(\hat{\mathbf{x}}(p_{s_k}) \cdot R_{t,s}\hat{\mathbf{x}}(p_{t_k}) \geq \rho\big) \wedge \big(\hat{\mathbf{z}}(p_{s_k}) \cdot R_{t,s}\hat{\mathbf{z}}(p_{t_k}) \geq \rho\big) \Big), \tag{3.5}$$

where $\mathrm{I}(\cdot)$ is an indicator function, $(\cdot)$ denotes the dot product between two vectors, and ρ is a threshold on the angle between the corresponding axes, 0.97 radians in our experiments, as proposed in (Petrelli and Di Stefano, 2012) . *Rep* measures the percentage of reference frames which are aligned, i. e., differ only by a small angle along all axes, between the two views. The final value of *Rep* for a given model is computed by averaging on all the pairs.

3.2.1 Test-time adaptation

Due to the self-supervised nature of Compass, it is possible to use the test set to train the network without incurring in data snooping, since there is no external ground-truth information involved. This test-time training can be carried out very quickly, right before the test, to adapt the network to unseen data and increase its performance, especially in transfer learning scenarios. This is common practice with self-supervised approaches (Luo et al., 2020).

3.2.2 Experimental setup

We train Compass on 3DMatch following the standard procedure of the benchmark, with 48 scenes for training and 6 for validation. From each point cloud, we uniformly pick a keypoint every 10 cm, the points within 30 cm are used as a local surface patch and fed to the network.

Once trained, the network is tested on the test split of 3DMatch. The network learned on 3DMatch is also tested on ETH and Stanford Views, using different radii for accounting for the different sizes of the models in these datasets: respectively 100 cm and 1.5 cm. We also apply test-time adaptation on ETH and Stanford Views: the test set is used for a quick 2-epoch training with a 20% validation split, right before being used to assess the performance of the network. We use Adam (Kingma and Ba, 2014) as the optimizer, with 0.001 as the learning rate when training on 3DMatch and for test-time adaptation on Stanford Views, and 0.0005 for adaptation on ETH. We compare our method with recent and established LRFs proposals: GFrames (Melzi et al., 2019), TOLDI (Yang et al., 2017), a variant of TOLDI recently proposed in Gojcic et al. (2019) that we refer to here as 3DSN, FLARE (Petrelli and Di Stefano, 2012), and SHOT (Salti et al., 2014). For all methods, we use publicly available implementations. However, the implementation provided for GFrames could not process the large point clouds of 3DMatch and ETH due to memory limits, and we can show results for GFrames only on Stanford Views.

3.2.3 Results

Table 3.1 reports repeatability on the 3DMatch test set. Compass outperforms the most competitive baseline FLARE, with larger gains over the other baselines. Results reported in Table 3.2 for Stanford Views and Table 3.3 for ETH confirm the advantage of a data-driven model like Compass over hand-crafted proposals: while the relative rank of the baselines changes according to which of the assumptions behind their design fits better the traits of the dataset under test, with SHOT taking the lead on ETH and the recently introduced GFrames on Stanford Views, Compass consistently outperforms them. Remarkably, this already happens when using pure transfer learning for Compass, i. e., the network trained on 3DMatch: despite of the large differences in acquisition modalities and shapes of the models between training and test time, Compass has learned a robust and general notion of canonical pose for a local patch. This is also confirmed by the slight improvement achieved with test-time augmentation, which sets the new state of the art on these datasets.

Results for 3DMatch are shown in Table 3.1: the performance gain achieved by Compass when deploying the proposed data augmentation validates its importance. Indeed, without the proposed augmentation, FLARE performs better than Compass on this dataset.

Table 3.1: LRF repeatability on the 3DMatch dataset. Best result for each row in bold.

	LRF Repeatability (*Rep* ↑)					
	SHOT	FLARE	TOLDI	3DSN	Compass (w/o aug)	Compass
Kitchen	0.189	**0.330**	0.171	0.181	0.274	0.315
Home 1	0.251	0.354	0.243	0.236	0.370	**0.397**
Home 2	0.226	0.339	0.213	0.214	0.353	**0.365**
Hotel 1	0.194	**0.385**	0.213	0.216	0.347	0.370
Hotel 2	0.193	**0.405**	0.223	0.226	0.349	0.393
Hotel 3	0.240	0.407	0.261	0.276	0.406	**0.446**
Study	0.186	0.351	0.195	0.192	0.307	**0.356**
Lab	0.220	0.310	0.198	0.223	0.360	**0.361**
Mean	0.212	0.360	0.215	0.220	0.346	**0.375**

Differently, when tested in transfer learning on Stanford Views, Compass achieves state-of-the-art performance even when not using data augmentation in training, as reported in Table 3.2. The positive effect of augmentation is confirmed, as deploying it significantly improves the overall repeatability even on this dataset.

Table 3.2: LRF repeatability on the Stanford Views dataset. Best result for each row in bold.

	LRF Repeatability (*Rep* ↑)							
	SHOT	FLARE	TOLDI	3DSN	GFrames	Compass(w/o aug)	Compass	Compass(adapted)
Armadillo	0.127	0.185	0.156	0.141	0.168	0.311	0.340	**0.359**
Buddha	0.134	0.194	0.202	0.192	0.181	0.295	0.312	**0.344**
Bunny	0.106	0.379	0.232	0.172	0.426	0.358	0.440	**0.463**
Dragon	0.161	0.207	0.201	0.188	0.251	0.343	0.352	**0.384**
Mean	0.132	0.241	0.197	0.173	0.256	0.326	0.361	**0.388**

The same observations of Stanford Views can be made on the ETH dataset, Table 3.3, where, however, the gain provided by the augmentation is smaller on average. By analyzing the single scenes, we can see that augmentation is beneficial on the Gazebo fragments (both winter and summer), while detrimental on Wood scenes.

Table 3.3: LRF repeatability on the ETH dataset. Best result for each row in bold.

	LRF Repeatability (*Rep* ↑)						
	SHOT	FLARE	TOLDI	3DSN	Compass (w/o aug)	Compass	Compass (adapted)
Gazebo Summer	0.293	**0.345**	0.241	0.241	0.291	0.337	0.330
Gazebo Winter	0.266	0.268	0.170	0.196	0.286	0.292	**0.303**
Wood Autumn	0.253	0.210	0.157	0.174	0.304	0.288	**0.307**
Wood Summer	0.279	0.236	0.171	0.198	0.329	0.314	**0.329**
Mean	0.273	0.264	0.185	0.202	0.303	0.308	**0.317**

3DMatch rotated is a synthetically rotated version of the 3DMatch dataset. This dataset has been specifically proposed to verify the invariance to rotations of the learned 3D descriptors (Deng et al., 2018a) and contains only a test split. In Table 3.4, we show a comparison between the repeatability obtained with Compass on 3DMatch and 3DMatch rotated. Thanks to the equivariance property of the Spherical CNNs, we can achieve similar performance on both datasets.

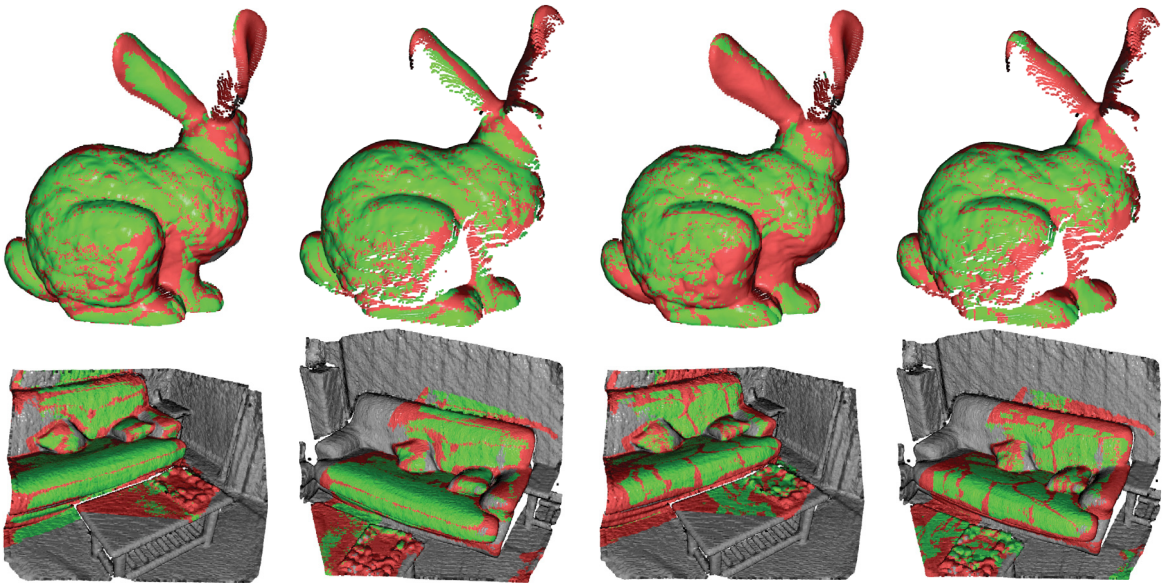
Table 3.4: LRF repeatability of Compass on 3DMatch and 3DMatch rotated.

	LRF Repeatability (*Rep* ↑)	
	Compass (3DMatch rotated)	Compass (3DMatch)
Kitchen	0.312	0.315
Home 1	0.391	0.397
Home 2	0.359	0.365
Hotel 1	0.361	0.370
Hotel 2	0.383	0.393
Hotel 3	0.447	0.446
Study	0.348	0.356
Lab	0.355	0.361
Mean	0.369	0.375

3.2.4 Qualitative results dealing with orienting local surface patches

This section provides qualitative results to show the effectiveness of Compass at computing the canonical pose for local surface patches. Given a pair of fragments, we visualize in both fragments at each point, the accuracy of the estimated LRF using two different metrics. In particular, in Figure 3.4, we show the repeatability of the estimated LRFs (as defined in the main paper), in Figure 3.5 the angular distance between two rotations used as the loss to train our network. In both figures, we visualize the results yielded by Compass alongside FLARE, which offers high performance across all datasets and is the runner-up on 3DMatch

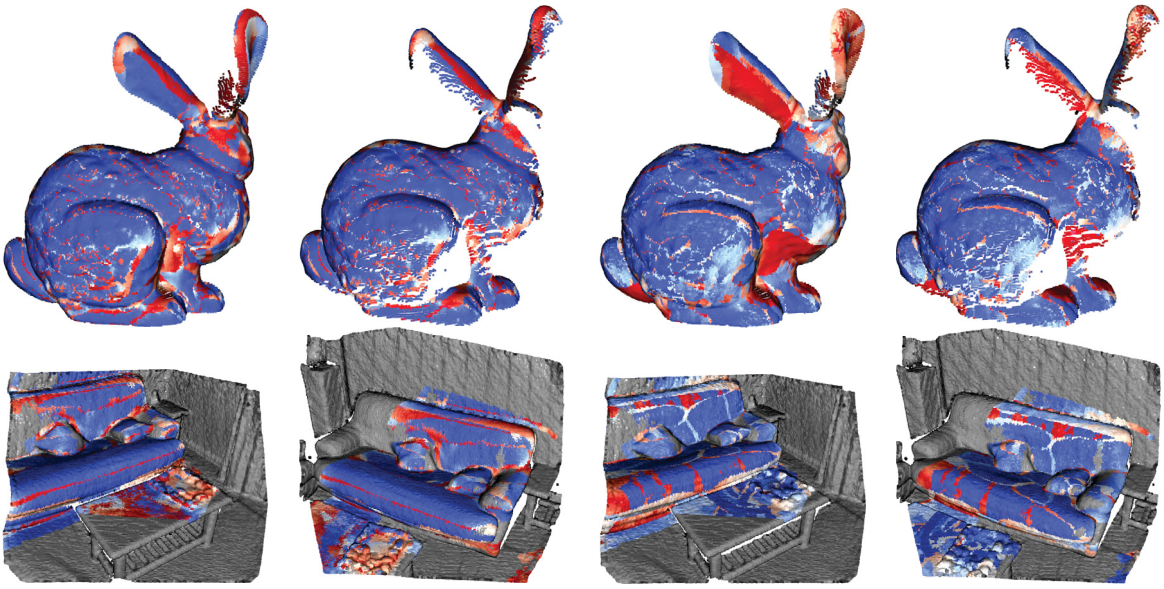
We can observe how Compass tends to yield larger areas in which the LRFs are accurately estimated, i. e., either green or blue ones, depending on the considered metric. It is worth pointing out how this is particularly evident across those challenging fragment areas affected by large missing parts in one of the two views, like, e. g. the left ear of the bunny in the fragments taken from the Stanford Views dataset.



(a) Compass (b) FLARE

Figure 3.4: Qualitative results on the repeatability of Compass and FLARE. Visualization of repeatability at corresponding points of two fragments, with repeatable LRFs in green, non-repeatable ones in red and non-overlapping areas in gray. First row: a pair of fragments from Stanford Views, second row: a pair of fragments from 3DMatch. (a) and (b): results yielded by Compass and FLARE, respectively.

We also provide in Figure 3.6, examples of how Compass orient fragment's patches. We can see that our proposal performs in canonicalize randomly rotated clouds of different portions of a fragment extracted from 3DMatch.



(a) Compass (b) FLARE

Figure 3.5: Qualitative results on the angular error of Compass and FLARE. Visualization of the angular error between the LRFs estimated at corresponding points of two fragments, with lower errors in blue, higher errors in red and non-overlapping areas in gray. First row: a pair of fragments from Stanford Views. Second row: a pair of fragments from 3DMatch. (a) and (b): results yielded by Compass and FLARE, respectively.

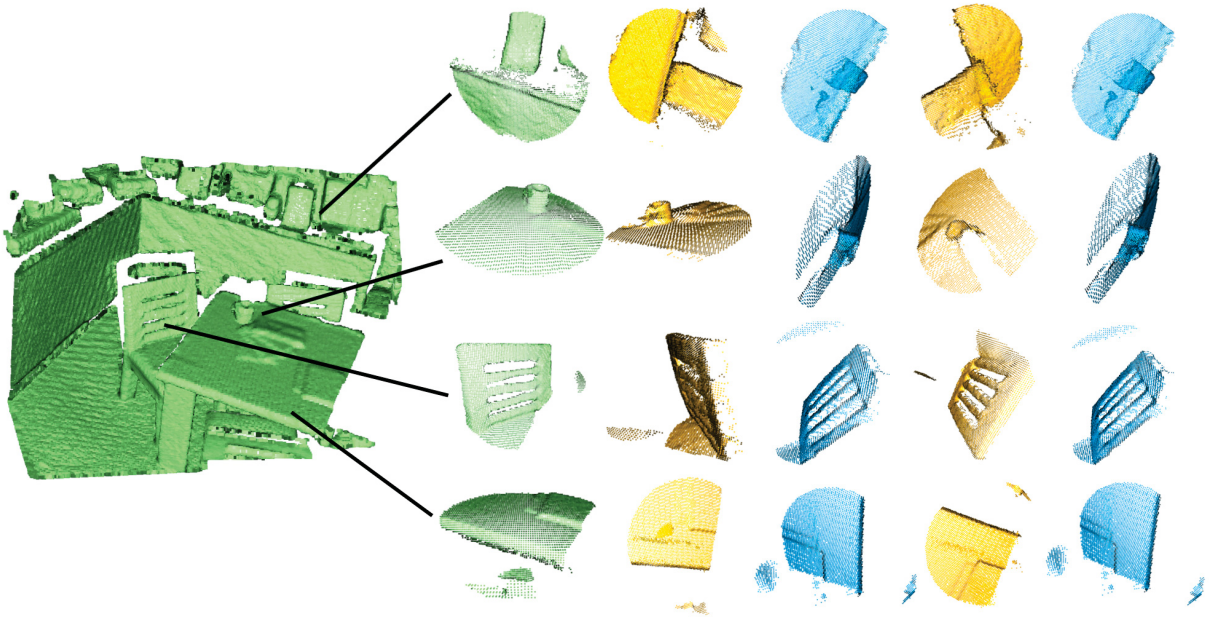


Figure 3.6: Qualitative results of Compass on the 3DMatch benchmark. Examples of patches extracted from the *red kitchen* scene. Black lines show the region on extraction of the patches (green clouds). Yellow clouds represent randomly rotated versions of the original patches and light blue, these clouds oriented by Compass

3.3 ROTATION-INVARIANT SHAPE CLASSIFICATION

Object classification is a central task in CV applications, and the main nuisance that methods processing 3D point clouds have to withstand is rotation. To show our proposal's general applicability and further assess its performance, we wrap Compass in a shape classification pipeline. Hence, in this experiment, Compass is used to orient full shapes rather than local patches. To stress the importance of correct pose neutralization, as shape classifier we rely on a simple PointNet (Qi et al., 2017a), and Compass is employed at the training and the testing phases to canonically orient shapes before sending them through the network.

3.3.1 Experimental setup

We train Compass on ModelNet40 using 8,192 samples for training and 1,648 for validation. Once Compass is trained, we train PointNet following the settings in (Qi et al., 2017a), disabling t-nets, and rotating the input point clouds to reach the canonical pose learned by Compass. We followed the protocol described in (You et al., 2018) to assess rotation-invariance of the selected methods: we do not augment the dataset with rotated versions of the input cloud when training PointNet; we then test it with the original test clouds, i. e., in the canonical pose provided by the dataset, and by arbitrary rotating them. We use Adam (Kingma and Ba, 2014) as the optimizer, with 0.001 as the learning rate.

Table 3.5: Classification accuracy on the ModelNet40 dataset when training with no rotation augmentation. NR column reports the accuracy attained when testing on the cloud in the canonical pose provided by the dataset and AR column when testing under arbitrary rotations. Best result for each column in bold.

Classification Accuracy (*Acc.* %)		
Method	NR	AR
PointNet (Qi et al., 2017a)	88.45	12.47
PointNet++ (Qi et al., 2017b)	89.82	21.35
Point2Sequence (Liu et al., 2019b)	92.60	10.53
Kd-Network (Klokov and Lempitsky, 2017)	86.20	8.49
Spherical CNN (Cohen et al., 2018)	81.73	55.62
DeepSets (Zaheer et al., 2017)	88.73	9.72
LDGCNN (Zhang et al., 2019a)	92.91	17.82
SO-Net (Li et al., 2018a)	**94.44**	9.64
PRIN (You et al., 2018)	80.13	70.35
Compass + PointNet	80.51	**72.20**

3.3.2 Results

Our results are reported in Table 3.5. Results for all the baselines come from (You et al., 2018). PointNet fails when trained without augmenting the training data with random rotations and tested with shapes under arbitrary rotations. Similarly, in these conditions, most of the state-of-the-art methods cannot generalize to unseen rotations. If, however, we first neutralize the object's pose by Compass and then run PointNet, it gains almost 60 points and achieves 72.20 of accuracy, outperforming the state-of-the-art on an arbitrarily rotated test set. This shows the feasibility and effectiveness of pursuing rotation-invariant processing by canonical pose estimation.

In Figure 3.7, we present some models from ModelNet40, randomly rotated, and then oriented by Compass. The models estimate a very consistent canonical pose for each object class, despite the classes' large shape variations.

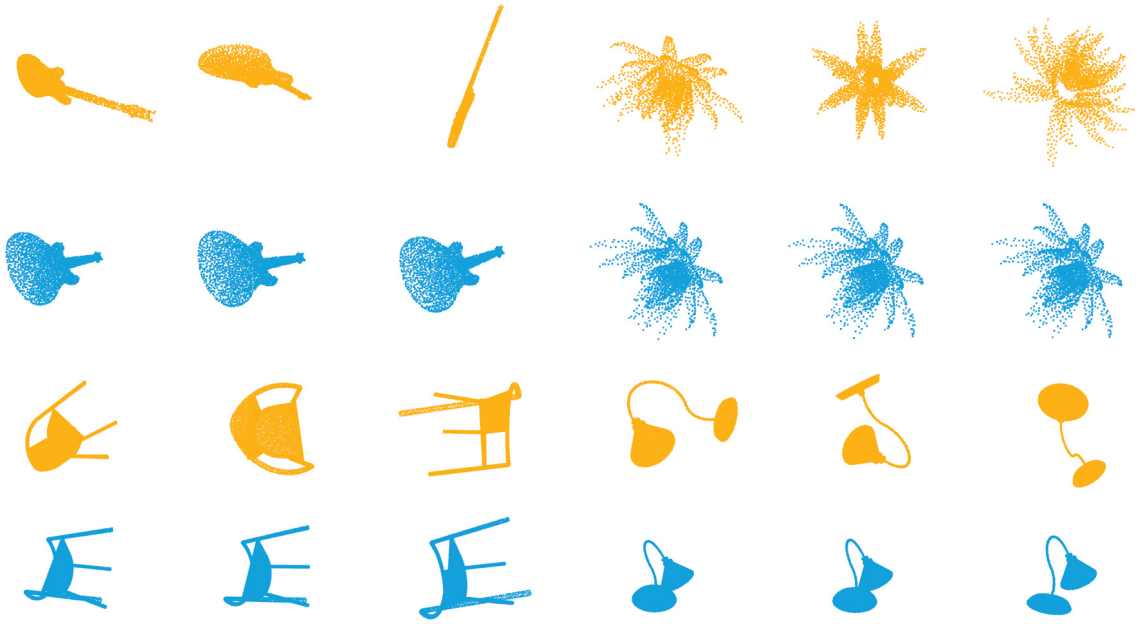
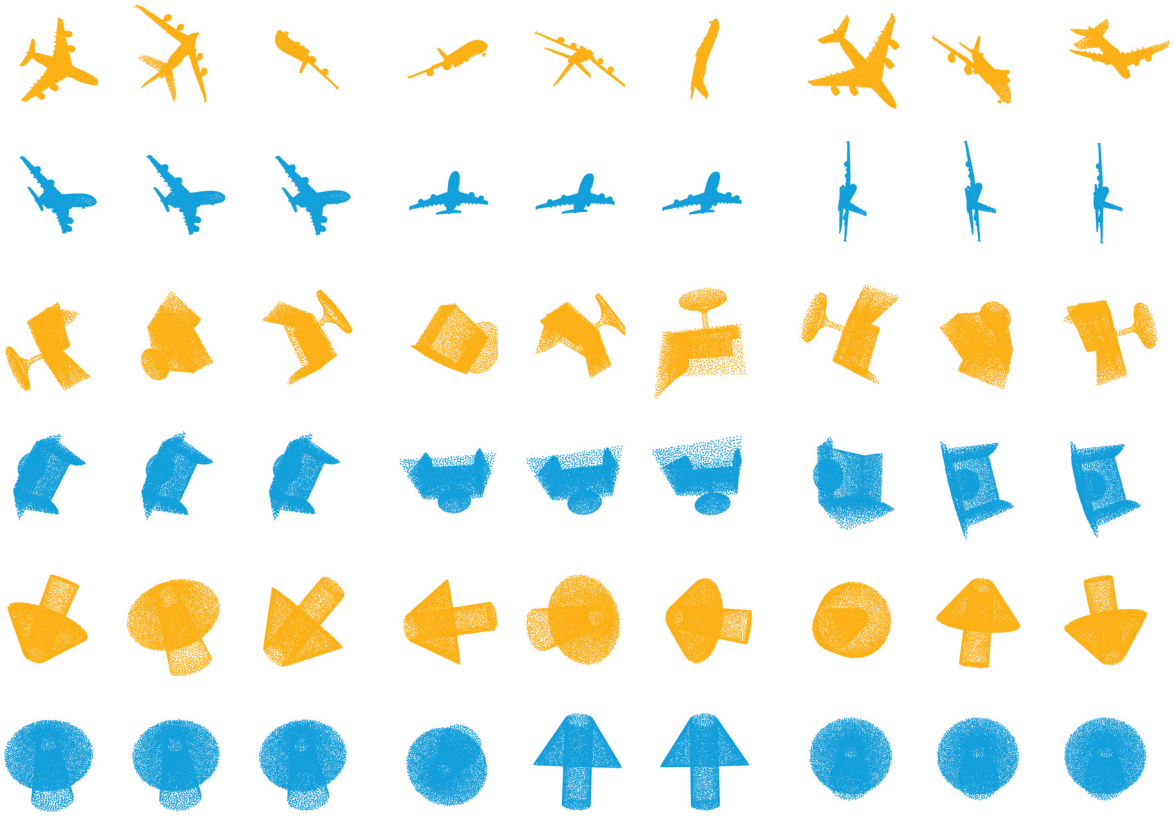


Figure 3.7: Qualitative results on ModelNet40 and ShapeNet in transfer learning. Top row: randomly rotated input cloud. Bottom row: cloud oriented by Compass.

Finally, to assess Compass's generalization abilities for full shapes, we performed qualitative transfer learning tests on the ShapeNet dataset, reported in Figure 3.7. Even if there are different classes, e. g. the lamp, the model trained on ModelNet40 can generalize to an unseen dataset and recovers similar canonical poses for the same object.

We stress the generalization capability of our model by adopting three different configurations to generate the training data. For this experiment, we consider only three categories: airplane, chair, and lamp. The results of this study are shown in Figure 3.8. In the first column, (a), we present results for a *category-specific* training, i. e., learning to orient only one category. Thus, we train one network for each category, and then we test on the test split of the same category. In (b), we present results for a category-agnostic network, i. e., a single model trained on samples from the three categories. Finally, in (c), we show the model's orientation results trained according to the protocol defined in the main paper, i. e., transferring to ShapeNet a model trained on the ModelNet40 dataset. From these results, we observe how the canonical pose can often be correctly recovered under random rotations. For each triplet of rotated objects (colored in yellow), the estimated canonical pose (in blue) is consistent, even in a transfer learning strategy. Interestingly, looking at the fourth and sixth row of the (b) and (c) cases, where the model has to define a canonical orientation for more than one category at once, the canonical pose learned by the network seems to be similar across the chair and lamp categories, which have as the first principal direction the direction of gravity. This suggests that our network may generalize the concept of canonical pose across objects of different categories that share a similar geometric structure.



(a) Category-specific training (b) Category-agnostic training (c) Transfer learning

Figure 3.8: Qualitative results on ShapeNet dataset under different training strategies. Clouds in yellow represent randomly rotated input clouds and the blue ones represent those oriented by Compass. In (a), we present orientation results after training Compass with examples belonging only to a specific category from ShapeNet; in (b), the orientation results after training Compass with a training set comprising *airplanes*, *chairs* and *lamps* together; and, in (c) the orientation results from the model trained on the ModelNet40 dataset and tested on the ShapeNet dataset.

3.4 FINAL REMARKS AND OVERVIEW

In this chapter, we have presented Compass, a novel method to canonically orient 3D shapes. Compass is, at the best of our knowledge, the first fully learned LRF. By using the Spherical CNNs and its inherent equivariant property, we let the network define the best-suited pose for the surfaces' underlying geometry. Our approach robustly handles occlusions thanks to effective data augmentation.

Experimental results demonstrate our approach's benefits in defining a canonical pose for local surface patches and rotation-invariant shape classification. Compass outperforms all existing hand-crafted methods in three benchmarking datasets. Compass outperforms traditional and consistent methods such as FLARE (Petrelli and Di Stefano, 2012), and also the most recent GFrames (Melzi et al., 2019), published as oral in CVPR'19. Our learned LRF can also serve as a transformation network attached to PointNet, achieving state-of-the-art on object classification in a non-rotated training setup and evaluation with rotation augmentation, with accuracy of 72.20%.

4 UNSUPERVISED LEARNING OF LOCAL DESCRIPTORS FOR POINT CLOUDS

Invariance to viewpoint changes is paramount to 3D descriptors. Unfortunately, learned approaches exhibit a performance drop when trained and tested on different 3D rotations (Zeng et al., 2017a; Deng et al., 2018b; Esteves et al., 2018). This drop is probably because 3D data under rotations induce different network features, as demonstrated in 3D Object Classification in (Sedaghat et al., 2016).

As a consequence, a popular strategy to endow a learned 3D descriptor with rotation invariance consists in expressing the 3D coordinates of the points belonging to the input patch w.r.t. a coordinate system centered at the feature point and defined according to an LRF (Khoury et al., 2017; Gojcic et al., 2019) or RA (Deng et al., 2018a). For instance, CGF (Khoury et al., 2017) and 3DSmoothNet (Gojcic et al., 2019) rely on the LRFs proposed by (Salti et al., 2014) and (Yang et al., 2017), respectively. However, again, hand-crafted choices may inject noisy orientation signals into the training process due to the non-ideal repeatability of the actual algorithm deployed to compute the LRF. As a matter of fact, and similarly to descriptors, the literature on LRFs vouches for the lack of a golden standard, with different algorithms behaving differently across datasets.

With the advent of new point convolution operators (Thomas et al., 2019; Choy et al., 2019a), some of the most recent approaches in the field (Choy et al., 2019b; Bai et al., 2020) employ fully-convolutional architectures (Long et al., 2015) to densely learn descriptors across the input cloud in one forward pass instead of considering as a single sample, the local neighbors of a 3D keypoint. By augmenting the training data by a random set of rotations, we achieve invariance to rotations, but they fail to generalize when trained and tested on different datasets (Yang et al., 2018a).

This chapter proposes a novel unsupervised framework to learn a *rotation-equivariant* local surface descriptor directly from the raw input data. To do so, we combine Spherical CNNs (Cohen et al., 2018; Esteves et al., 2018), a recently introduced deep learning machinery which extends the correlation operator to signals living in S^2 and $\mathrm{SO}(3)$, together with *folding* operators (Groueix et al., 2018; Yang et al., 2018b). In our training architecture, a spherical encoder learns to compress the input 3D patch's geometric traits into a low-dimensional latent space. Simultaneously, a plane folding decoder (Groueix et al., 2018) warps a 2D grid to reconstruct the input patch. As usual, in unsupervised learning through encoder-decoder architectures, the decoder is unused at inference time, and the low-dimensional representation computed by the encoder provides the patch descriptor. Due to its unsupervised nature, our learning framework does not require ground-truth annotations or complex *non-matching* pairs sampling strategies to supervise the network (Hermans et al., 2017; Wu et al., 2017). The latter is key to many feature learning algorithms (Gojcic et al., 2019; Choy et al., 2019b; Bai et al., 2020), but not trivial to design.

While the encoder-decoder architecture proposed in Deng et al. (2018a) compresses and reconstructs pose-invariant Point Pair Features (Drost et al., 2010), ours does so for raw 3D coordinates under arbitrary poses. As shown in Section 4.1, pose information is needed to correctly reconstruct a patch of 3D points by a plane folding decoder under an arbitrary pose. To encode pose into the latent space, we rely on the peculiar *rotation-equivariance* property of Spherical CNNs. The spherical encoder consists of layers that compute feature maps defined on $\mathrm{SO}(3)$ that are equivariant to a 3D rotation of the raw input data. Thus, we do not need to rely on any hand-crafted, and possibly fragile choice to either express the input 3D patch in a canonical

pose or remap it into a pose-invariant representation. Rather, we learn a rotation-equivariant bottleneck from the raw training data. To pursue pose-invariance on the proposed descriptor, we employ two different approaches:

- To learn a spherical bottleneck, used as a descriptor, we canonicalize it by applying a 3D rotation provided by an off-the-shelf LRF algorithm. We introduce **LEAD: Learned EquivariAnt Descriptor** in Section 4.1, the first rotation-equivariant learned descriptor for 3D keypoint. We also propose an approach based on the PointNet (Qi et al., 2017a) architecture, named **LEAD-PN**.
- To learn an end-to-end self-orienting descriptor that extracts discriminant embeddings and, using the findings previously faced in Chapter 3, also the orientation of patches. This strategy, named as **SOUND: Self-Orienting UNsupervised Descriptor**, is proposed in Section 4.2, and merges LEAD and Compass, two state-of-the-art approaches in a single method.

4.1 LEAD: A ROTATION-EQUIVARIANT DESCRIPTOR

The LEAD training workflow, depicted in Figure 4.1, includes the following steps: (i) a radius search for the neighboring points surrounding a feature point $\mathbf{p}$ is performed; (ii) the resulting support is then converted into a spherical signal discretized along the azimuth, elevation and radial dimensions; (iii) the descriptor is inferred by feeding the spherical encoder with the input spherical signal; (iv) an ensemble of random points forming a 2D grid is concatenated with the learned latent space; (v) through a plane folding decoder (Groueix et al., 2018), the 2D grid is warped according to the learned descriptor in order to reconstruct the input 3D patch; (vi) Finally, the calculus of the Chamfer distance between the original input patch and its reconstructed version is adopted to train the network. Spherical CNNs require spherical signals as input. To do so, we adopt the same preprocessing stage previously described in subsection 3.1.1.

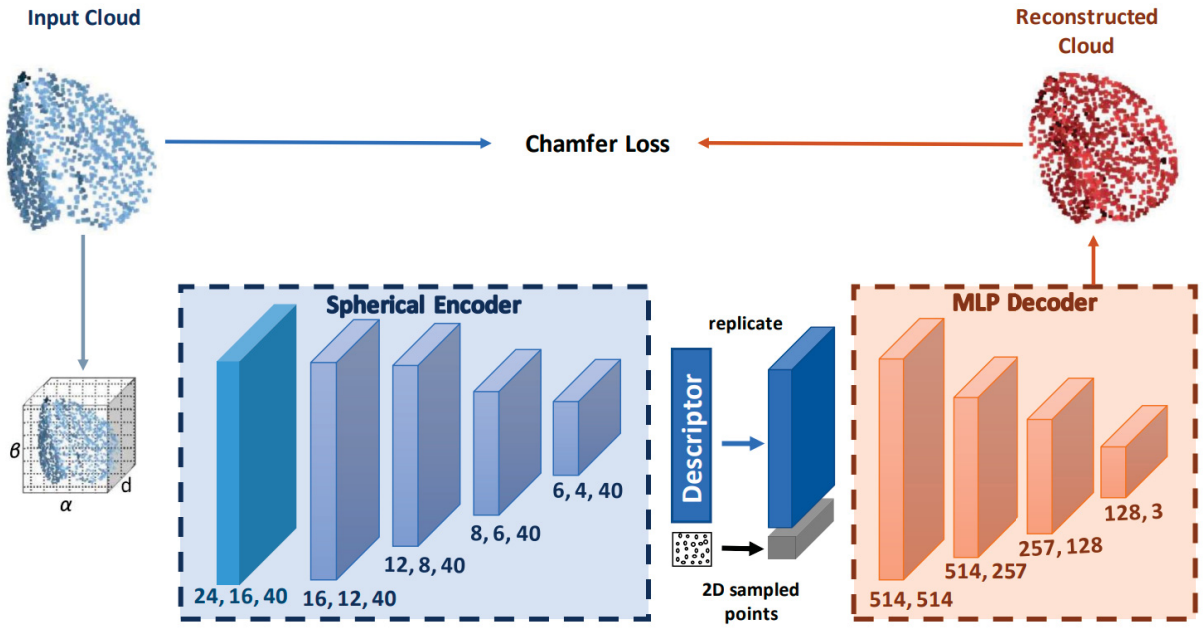


Figure 4.1: Architecture of the **LEAD** network

The points within the local support of a given feature point p are converted into a spherical signal representation and then sent through the spherical encoder to get an equivariant

descriptor. The numbers below the spherical signal indicate the value of cells along the α, β and d dimensions. Operations in the encoder are implemented through the Generalized Fourier Transform with signals discretized according to a bandwidth parameter (Cohen et al., 2018). Thus, the triplets below the encoder layers indicate the input bandwidth, output bandwidth, and channels. Starting from the information stored in the descriptor, the decoder reconstructs the original point cloud deforming the 2D grid. The numbers under each MLP layer denote the value of input and output channels, respectively.

Thanks to the equivariance property, we do not need to feed the network with invariant representations at training time, such as in Deng et al. (2018a), Khoury et al. (2017), and Gojcic et al. (2019), and delay this choice at test time, providing two essential benefits to the resulting embedding. First, we can train the network from less preprocessed input data than existing proposals, since we do not have to choose a specific LRF, secondly with an *LRF-agnostic train* approach, we can avoid some intrinsic errors on the chosen LRF, which can inject noise in the training process. Not being tied to a specific LRF enables us to choose the best way to define a canonical representation at a test time without retraining the network from scratch. The domain shift problem is particularly critical for 3D data because of the heterogeneous acquisition techniques and the different sensing modalities. Consequently, the same LRF estimator can behave very differently across datasets resulting in not stable performance. Being invariant to rotation only at test time makes our approach flexible and prone to generalization. Offering this property is almost impossible for all the other standard representations, e. g. the output of an MLP as used in PointNet, which cannot be rotated after having its computation. Only a descriptor that lives in $\mathrm{SO}(3)$ can be rotated at test time, i. e., only the output of a Spherical CNN to date.

We can alternatively employ a standard PointNet as an encoder to learn a local 3D feature descriptor from raw point cloud data and explore two possible ways to attain invariance to rotation: trough data-augmentation or rely on an LRF to provide canonically oriented patches to the network. To highlight the benefits of learning an equivariant descriptor, which can be turned into an invariant one only a test time, in the following, we explore both of the ways as mentioned earlier, validating the advantages of our design choice trough dedicate experiments. We replace the Spherical encoder in the architecture illustrated in Figure 4.1 with a standard PointNet. As the first demonstration, we force the network to learn a rotation-invariant embedding without applying a canonical orientation to the input data, hoping the network learns it through data augmentation by observing randomly rotated versions of the same neighborhood during training without direct supervision. To verify if the learned descriptor achieved invariance to rotation, we measure the distance between descriptors belonging to the same keypoint but under different poses. In Figure 4.2, we show a comparison between a PointNet encoder trained on the 3DMatch Benchmark presented in Section 4.3.1, and a spherical encoder with randomly initialized weights. Since that equivariance is a theoretical property of a Spherical CNN, it is not necessary to train it for this study. Given a neighborhood, we rotate it around a random axis by a growing angle, whose value is reported along the chart's horizontal axis. For every rotation, we forward the rotated neighborhood through a Spherical CNN encoder and a PointNet encoder. Then, we rotate the output of the Spherical CNN by the inverse of the applied rotation (simulating the availability of a perfect LRF) and plot the distance between the descriptor obtained from the rotated and the un-rotated neighborhood. PointNet cannot learn an invariant descriptor in our setup, while the equivariant representation provided by a Spherical CNN can achieve almost perfect invariance when properly rotated. Alternatively, the invariance to rotation can be forced by computing and applying a canonical orientation to the 3D patches before sending them to the encoder at training and testing time. This is the standard procedure adopted both by hand-crafted (Salti et al., 2014; Guo et al., 2013a; Tombari et al., 2010) and by the learned approaches (Gojcic et al.,

2019; Khoury et al., 2017). In this experiment, the PointNet encoder is trained in neighborhoods oriented according to an external LRF (Petrelli and Di Stefano, 2011). The results presented in Tables 4.1 and 4.2, show that canonically orient an equivariant descriptor at test time, instead of learning an invariant one yields better performance in a surface registration pipeline, and, again, confirms the strength of our proposal.

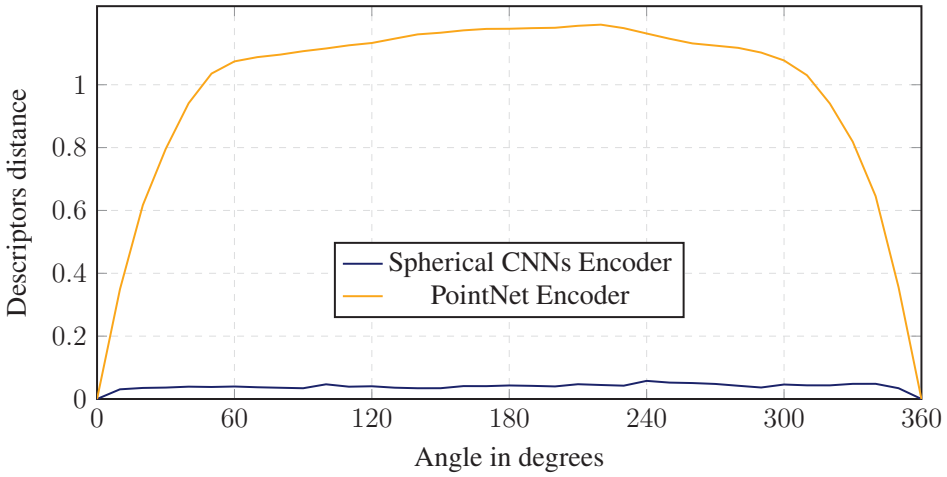


Figure 4.2: Comparison between PointNet and Spherical CNN used as encoders in our framework.

Spherical CNNs have been successfully exploited in Cohen et al. (2018), Esteves et al. (2018), and You et al. (2018) to learn an invariant global shape embedding for 3D Object Classification under random rotations. To this end, a *max-pool* layer is inserted between a chain of S^2-$\mathrm{SO}(3)$ correlation layers and the last *fully-connected* layer performing the classification, to select the most distinctive features regardless the pose under which the object may appear.

4.1.1 Test-Time Invariant Feature Descriptor

At test time, before matching the computed feature descriptors, we need to rotate them w.r.t. a canonical orientation to deliver rotation invariant feature descriptors that can be matched across poses. Differently from the state-of-the-art methods that transform the 3D input patches, we, instead, apply the rotation to the descriptors, i. e., $\mathrm{SO}(3)$ feature maps obtained from the unrotated inputs. Signals in the $\mathrm{SO}(3)$ domain can be rotated by remodulating the spherical harmonics functions resulting from their Fourier transform. A thorough treatment of the topic and the mathematical details of this procedure can be found in Risbo (1996). To canonicalize our learned descriptors, we follow the same setup of Spezialetti et al. (2019), in this case, by adopting the LRF proposed in Petrelli and Di Stefano (2012), named FLARE.

4.1.2 Architecture

The methodology newly outlined is implemented through an architecture made up of two basic parts, a spherical encoder and a multi-layer perceptron decoder. While the last layer of the encoder outputs a codeword, which we employ as an equivariant descriptor, the decoder is responsible for the reconstructions. As a consequence, the decoder is used only at training time. These two components are presented in a more detailed way in the next sections. The training loss is calculated using the Chamfer Loss (Equation 2.7).

4.1.2.1 Encoder

The procedure described in subsection 3.1.1 converts a set of 3D points into a signal defined on the S^2 sphere. Thus, we need a S^2 correlation layer as the first layer of our spherical encoder. Unlike the standard definition of spherical convolution (Driscoll and Healy, 1994), which gives as output a function on the sphere S^2, we adopt an implementation of Cohen et al. (2018), which yields a signal on $\mathrm{SO}(3)$. The use of a conventional convolution definition would limit the network's expressive capacity due to the symmetry along the axis $\mathbf{Z}$ of the learned filters. To further process the resulting $\mathrm{SO}(3)$ feature map, we stack an ensemble of $\mathrm{SO}(3)$ correlation layers, where the last one outputs the equivariant feature descriptor.

4.1.2.2 Decoder

The encoder learns to compress the most meaningful information about the local neighborhood of the input keypoint $\mathbf{p}$ in the descriptor, to produce a descriptive representation. The best way to verify it is to employ an MLP decoder to reconstruct the entire set of points, making up the support of $\mathbf{p}$ starting from the descriptor. A key tool to accomplish this goal in an unsupervised fashion way, is the plane *folding* operator proposed by Groueix et al. (2018) and Yang et al. (2018b). Following Groueix et al. (2018), our decoder tries to deform points in $\mathbb{R}^2$ to surface points in $\mathbb{R}^3$ according to the learned descriptor. Given a feature representation $\mathbf{d}$ for a 3D surface, let $\mathcal{A}$ be a set of points sampled in the unit square $[0, 1]^2$, we concatenate the descriptor $\mathbf{d}$ with the sampled point coordinates $(a_x, a_y) \in \mathcal{A}$ and then forward them through the decoder composed by MLP layers, as shown in Figure 4.1.

4.1.2.3 Hyper and training parameters

The spherical encoder has a first S^2 convolution layer and more four $\mathrm{SO}(3)$ convolution layers with a constant number of channels, 40. The input bandwidths of these five layers are 24, 16, 12, 8, and 6. The selection of the network's design is according to the ablation study presented in subsubsection 4.3.3.1. After each layer we apply a *BatchNorm* step and *ReLU* non-linearities. The last layer of the encoder outputs the descriptor, with 512 entries. On the other hand, the MLP decoder has four fully-connected layers, with BatchNorm and ReLU after the first three layers and tanh on the last output layer. We trained the network with mini-batches of size 100 by using Adam (Kingma and Ba, 2014) and a fixed learning rate of 0.001.

4.2 SOUND: SELF-ORIENTING UNSUPERVISED DESCRIPTOR

The SOUND training workflow is presented in Figure 4.3. As previously pointed, this pipeline combines the LEAD and Compass architectures in a single network. The training process follows: (i) perform a radius search for the neighboring points surrounding a feature point $\mathbf{p}$; (ii) convert the resulting support into a spherical signal (according to subsection 3.1.1); (iii) this signal passes through the S^2 layer, shared between the encoder and the LRF Estimator networks; (iv) on the descriptor network, the spherical encoder outputs the codework bottleneck (v) deforms a 2D grid of random points by a plane folding decoder (Groueix et al., 2018); (vi) at the end of de descriptor branch, we calculate the Chamfer distance between the original input patch and its reconstructed version is adopted to train the network; (vii) given the input patch we apply a random rotation to it, and pass through the LRF estimator in a siamese fashion; (viii) we apply a soft argmax 3D to extract a single triplet (α, β, γ) from last layer the feature map; (ix) at the end of the LRF

estimator network, we calculate the geodesic distance between rotations; (x) we calculate the network loss by combining Chamfer and geodesic distances.

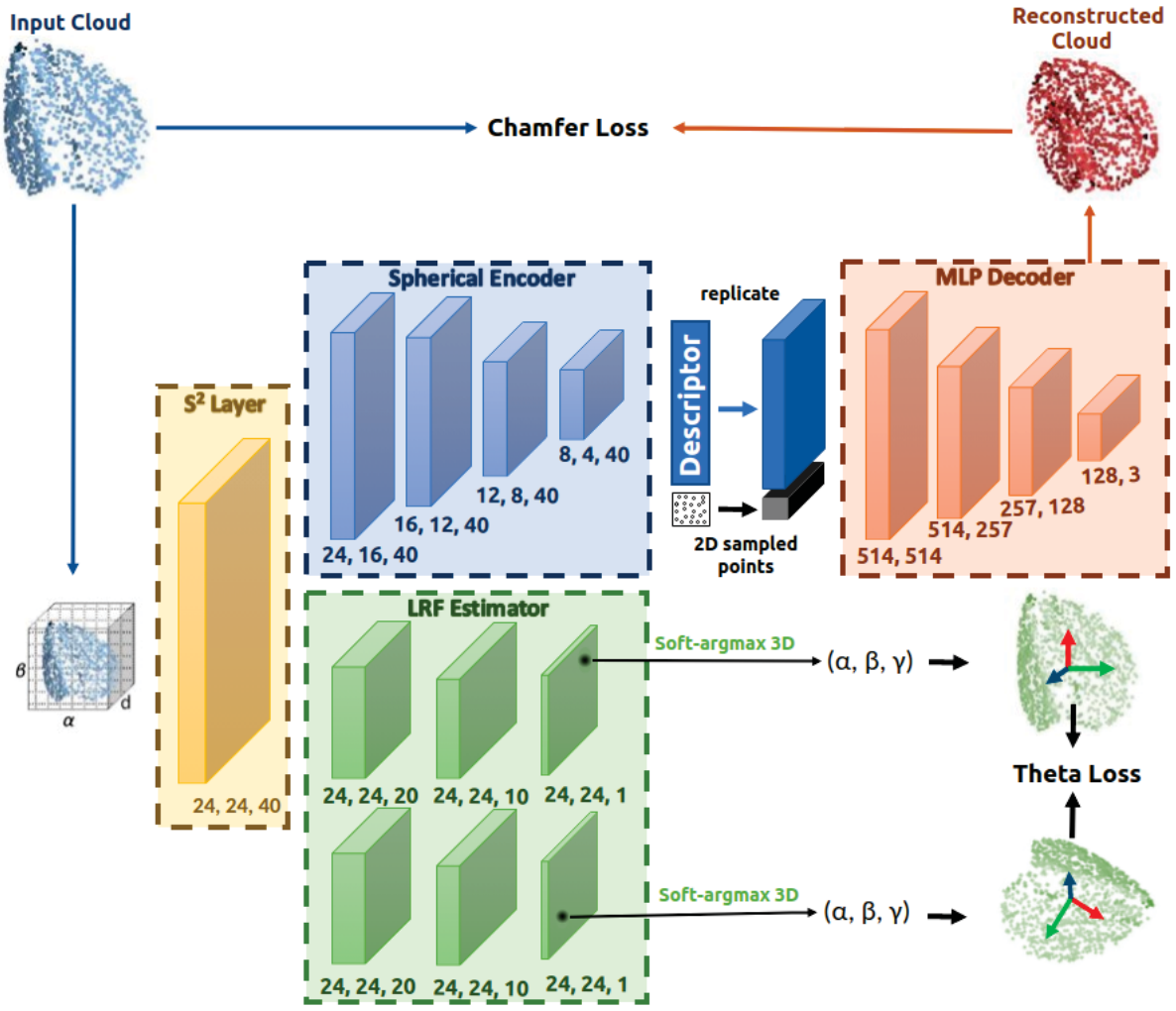


Figure 4.3: Architecture of the **SOUND** network

The points within the local support of a given feature point p are converted into a spherical signal representation and then sent through the spherical encoder to get an equivariant descriptor. The numbers below the spherical signal indicate the value of cells along the α, β, and d dimensions. Operations in the encoder are implemented through the Generalized Fourier Transform with signals discretized according to a bandwidth parameter (Cohen et al., 2018). Thus, the triplets below the encoder layers indicate the input bandwidth, output bandwidth, and channels. Starting from the information stored in the descriptor, the decoder reconstructs the original point cloud deforming the 2D grid. The numbers under each MLP layer denote the value of input and output channels, respectively. On the LRF estimator network, we apply a random rotation on the input 3D patch, feeding the network with the original and augmented version to extract the aligning rotation between them. At test time, only one branch is involved.

4.2.1 Architecture

To understand the SOUND architecture, we split it into four subnets, as depicted in Figure 4.3: The S^2 **layer**, that produces shared weights between the branches of the network; the **encoder**, responsible by learning discriminative features of the descriptor; the **decoder**, used only at

training time to reconstruct the patches and aid the encoder on the learning process; and finally, **LRF estimator**, that extracts the orientation of patches. The Encoder and Decoder follow the same principles addressed on the LEAD descriptors, and the LRF estimator follows the Compass network.

4.2.1.1 Hyperparameters

The layer's structure remains unchanged concerning the individualized architectures, i.e, channels, spherical signal, descriptor size, and LRF Estimator and Decoder. However, we had to change the *encoder*, synchronizing the inputs of both networks and the output of S^2 layer. Despite outputting a bandwidth of 16 filters on the S^2 layer, we changed it to 24, consequently, the following input/output configuration. Hence, the input bandwidths considering only the descriptor's part are 24, 24, 16, 12, and 8.

In consonance with the ablation study presented in subsubsection 4.3.3.1, we employ a near-identity grid convolution. The last layer of the encoder outputs the descriptor, with 512 bins. We trained the network with mini-batches of size 8 by using Adam (Kingma and Ba, 2014) and a fixed learning rate of 0.001.

4.2.2 Loss

The overall loss of SOUND network is given by (4.1), and considers the Chamfer distance, presented by the Equation (2.7), and the angular distance, from the Equation (3.3):

$$\mathcal{L}_{full} = 100 \times \mathcal{L}_{chamfer} + \mathcal{L}_{angular} \tag{4.1}$$

We choose this weighting by 100 empirically, based on some previous observations and studies we made that showed that the LRF network tends to converge and overfit more rapidly compared to the descriptor network. Thus, to hinder this development, we first force the descriptor convergence by overweighting the chamfer loss. Only when the chamfer loss is relatively little, after some epochs, the LRF estimator converges more efficiently.

4.3 EXPERIMENTAL RESULTS

4.3.1 Experimental setup

In this section, we test LEAD and SOUND in a pairwise surface registration scenario considering indoor and outdoor datasets. For indoor we use the 3DMatch benchmark (Zeng et al., 2017a), together with the *Rotated*, and for outdoor environments, we adopt the ETH dataset (Pomerleau et al., 2012). We adopt the same setup proposed in Deng et al. (2018a): each fragment is downsampled by a voxel grid filter with a leaf of 2 cm and the the surface normals are estimated using a 17-point neighborhood (Hoppe et al., 1992). Regarding the radius for the descriptors, to take into account the different scales of the represented geometries, similar to Gojcic et al. (2019), and Deng et al. (2018a), we consider 0.3 m for 3DMatch and 1.0 m on the ETH.

4.3.2 Evaluation protocol

We evaluate both descriptors in a pairwise registration pipeline following the standard protocol (Deng et al., 2018a; Gojcic et al., 2019; Li et al., 2020a; Bai et al., 2020). For each scene, we consider all the pairs of fragments with at least 30% of overlap between them, and we describe each fragment 5000 uniformly sampled keypoints made available by the authors of the

benchmark (Zeng et al., 2017a). Each couple of fragments' correspondences are established by performing the reciprocal nearest neighbor in the descriptor space (Zeng et al., 2017a). Once the correspondences are created, we use standard metrics to measure their correctness. We employ two kinds of measurements: direct measures aimed at verifying the percentage of correctly matched keypoints such as *Feature-match recall* and the average number of correct matched keypoints, and *indirect* measures designed to inspect the rigid motion matrix derived by matching local descriptors, such as the Relative Rotation Error (RRE) and Relative Translation Error (RTE).

As for the comparison against the state-of-the-art methods, we adopt the commonly hand-crafted descriptors FPFH (Rusu et al., 2009), Spin Images (Johnson and Hebert, 1999), SHOT (Salti et al., 2014) and USC (Tombari et al., 2010). Moreover, we compare against the current state-of-the-art in learned 3D feature descriptors considering: 3DMatch (Zeng et al., 2017a), CGF (Khoury et al., 2017), PPFNet (Deng et al., 2018b), 3DSmoothNet (Gojcic et al., 2019), FCGF (Choy et al., 2019b), D3Feat (Bai et al., 2020) and Li *et al.* (Li et al., 2020a) as supervised methods, while PPFFoldNet (Deng et al., 2018a), 3DPointCaps (Zhao et al., 2019) and the previous version of LEAD (Spezialetti et al., 2019), as unsupervised ones. We take the implementations of the hand-crafted methods from the PCL library (Rusu and Cousins, 2011), while for learned descriptors, we grab the results from the original papers, except for the FCGF results on the ETH dataset (Table 4.6), not provided by the authors, but generated using their original code.

4.3.2.1 Feature-match recall

The feature-match recall (Deng et al., 2018a) assesses the percentage of fragment pairs that can recover the pose with high confidence. According to this metric, a pair of fragments is correctly registered when the number of matched keypoints is greater than the τ_2 threshold, set to 5% of the extracted keypoints. We consider a correct match between two keypoints if the distance l_2 between them, after being aligned with the ground truth transformation, is below a threshold $\tau_1 = 10$ cm.

4.3.2.2 RRE and RTE

The relative rotation and translation errors measure the quality of the estimated rigid motion after applying RANSAC on the set of correspondences established by matching local 3D feature descriptors, compared to the ground-truth one. Given, the translation ($\hat{T}$) and rotation ($\hat{R}$) output of RANSAC, and the respective ground-truth T^* and R^*, we calculate the RRE and RTE by Eqs. (4.2) and (4.3):

$$RRE = \arccos\left(\frac{(tr(\hat{R}^T R*) - 1)}{2}\right) \tag{4.2}$$

$$RTE = \left|\hat{T} - T^*\right| \tag{4.3}$$

4.3.3 Results on the 3DMatch Benchmark dataset

We report the experimental evaluation results on the 3D Match benchmark in terms of feature-match recall in Table 4.1. The first outcome of our experiments is that LEAD improves over the previous proposal (Spezialetti et al., 2019). Moreover, the average recall of 95.84% outperforms all state-of-the-art 3D local descriptors on the standard registration benchmark, except for Li et al.

(2020a), which achieves the best performance. However, it is worth noting that the work proposed by Li et al. (2020a) employs a neural render to parameterize the local neighborhood of a 3D keypoint into a collection of depth images to perform multi-view reasoning. Regarding SOUND, the results are also inspiring and demonstrate the feasibility of such integrated approaches, being outperformed only by very recent, and supervised approaches like Li et al. (2020a), D3Feat (Bai et al., 2020), and LEAD.

The second outcome which magnifies even more our work, is that with unsupervised methods we outperform most of the supervised approaches, i. e., FCGF (Choy et al., 2019b) and 3DSmoothNet (Gojcic et al., 2019). We also perform consistently well against the unsupervised proposals such as PointCaps3D (Zhao et al., 2019), PPF-FoldNet (Deng et al., 2018a) with a gain of 0.17 over the top performer PointCaps3D, in this regard we run the benchmark using only 2000 keypoints, i. e., LEAD 2K, since the authors in (Zhao et al., 2019) provided results for a limited number of keypoints. The hand-crafted proposals, leaded by SHOT (Salti et al., 2014) and USC (Tombari et al., 2010) are able to compete and, sometimes outperform the learning approaches such as PPFFoldNet (Deng et al., 2018a) and PointCaps3D (Zhao et al., 2019).

Table 4.1: Results on the 3DMatch benchmark in terms of feature-match recall. Test data are from SUN3D (Xiao et al., 2013), except for *Kitchen* data which is from 7-scenes (Shotton et al., 2013). Best result on each column is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
PPFFoldNet (2K)	0.7352	0.7564	0.6250	0.6593	0.6058	0.8889	0.5753	0.5974	0.6804
PointCaps3D (2K)	0.8518	0.8333	0.7740	0.7699	0.7308	0.9444	0.7397	0.6494	0.7867
LEAD (2K)	0.9822	0.9679	0.9087	0.9956	0.9519	0.9815	0.9281	0.8961	0.9515
FPFH	0.7391	0.7885	0.6442	0.8142	0.7115	0.8889	0.7432	0.7013	0.7539
Spin Images	0.6561	0.7564	0.6731	0.6770	0.6346	0.7407	0.4692	0.4545	0.6327
SHOT	0.8893	0.8974	0.8221	0.9336	0.8750	0.8889	0.8630	0.8312	0.8751
USC	0.9308	0.9103	0.7788	0.9204	0.8462	0.8889	0.8664	0.8052	0.8684
3DMatch	0.5810	0.7244	0.6154	0.5442	0.4808	0.6111	0.5171	0.5065	0.5726
CGF	0.4605	0.6154	0.5625	0.4469	0.3846	0.5926	0.4075	0.3506	0.4776
PPFNet	0.8972	0.5577	0.5913	0.5796	0.5769	0.6111	0.5342	0.6364	0.6231
PPFFoldNet	0.7866	0.7628	0.6154	0.6814	0.7115	0.9444	0.6199	0.6234	0.7182
Spezialetti et al. (2019)	0.9802	0.9615	0.8942	0.9823	0.9519	0.9815	0.9144	0.8701	0.9420
3DSmoothNet	0.9700	0.9550	0.8940	0.9650	0.9330	0.9820	0.9450	0.9350	0.9474
FCGF	0.9860	0.9620	0.9330	0.9780	0.9420	0.9820	0.9350	0.8960	0.9518
D3Feat	-	-	-	-	-	-	-	-	0.9580
Li et al. (2020a)	**0.9940**	**0.9870**	**0.9470**	**0.9960**	**1.0000**	**1.0000**	**0.9550**	**0.9220**	**0.9750**
LEAD-PN	0.9348	0.9167	0.8269	0.9115	0.8269	0.9259	0.8219	0.7792	0.8680
LEAD	0,9901	0.9808	0.9135	0.9956	0.9808	0.9815	0.9418	0.8831	0.9584
SOUND	0,9862	0.9679	0.9183	0.9956	0.9615	0.9815	0.9315	0.8831	0.9532

In Figure 4.4, we report results when varying the threshold τ_2 on the percentage of correct matches to establish a pair of fragment correctly aligned (Deng et al., 2018a). LEAD and SOUND outperform the others competitors for almost all thresholds, and this difference is more prominent when the value of the threshold is increased, giving a difference of almost 10% regarding 3DSmoothNet (Gojcic et al., 2019) and FCGF (Choy et al., 2019b) at $\tau_2 = 0.2$.

Finally, in Table 4.2 we report the results for the rotated 3DMatch benchmark (Deng et al., 2018a). As expected, all the rotation-invariant methods get performance similar to the results reported in 4.1, and our equivariant descriptor oriented at test time with FLARE (Petrelli and Di Stefano, 2012) still delivers competitive performance against Li et al. (2020a).

To get a thorough evaluation, we investigated on the quality of the found correspondences, recalling that we consider two keypoints a match when the l_2 distance between a keypoint on the first fragment and its correspondent on the second one after the alignment with the ground-truth

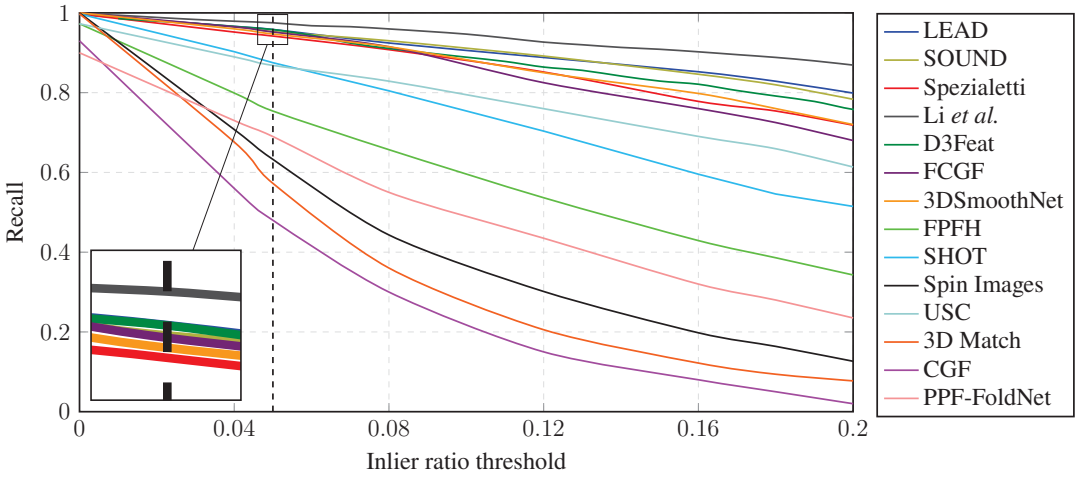


Figure 4.4: Results of different methods under varying inlier ratio threshold τ_2.

Table 4.2: Results on the rotated 3DMatch benchmark in terms of feature-match recall. Test data are from SUN3D (Xiao et al., 2013), except for *Kitchen* data which is from 7-scenes (Shotton et al., 2013). Best result on each column is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
PointCaps3D (2K)	0.8498	0.8525	0.7692	0.8141	0.7596	0.9259	0.7602	0.7272	0.8073
PPFFoldNet (2K)	0.7352	0.7692	0.6202	0.6637	0.6058	0.9259	0.5616	0.6104	0.6865
LEAD (2K)	0.9862	0.9744	0.8942	0.9956	0.9615	0.9815	0.9315	0.8571	0.9478
FPFH	0.7451	0.7949	0.6587	0.8142	0.7212	0.9259	0.7260	0.7530	0.7674
Spin Images	0.6502	0.7628	0.6635	0.6903	0.6635	0.7222	0.4692	0.4935	0.6394
SHOT	0.8794	0.8910	0.8317	0.9425	0.8654	0.9074	0.8493	0.8312	0.8747
USC	0.9170	0.9103	0.7548	0.9292	0.8558	0.9074	0.8836	0.8571	0.8769
3DMatch	0.0040	0.0128	0.0337	0.0044	0.0000	0.0096	0.0000	0.0260	0.0113
CGF	0.4466	0.6667	0.5288	0.4425	0.4423	0.6296	0.4178	0.4156	0.4987
PPFNet	0.0020	0.0000	0.0144	0.0044	0.0000	0.0000	0.0000	0.0000	0.0026
PPFFoldNet	0.7885	0.7821	0.6442	0.6770	0.6923	0.9630	0.6267	0.6753	0.7311
Spezialetti et al. (2019)	0.9763	0.9679	0.8894	0.9779	0.9615	**0.9815**	0.9110	0.8442	0.9387
3DSmoothNet	0.9720	0.9620	0.9090	0.9650	0.9230	**0.9820**	**0.9450**	**0.9350**	0.9491
FCGF	0.9783	**0.9744**	**0.9183**	0.9735	**0.9712**	**0.9815**	**0.9452**	0.8831	0.9532
D3Feat	-	-	-	-	-	-	-	-	0.9550
Li et al. (2020a)	-	-	-	-	-	-	-	-	**0.9690**
LEAD-PN	0.9328	0.9295	0.8462	0.9204	0.8462	0.9259	0.8253	0.7662	0.8741
LEAD	**0.9921**	**0.9744**	0.8990	**0.9956**	**0.9712**	**0.9815**	**0.9452**	0.9221	0.9601

is smaller than $\tau_1 = 10cm$. The results reported in Table 4.3 show dominance on the benchmark's scenes and the average by the multi-view approach of Li et al. (2020a) against the pure 3D based approaches. However, SOUND is the second-best method, followed by LEAD, both with a significant gain over 3DSmoothNet.

Additionally, we adopt a more application-oriented metric to verify the quality of the alignment carried out by our descriptor in a full pairwise registration pipeline. Thus, we also evaluate our proposal in terms of the RRE and RTE presented in subsubsection 4.3.2.2 after applying RANSAC. From the results exhibited in Table 4.4, we can claim that SOUND outperforms the other competitors, and LEAD performs at the same level with Gojcic et al. (2019), with a slightly better performance in RTE on average. It is valuable to recall how RANSAC operates, at least 3 matches are necessary to correctly estimate a rigid motion between two fragments, this justify why the gain in performance of our methods seem modest against 3DSmoothNet (Gojcic et al., 2019), even if the number of correctly matched keypoint is larger as we showed in Table 4.3.

Table 4.3: Average number of correct correspondences on the 3DMatch Benchmark. Best result on each column is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
FPFH	89	142	125	86	94	119	56	74	98
SHOT	154	206	182	131	124	159	84	121	145
SI	120	145	152	102	91	111	51	71	105
USC	150	216	175	147	120	159	97	161	153
3DMatch	103	134	125	73	64	64	64	84	88
CGF	125	156	142	90	94	130	55	78	108
Spezialetti et al. (2019)	265	333	304	296	261	293	223	292	283
3DSmoothNet	274	324	318	272	238	276	171	246	264
Li et al. (2020a)	**380**	**438**	**395**	**457**	**407**	**446**	**299**	**366**	**398**
LEAD-PN	205	255	246	194	182	212	152	185	204
LEAD	273	336	314	307	277	310	226	290	292
SOUND	276	338	316	313	279	310	229	292	294

Table 4.4: Results on the 3DMatchBenchmark in terms of RRE and RTE after RANSAC. Best result on each column is in bold.

	Kitchen		Home 1		Home 2		Hotel 1		Hotel 2		Hotel 3		Study		MIT Lab		Average	
	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE
FPFH	11.58	0.31	13.77	0.48	30.51	0.78	8.02	0.25	18.26	0.44	15.60	0.27	18.40	0.54	14.43	0.51	16.32	0.45
SHOT	5.42	0.14	9.59	0.29	14.40	0.44	4.65	0.17	14.86	0.33	10.82	0.14	13.16	0.39	9.34	0.33	10.28	0.28
SI	9.86	0.27	16.37	0.47	20.04	0.60	10.00	0.32	20.14	0.53	16.64	0.29	23.57	0.72	23.73	0.54	17.55	0.47
USC	9.85	0.25	13.38	0.43	30.77	0.82	10.06	0.33	27.33	0.64	16.55	0.25	13.39	0.42	16.87	0.50	17.27	0.46
3DMatch	9.37	0.29	9.31	0.29	16.64	0.58	13.97	0.53	29.21	0.80	23.47	0.41	16.09	0.51	20.63	0.79	17.34	0.52
Spezialetti et al. (2019)	4.00	0.10	7.04	0.22	13.64	0.34	2.53	0.10	7.09	0.18	8.51	0.11	7.87	0.25	9.50	0.29	7.52	0.20
3DSmoothNet	3.88	0.10	8.62	0.27	10.36	0.29	2.23	**0.07**	8.40	0.29	8.01	0.10	8.68	0.27	8.54	0.34	7.34	0.22
Li *et al.*	**2.33**	**0.06**	**2.99**	**0.10**	**6.11**	**0.23**	**2.20**	**0.07**	**4.31**	**0.12**	**4.98**	**0.08**	**5.97**	**0.21**	**4.20**	**0.17**	**4.14**	**0.13**
LEAD	3.68	0.10	6.72	0.19	12.46	0.34	2.44	0.08	7.83	0.22	6.77	0.11	8.00	0.26	10.81	0.32	7.34	0.20
SOUND	3.56	0.10	7.01	0.20	10.96	0.31	2.85	0.09	7.81	0.21	6.90	0.10	8.06	0.26	10.52	0.30	7.21	0.20

4.3.3.1 Ablation study: Near Identity versus Equatorial Grid

As design choice of Spherical CNNs (Cohen et al., 2018) architecture, two distinct types of spherical grids and hyper-parameters are available to perform both S^2 and $\mathrm{SO}(3)$ correlations: the near identity and equatorial grids. The former defines spatially localized kernels, initialized on the north pole, and rotated over the sphere via the action of $\mathrm{SO}(3)$, the latter one, defines a ring-like kernel around the equator. Choosing the more appropriate grid and hyper-parameters to use in the architecture is key to our framework's performance. Thus, we conduct thorough ablative experiments on 3DMatch dataset aimed at improving the performance of the descriptor regarding the original architecture proposed in Spezialetti et al. (2019), in terms of feature-match recall and description time as well.

We build up different architectures by varying the type of spherical grid, the number of $\mathrm{SO}(3)$ layers, the channels, and the layers' bandwidth. We train a different network for all the configurations presented in Table 4.5 and execute the trials for the learned descriptors on the test split of 3DMatch Benchmark (Zeng et al., 2017a) considering a reduced number of keypoints, 500 instead of 5000. To select the best architecture, we examine the feature-match recall and the time required to forward-pass a constant mini-batch of 25 samples. As a unit of measure for the computation time, we adopt the percentage of *speed-up* relative to the original architecture (A) (Spezialetti et al., 2019).

Finally, the best trade-off configuration is selected according to the Pareto analysis presented in Figure 4.5. It turns out that the version N (Table 4.5) represents the best feature-match recall performance on the Pareto frontier, with a significant reduction on the time processing.

Table 4.5: Ablation study results on the 3DMatch benchmark. With Time we refer a time relative to the base network architecture (A). Networks on the Pareto frontier on the column Network, best values on recall and Normalized time in bold. Tests are performed for a subset of 500 keypoints

Network	Grid		SO(3)	Input	Channels	Recall	Time
	Equatorial	Near Identity	layers	Bandwidths			
A	✓		3	[24, 24, 4]	40	0.924	1.000
B		✓	3	[24, 24, 4]	40	0.922	1.025
C	✓		3	[24, 24, 24]	40	0.922	1.238
D		✓	3	[24, 24, 24]	40	**0.929**	1.253
E		✓	2	[24, 24]	40	0.919	1.025
F		✓	3	[12, 8, 6]	40	0.922	0.636
G		✓	3	[16, 12, 8]	60	0.899	0.679
H	✓		3	[16, 12, 8]	40	0.915	0.632
I		✓	3	[16, 12, 8]	40	0.924	0.634
J		✓	3	[16, 12, 8]	30	0.902	0.611
K		✓	3	[16, 12, 8]	20	0.916	**0.587**
L		✓	2	[16, 8]	40	0.920	0.604
M	✓		4	[16, 12, 8, 6]	40	0.908	0.637
N		✓	4	[16, 12, 8, 6]	40	**0.929**	0.632

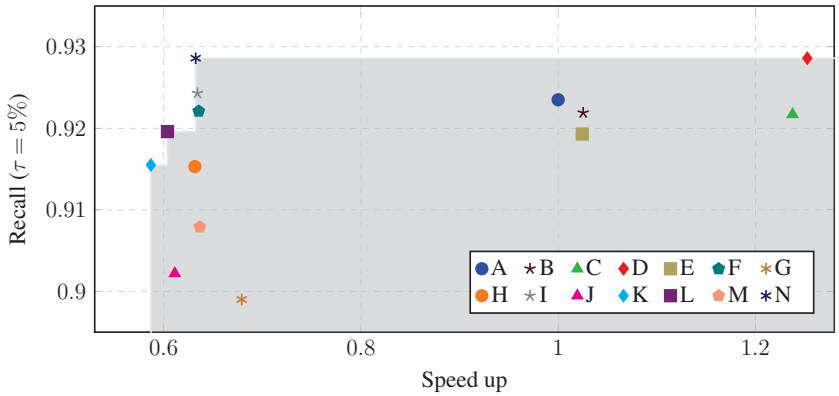


Figure 4.5: Ablation study comparing the different configurations in terms of feature-match recall and speed up. Each configuration is detailed on the Table 4.5. The light gray area shows the Pareto frontier of the test.

4.3.3.2 Ablation study: Rotation Invariant versus Rotation Equivariant Descriptor

To better analyze the importance of our major claim about learning a rotation equivariant embedding than an invariant one, we train a network replacing the Spherical encoder with a three-layer PointNet (Qi et al., 2017a). As we stated in section 4.1, the PointNet encoder cannot learn the invariance to the object's pose, so we feed the network with raw point cloud patches rotated according to the canonical orientation extracted by an LRF, at training and testing time as well. For a fair comparison with the LEAD descriptor, we rely on the same LRF, i. e., FLARE (Petrelli and Di Stefano, 2012), and learn a descriptor of the same size, 512 bins. Both networks are trained with the same procedure on the same train split with a local radius of 0.30 m. Please notice, that for LEAD, the LRF is only used a test time to rotate the descriptor. In Tables 4.1 and 4.2, we present the comparison between LEAD descriptor and the invariant one, LEAD-PN for the 3DMatch and 3DMatch Rotated datasets. These results corroborate with our claim showing a significant improvement margin of almost 10% of the equivariant descriptor, LEAD, on the invariant learned with PointNet. This experiment validates that if we want to depart from the need to feed a specific invariant input representation to the network, we have to learn an equivariant representation, and Spherical CNN is the proper tool to achieve this goal.

4.3.4 Results on the ETH dataset

To evaluate how our proposal performs in an outdoor scenario and how it generalizes to new environments, we tested LEAD and SOUND on the ETH dataset. We get the models trained on the 3DMatch and perform a transfer learning on this dataset. We present the results for feature-match recall in Table 4.6. Our descriptors achieve by far the best performances on this very challenging dataset, recording 97.5% for LEAD, and 97.1% for SOUND on average recall, and outperforming all the state-of-the-art techniques by almost 20%. Surprisingly, FCGF, which remarkably performs on the 3DMatch, does not transfer very well on outdoor conditions resulting in poor scores. The same problem arises both for Li et al. (2020a) and D3Feat, the latter in particular achieves acceptable performance only when coupled with the keypoint detector jointly learned with the descriptor, i. e., D3Feat (*pred*), while when used to describe the set of keypoints provided for the benchmark, i. e., D3Feat (*rand*), exhibits a drop in performance. However, these results show that our unsupervised approach is adaptable and could present remarkable results on a very challenging dataset, such as ETH, without being trained on it. It is worth noting that SOUND's performance is compromised due to the *gazebo summer* scene performance. For two of the scenes, SOUND performs perfectly on this metric.

Table 4.6: Results on the ETH data set in terms of feature-match recall. Best result on each column is in bold.

Method	Gazebo		Wood		Average
	Summer	Winter	Summer	Autumn	
FPFH	0.3860	0.1420	0.1480	0.2080	0.2210
SHOT	0.7450	0.4530	0.6320	0.6170	0.6118
SI	0.6957	0.3979	0.5520	0.5043	0.5375
USC	0.7065	0.2872	0.6160	0.6348	0.5611
CGF	0.3750	0.1380	0.1920	0.1040	0.2023
3DMatch	0.2280	0.0830	0.2240	0.1390	0.1685
Spezialetti et al. (2019)	0.6739	0.4844	0.5920	0.5304	0.5702
3DSmoothNet	0.9130	0.8410	0.7280	0.6780	0.7900
FCGF	0.2554	0.1661	0.2348	0.3040	0.2410
D3Feat (*rand*)	0.4570	0.2390	0.1300	0.2240	0.2620
D3Feat (*pred*)	0.8590	0.6300	0.4960	0.4800	0.6160
Li *et al.*	0.8530	0.7200	0.8400	0.7830	0.7990
LEAD	**0.9239**	**0.9862**	0.9913	**1.0000**	**0.9753**
SOUND	0.8967	**0.9862**	**1.0000**	**1.0000**	0.9707

Table 4.7: Average number of correct correspondences on the ETH dataset. Best result on each row is in bold.

Method	Gazebo		Wood		Average
	Summer	Winter	Summer	Autumn	
FPFH	69	33	32	26	40
SHOT	132	85	100	81	100
SI	106	59	64	53	71
USC	72	22	32	24	37
CGF	73	36	37	32	44
3DMatch	40	16	47	29	33
Spezialetti et al. (2019)	124	94	111	90	105
3DSmoothNet	**182**	139	157	127	151
Li *et al.*	160	117	160	121	139
LEAD	149	142	168	139	149
SOUND	149	**157**	**200**	**172**	**170**

Similar to the 3DMatch, in Table 4.7 we report the results for the matched descriptors and in Table 4.8 for the RRE (degrees) and RTE (meters).

Table 4.8: Results on the ETH dataset (Pomerleau et al., 2012) in terms of RRE and RTE . Best result on each column is in bold.

Method	Gazebo				Wood				Average	
	Summer		Winter		Autumn		Summer			
	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE
FPFH	27.31	0.74	55.69	1.76	34.18	1.10	38.62	1.07	38.95	1.17
SHOT	14.96	0.37	54.90	1.58	4.04	0.48	3.73	0.32	19.41	0.69
SI	8.35	0.20	39.80	1.17	7.23	0.66	14.68	0.55	17.51	0.65
USC	38.62	0.92	76.88	2.21	14.59	0.96	23.98	0.94	38.52	1.26
CGF	29.04	0.69	70.87	2.00	30.62	1.35	41.91	1.26	43.11	1.32
3DMatch	60.03	1.40	78.88	2.28	31.34	1.07	42.76	1.33	53.25	1.52
Spezialetti et al. (2019)	6.98	0.24	41.62	1.20	4.81	0.40	5.62	0.34	14.76	0.55
3DSmoothNet	1.52	0.07	30.82	0.86	1.38	0.38	1.06	0.20	8.70	0.38
Li *et al.*	20.49	0.50	43.12	1.19	0.88	**0.05**	1.06	0.09	16.39	0.46
LEAD	**1.29**	**0.05**	35.00	0.99	0.74	0.10	0.90	**0.08**	9.48	0.31
SOUND	6.56	0.14	**16.77**	**0.46**	**0.64**	0.07	**0.71**	0.11	**6.17**	**0.20**

When we consider the RRE and RTE errors, SOUND presents the best performance over the competitors, with significant relative error improvements. Compared to LEAD, a similar descriptor, pushed by a very efficient LRF such FLARE, these results show how this approach is promising. Two scenes deserve attention: the first is the *gazebo summer*, where SOUND performs relatively worst than LEAD and 3DSmoothNet, corroborating with the results from Table 4.6. The second shows considerable improvement on the RRE on the *gazebo winter* scene, where SOUND presents an error of $16.8°$, while in the other competitors when the error is higher than $30°$. Regarding the RTE metric, the overall results of SOUND also demonstrate this method's robustness by presenting a translational error 10cm lower than the other competitors on average.

Table 4.7 presents the same tendency of the previous table, i.e., on the *Gazebo Summer* SOUND and LEAD present discrepant results regarding the others, also contributing to the average value. Again, SOUND outperforms by far the other tested methods on average.

4.3.5 Computation time

We run our algorithm on a system with a CPU i7 3.2 GHz, a GPU RTX 2080Ti, and 64 GB of RAM. On this hardware, each keypoint description time is about 5.98 ms on average, with an inference time on the GPU of 0.08 ms. This time is comparable to the results got by 3DMatch (Zeng et al., 2017a) and 3DSmoothNet (Gojcic et al., 2019), of 5.0 and 4.6 ms respectively. Comparing to other proposals such as PPF-FoldNet (Deng et al., 2018a) (0.794ms), PointCaps3D (Zhao et al., 2019) (1.208 ms) and FCGF (Choy et al., 2019b) (0.009 ms). From the first one, the low performance of PPFFoldNet compared to our method does not encourage its use despite the time performance. FCGF presents a swift description time but does not perform well in transfer learning for the ETH dataset, and it is also important to point out it demands a higher number of keypoints, to get the results presented on Tables 4.1, 4.2 and 4.6, as explained in Choy et al. (2019b). One way to improve our inference processing time is by leveraging the recent improvements on the Spherical CNNs, called Icosahedral CNNs (Cohen et al., 2019), which, according to the authors, can unlock a substantial boost on the time performance. In this stage, we do the conversion from the point coordinates of the 3D patch to the CPU's spherical signal, and indeed it can reduce this processing time when performing on the GPU.

4.3.6 Qualitative results

To verify the quality of registering a pair of fragments, we present qualitative results after aligning the point clouds with the estimated rigid motion matrix. We present them in Figure 4.6 for the 3DMatch dataset and in Figure 4.7 for the ETH dataset. We compare our descriptor's registration results with the 3DMatch descriptor (Zeng et al., 2017a) as the baseline method for this dataset and with 3DSmoothNet (Gojcic et al., 2019), both supervised approaches. The examples show that LEAD produces correct alignments and sometimes present results even better than the ground truth ones, as we can see on the *Home 2* and *Hotel 2* scenes.

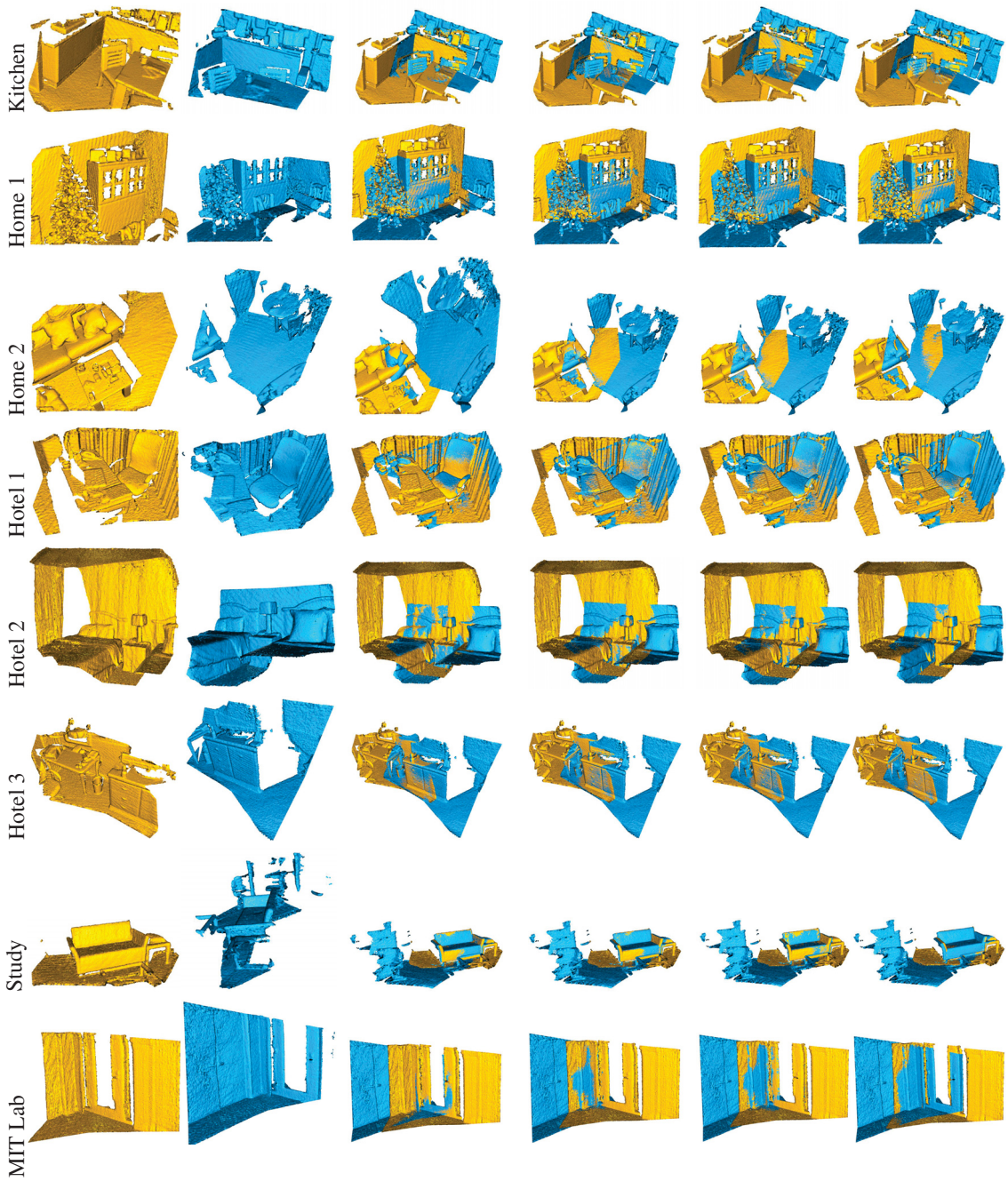


Figure 4.6: Registration results on the 3DMatch Benchmark after RANSAC.

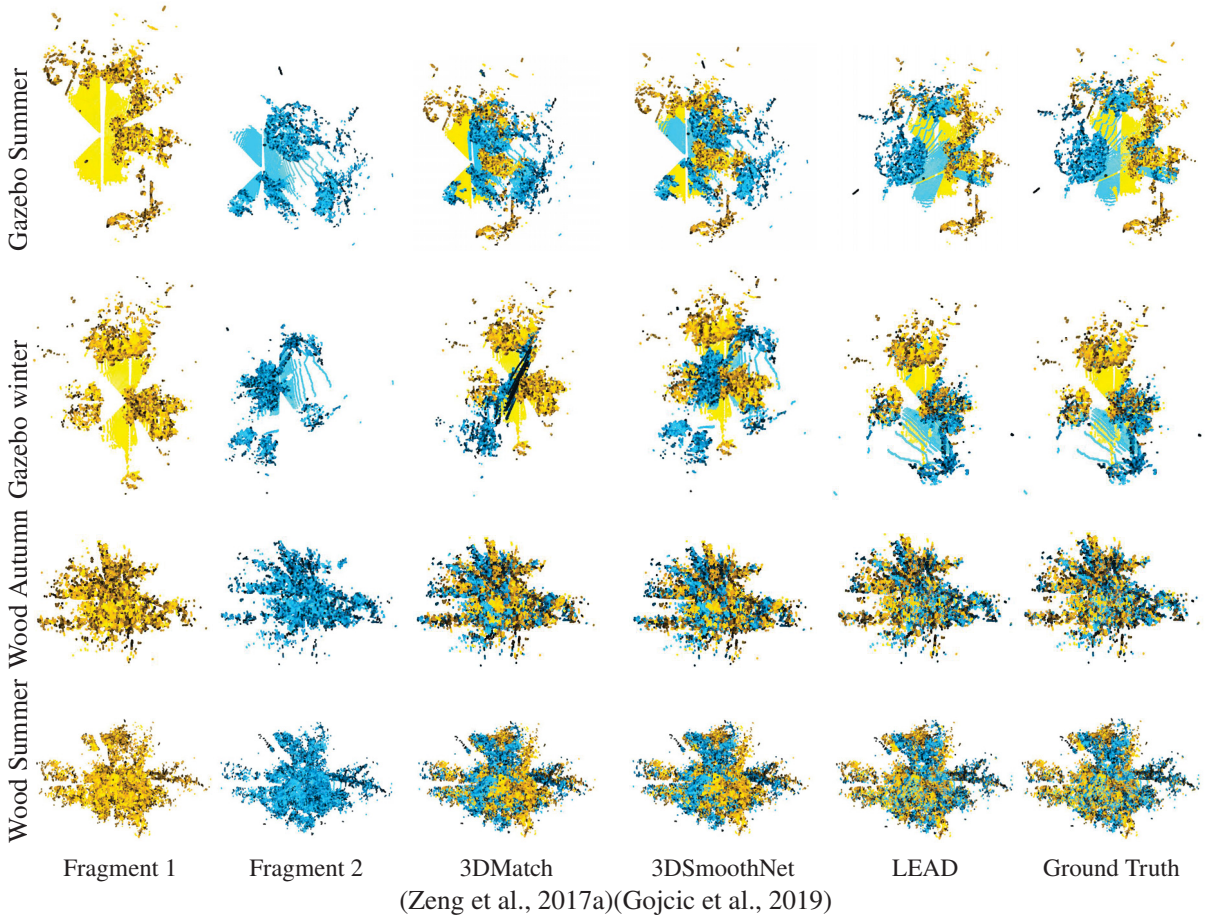


Figure 4.7: Registration results on the ETH Benchmark after RANSAC.

4.4 FINAL REMARKS AND OVERVIEW

In this chapter, we presented LEAD, an unsupervised approach to learning an equivariant descriptor. This descriptor offers outstanding innovation. It is the first orientable descriptor at test time, thanks to the Spherical CNNs framework employed on its development. The results reported show that despite the high accuracy of the standard 3DMatch benchmark, LEAD outperforms the other competitors in transfer learning, featuring state-of-the-art on the ETH dataset. LEAD, extends the proposal of Spezialetti et al. (2019), with significant improvements on the performance of both datasets and the computation and training time. Among the upgrades: An ablation study on the architecture of the network; The use of the ETH dataset on the experiments; The proposal of an invariant descriptor based on the PointNet framework; and a speedup of 37% on the description time, 2% for 3DMatch and 71% for ETH on the feature-match recall.

We also presented, to the best of our knowledge, the first self-orienting descriptor, leveraged by the Spherical CNN framework, which combines two architectures proposed by ours: the LEAD, to learn an embedding discriminant feature vector; and Compass, to extract the orientation of the patches. This proposal presents significant improvements in a transfer learning scenario, on the ETH dataset, on metrics such as RRE, RTE, and the number of matched descriptors. SOUND is the first end-to-end descriptor that achieves invariancy by learning discriminative features from data and its orientation without needing any pre-labeled data.

5 BOOSTING OBJECT RECOGNITION IN POINT CLOUDS BY SALIENCY DETECTION

The application of CV techniques aimed at object recognition is gathering increasing attention in industrial applications. Among others, prominent applications in this space include robot picking in assembly lines and surface inspection. To address such tasks, the vision system must estimate the 6DoF pose of the sought objects, which calls for a 3D object recognition approach. Moreover, in industrial settings, robustness, accuracy, as well as run-time performance are particularly important.

Reliance on RGB-D sensors providing both depth and color information is conducive to 3D object recognition. Nevertheless, typical nuisances to be dealt with in 3D object recognition applications include clutter, occlusions, and the significant degree of noise, which affects most RGB-D cameras. Many studies, such as Johnson and Hebert (1999), Guo et al. (2014b), have investigated these problems and highlighted how local 3D descriptors could effectively withstand clutter, occlusions, and noise in 3D object recognition.

The local descriptors pipeline for 3D object recognition is, however, relatively slow. Indeed, RGB-D cameras generate a high amount of data (over 30MB/s), and, as this may hinder performance in embedded and real-time applications, sampling strategies are needed. In order to reduce processing time, keypoint extraction techniques are widely used. Besides, some solutions propose to assign higher priority to specific image areas, like, for example, in the foveation technique (Gomes et al., 2013). Another approach, inspired by human perception and widely explored for 2D image segmentation, consists of saliency detection, which identifies the most prominent points within an image (Aytekin et al., 2018). Unlike the foveation, which processes arbitrary regions, saliency allows for highlighting image regions that are known to be more important.

This chapter proposes a solution to improve the standard local descriptors pipeline's performance for 3D object recognition from point clouds. The idea consists of adding a preliminary step, referred to as Saliency Boost, which filters the point clouds using a saliency mask to reduce the number of processed points and, consequently, the whole processing time. Besides, by selecting only salient regions, our approach may yield a reduction in the number of false positives, thereby often also enhancing object recognition accuracy.

5.1 FUNDAMENTALS

3D object recognition systems based on local descriptors typically deploy two stages, one carried out offline and the other online, referred to as training and testing, respectively. The training stage builds the database of objects, storing their features for later use. In the testing stage, then, features are extracted from scene images. Given a scene, the typical pipeline, depicted in Figure 5.1 and described, e.g., in Chen and Bhanu (2007), consists of the following steps 1) Keypoints extraction; 2) Local descriptors calculation; 3) Matching; 4) Grouping correspondences; and 5) Absolute orientation estimation (also presented in Figure 2.11). The first two, described in more detail below, are those that distinguish the various approaches and impact performance most directly.

5.1.1 Saliency Detection

Salient object detection is a topic inspired by human perception, which affirms that human beings tend to select visual information based on attention mechanisms in the brain (Kastner and Ungerleider, 2000). Its objective is to emphasize regions of interest in a scene (Aytekin et al., 2018). Many applications benefit from saliency, such as object tracking and recognition, image retrieval, restoring, and segmentation.

The majority of recent works perform saliency detection using either RGB (Hou et al., 2017; Aytekin et al., 2018; Liu et al., 2019a) or RGB-D (Li et al., 2018b; Chen et al., 2019) images and are based on Deep Learning algorithms.

5.2 PROPOSED APPROACH

We present a way to significantly improve the time performance and the memory efficiency of the standard pipeline described above by adding a step to the original pipeline. We refer to this step as *Saliency Boost*. It leverages the RGB scene image by detecting salient regions within it, which are then used to filter the point cloud and execute the local descriptors' pipeline only on salient regions. In particular, we use the saliency mask to reduce the search space for 3D keypoints by letting them run on the part of the point cloud, which corresponds to the salient regions of the image. To project saliency information from the 2D domain of the RGB image to the point cloud, we leverage RGB-D cameras' registration information. Figure 5.1 presents a graphical overview of the approach. In the case of 2D keypoint detectors, instead, we run them on the full RGB image, and we then filter out keypoints not belonging to the salient regions: we do not filter the image before the keypoint extraction step because 2D detectors also exploit pixels from the background to define blobs and edges/corners to detect keypoints. In the 3D case, instead, points from the background are usually far away, and outside the sphere used to define the keypoint neighborhood, so it is possible to filter them before without affecting the detector performance.

Our approach is not dependent on a specific saliency detection technique. In this work, we choose the DSS algorithm (Hou et al., 2017), and we detect salient areas by running the trained model provided by the authors.

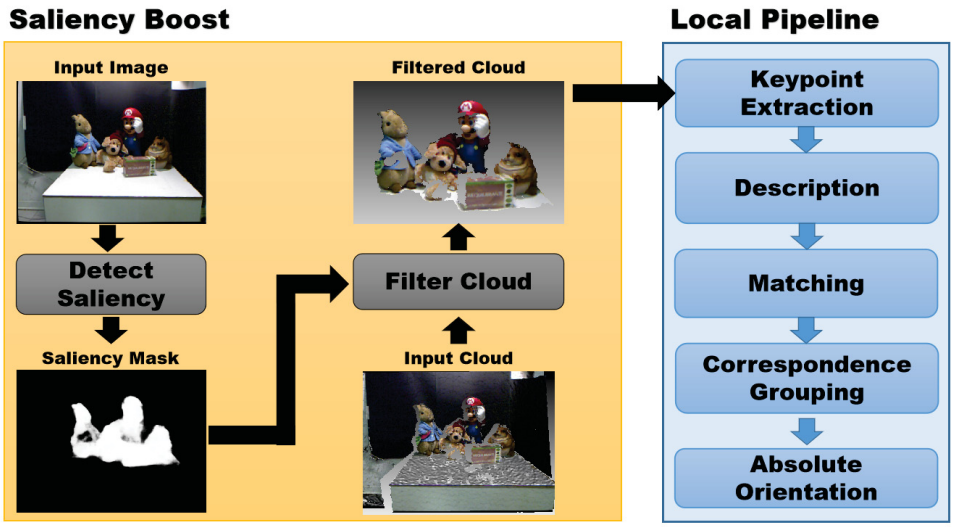


Figure 5.1: Local descriptor pipeline with saliency boost.

5.3 EXPERIMENTAL RESULTS

5.3.1 Local Descriptors Pipeline

In the local feature pipeline for object recognition, the choice of the keypoint extraction and description methods is key. It depends on the type of application, the kind of 3D representation available and resolution, or the inherent sensor noise. To evaluate the performance of the proposed approach in an application-agnostic scenario, we test combinations of several descriptors and detectors. The selected descriptors are: SHOT and CSHOT (Salti et al., 2014), FPFH (Rusu et al., 2009) and PFHRGB (Rusu et al., 2008). The keypoint detectors working on 3D data are Uniform sampling (US), with leaf sizes ranging from 2 to 5 cm with a step of 1 cm, and ISS (Zhong, 2009), while on images we test SIFT (Lowe, 1999) and FAST (Rosten and Drummond, 2006), run on the RGB image and projected on the point cloud, as discussed.

The matching step is performed by the NN search implemented by the FLANN library, integrated into the PCL (Rusu and Cousins, 2011). A KdTree is built for each view of each model in the database, and each keypoint on the scene is matched to only one point of one view of one model in the database by selecting the closest descriptor among views and models. After this process, all matches pointing to a view of a model are processed by the GCG algorithm (Chen and Bhanu, 2007), which selects all the subsets of geometrically consistent matches between the view and the scene, and estimates the aligning transformation. The transformation obtained from the largest correspondence group among all the object's views is considered the best estimation of the aligning transformation for that object. If an object fails to have a geometrically consistent subset with at least three matches among all its views, it is estimated as being not present in the considered scene.

The experiments were carried out on the Kinect dataset from the University of Bologna, presented in (Salti et al., 2014) and on the Washington RGB-D Object/Scenes datasets (Lai et al., 2011a).

5.3.2 Evaluation Protocol

To evaluate the performance of the proposed object detection pipeline, the correctness of predictions both of object presence and pose are evaluated. We adopt the Intersection over Union (IoU) metric (Equation 5.1), also known as the Jaccard index, and defined as the ratio between the intersection and the union of the estimated bounding box (BB_{Est}) and the ground truth bounding box (BB_{GT}).

$$IoU = \frac{BB_{GT} \cap BB_{Est}}{BB_{GT} \cup BB_{Est}} \tag{5.1}$$

A detection is evaluated as correct if its IoU with the ground truth is greater than 0.25, as in Song and Xiao (2016). Given detections and ground truth boxes, we calculate precision and recall (Equations 5.2 and 5.3) by considering a correct estimation as True Positive (TP), i.e., $IoU \geq 0.25$, an estimation of an absent object as False Positive (FP), and misdetections or detections with $IoU < 0.25$ as False Negative (FN).

$$precision = \frac{TP}{(TP + FP)} \tag{5.2}$$

$$recall = \frac{TP}{(TP + FN)} \tag{5.3}$$

To calculate precision-recall curves (PRC), we varied the threshold on the number of geometrically consistent correspondences to declare a detection, increasing it from the minimum value of 3 up to when no more detections are found in a scene. The area under the PRC curve (AUC) is then computed for each combination detector/descriptor and used to compare and rank the pipelines.

5.3.3 Implementation Details

Tests were performed on a Linux Ubuntu 16.04 LTS machine, using PCL version 1.8.1, OpenCV 3.4.1, and the VTK 6.2 library. For comparison purposes, all trials were performed on the same computer, equipped with an Intel Core i7-3632QM processor and 8GB of RAM. When available in PCL, the parallel version of each descriptor was used, e.g., for SHOT, CSHOT, and FPFH.

As for detectors' parameters, the ISS Non-Maxima Suppression radius was set to 0.6 cm, and the neighborhood radius to 1 cm, while for SIFT and FAST, we used the default values provided in OpenCV. As for descriptors, to estimate the normals, we used the first ten neighbors of each point while the description radius was set to 5 cm for all the considered.

5.3.4 Results

In this section, we present the results obtained in the experiments. All trials were performed on Bologna Kinect and Washington RGB-D Scenes datasets, comparing the original pipeline (blue part in Figure 5.1) with the proposed pipeline with saliency boosting. We tested seven keypoint extractors for each descriptor and each pipeline, totaling 56 trials for Bologna, and 52 for Washington. We chose to let the pairs PFHRGB and US at 2 and 3 cm away due to the high computational time of these configurations. The scene processing time, which comprises the saliency detection (only for the boosted pipeline), keypoint extraction, description, matching correspondences, clustering, and pose estimation, was measured to verify the proposed modification's impact on processing time.

Results in terms of the number of keypoints extracted are presented in Table 5.1. The saliency filtering significantly reduces the average number of keypoints extracted by each detector. Evaluating the Bologna dataset, the reduction using saliency boost ranges from 24.58% to almost 80% with an average of 56%. For Washington RGB-D Scenes, this result is even better when the reduction is from 51.27% to more than 88% for most detectors, and 77.51% on average.

Table 5.1: Average number of keypoints extracted from scenes in the trials with the traditional local pipeline (LP) and boosted by saliency (Boost). The column "%" represents the reduction between LP w.r.t. Boost. Best value in bold.

Keypoints	**Bologna**			**Washington**		
	LP	Boost	%	LP	Boost	%
FAST	489.71	369.36	24.58	806.77	393.17	51.27
ISS	4201.16	846.75	**79.84**	6405.17	1347.68	78.96
SIFT	282.79	199.79	29.35	373.56	179.22	52.02
$US_{0.02}$	4559.80	1457.86	68.03	10174.89	1171.85	88.48
$US_{0.03}$	2144.07	731.36	65.89	5340.91	613.74	**88.51**
$US_{0.04}$	1266.00	446.29	64.75	3368.94	394.41	88.29
$US_{0.05}$	820.57	303.71	62.99	2343.36	280.44	88.03
Average	-	-	56.49	-	-	76.51

The number of keypoints extracted directly impacts the pipeline's running time, mainly by two factors: the number of descriptors that have to be computed and the time it takes to match them. The SHOT and CSHOT descriptors are calculated relatively fast, but due to their length

(352 and 1344 bins respectively), the matching phase is slower, accounting for 97 and 99% of the processing time. The PFHRGB and FPFH are shorter descriptors (250 and 33 bins, respectively), but the description is slower and requires 94 and 89% of the overall time.

As shown in Tables 5.2 and 5.3, the extraction of keypoints only in salient regions reduces the processing time for both kinds of descriptors drastically. In the best case, processing time reduction is as high as 80%, i.e., the boosted pipeline is five times faster due to the proposed saliency boosting. For all the considered detector/descriptor combinations, deployment of the saliency boosting step always reduces the processing time significantly. For the Bologna dataset, these reductions are from 22% for FAST/SHOT to 83% for ISS and $US_{0.05}$ with FPFH. In Washington RGB-D Scenes, the improvement is even better, ranging from 51% on FAST/CSHOT pair to 88%, with a consistent reduction of more than 70% on average for all descriptors.

Despite the results consonant with faster processing time, we observe that SIFT and FAST keypoint extractors present the lowest improvements concerning the other detectors. These results are expected due to their 2D nature, as well as the saliency detection. In the case of 3D extractors, this decrease is more consistent. A reason for that could be the presence of areas with higher 3D roughness and lower 2D texture variance (naturally non-salient), or even the high amount of detected points on more distant background areas.

Table 5.2: Average scene processing time (s) on the Bologna Kinect dataset in the trials with the traditional Local Pipeline (LP) and boosted by saliency (Boost). The column "%" represents the reduction between LP w.r.t. Boost. Best value in each column in bold.

Bologna Kinect dataset												
Keypoints	**CSHOT**			**SHOT**			**PFHRGB**			**FPFH**		
	LP	Boost	%	LP	Boost	%	LP	Boost	%	LP	Boost	%
FAST	244.0	174.8	28.36	59.1	45.9	22.31	351.4	238.8	32.06	46.6	19.0	59.14
ISS	226.1	**47.7**	**78.90**	72.4	**17.1**	**76.45**	2580.4	489.1	**81.05**	141.9	24.3	82.92
SIFT	**132.2**	94.5	28.50	**34.3**	25.7	25.29	**195.3**	138.2	29.24	**31.3**	**17.7**	43.39
$US_{0.02}$	2167.9	668.8	69.15	505.7	174.5	65.50	2100.9	455.5	78.32	150.6	29.6	80.32
$US_{0.03}$	988.0	335.6	66.04	238.8	88.0	63.16	913.2	191.5	79.03	137.4	24.1	82.47
$US_{0.04}$	583.4	205.2	64.83	139.7	54.1	61.29	506.1	103.5	79.56	130.9	22.2	83.08
$US_{0.05}$	378.1	139.9	63.01	90.7	37.2	58.99	304.3	**62.1**	79.61	128.7	20.9	**83.76**
Average			56.97			53.29			65.55			73.58

Table 5.3: Average scene processing time (s) on the Washington RGB-D Scenes dataset in the trials with the traditional Local Pipeline (LP) and boosted by saliency (Boost). The column "%" represents the reduction between LP w.r.t. Boost. Best value in each column in bold.

Washington RGB-D Scenes dataset												
Keypoints	**CSHOT**			**SHOT**			**PFHRGB**			**FPFH**		
	LP	Boost	%	LP	Boost	%	LP	Boost	%	LP	Boost	%
FAST	514.5	250.5	51.31	129.8	62.1	52.12	707.0	294.7	58.32	63.5	16.5	73.97
ISS	1424.8	306.6	78.48	369.0	250.5	32.12	3719.4	761.8	79.52	146.2	27.7	81.05
SIFT	238.6	114.4	52.05	**60.7**	28.6	52.99	324.7	139.9	56.90	**49.2**	**13.0**	73.58
$US_{0.02}$	1043.5	119.5	**88.55**	281.6	32.6	88.41	-	-	-	129.9	19.8	**84.80**
$US_{0.03}$	558.5	64.6	88.43	147.6	17.1	**88,44**	-	-	-	113.4	17.8	84.31
$US_{0.04}$	344.7	40.7	88.19	92.8	11.0	88.20	486.7	80.8	**83.40**	106.9	17.2	83.91
$US_{0.05}$	**235.9**	**28.5**	87.93	64.4	**7.8**	87.86	**311.7**	**52.7**	83.10	103.4	17.0	83.56
Average			76.42			70.02			72.25			80.74

Reducing processing time is only beneficial if it does not harm recognition and localization performance. Interestingly, deployment of the saliency boosting step very often

improve AUC concerning the traditional pipeline, as shown in Table 5.4 for Bologna Kinect, and Table 5.5 for Washington RGB-D Scenes datasets.

Table 5.4: AUC results in the trials for Bologna dataset with the traditional Local Pipeline (LP) and boosted by saliency (Boost). The column "%" represents the variation between them and negative values represent an accuracy performance lost. Best value in each column in bold.

Bologna Kinect dataset												
	CSHOT			**SHOT**			**PFHRGB**			**FPFH**		
Keypoints	**LP**	**Boost**	**%**	**LP**	**Boost**	**%**	**LP**	**Boost**	**%**	**LP**	**Boost**	**%**
FAST	0.946	0.874	-7.61	0.915	0.892	-2.45	0.743	0.761	2.43	0.631	0.668	5.89
ISS	0.868	0.881	1.52	0.866	0.912	**5.30**	**0.745**	**0.900**	20.68	0.491	**0.752**	**53.04**
SIFT	0.864	0.889	2.83	0.903	0.820	-9.15	0.472	0.549	16.41	0.529	0.476	-10.13
$US_{0.02}$	**0.949**	**0.948**	-0.07	**0.941**	**0.938**	-0.31	0.739	0.807	9.19	**0.641**	0.728	13.48
$US_{0.03}$	0.861	0.905	5.08	0.875	0.843	-3.58	0.731	0.814	11.37	0.488	0.621	27.26
$US_{0.04}$	0.832	0.875	5.23	0.824	0.817	-0.92	0.564	0.700	24.22	0.289	0.368	27.14
$US_{0.05}$	0.582	0.619	**6.19**	0.682	0.644	-5.64	0.373	0.599	**60.76**	0.145	0.162	11.77
Average			1.88			-2.39			20.72			18.35

Analyzing first, the results from the Bologna Kinect dataset, particularly for 19 of the 28 trials, which included the saliency boosting step, the pipeline boosted by saliency performed better on AUC, with significant improvements by more than 50% for PFHRGB and FPFH. Viceversa, when the AUC decreases due to the saliency boost's deployment, it usually does it marginally, by 1 or 2%, with the worst decrease in AUC being more significant than 10% only once when using the SIFT detector. While the AUC generally increases with the boosted pipeline, it does not do so on average when deployed with the SHOT descriptor. However, it does increase by 5% in the very relevant case of combining SHOT with the ISS detector, the combination that delivers the fastest running time among all the tested variants (as shown in Table 5.2).

Table 5.5: AUC results in the trials for the Washington RGB-D Scenes dataset with the traditional Local Pipeline (LP) and boosted by saliency (Boost). Best value in each column in bold.

Washington RGB-D Scenes dataset								
Keypoints	**CSHOT**		**SHOT**		**PFHRGB**		**FPFH**	
	LP	**Boost**	**LP**	**Boost**	**LP**	**Boost**	**LP**	**Boost**
FAST	0.0406	0.1062	0.0227	0.0558	**0.0351**	0.0544	0.0014	0.0053
ISS	0.0568	0.1329	0.0930	**0.1708**	0.0082	**0.0621**	**0.0047**	**0.0237**
SIFT	0.0241	0.0736	0.0078	0.0220	0.0063	0.0159	0.0002	0.0010
$US_{0.02}$	**0.0963**	**0.1957**	**0.0998**	0.1706	-	-	0.0006	0.0017
$US_{0.03}$	0.0346	0.1069	0.0407	0.1043	-	-	0.0002	0.0048
$US_{0.04}$	0.0078	0.0525	0.0186	0.0604	0.0020	0.0017	0.0000	0.0001
$US_{0.05}$	0.0003	0.0166	0.0037	0.0197	0.0000	0.0021	0.0000	0.0019

Based on the evaluations, we encounter an extremely challenging dataset on the Washington RGB-D Scenes on the proposed methodology. We face an accuracy lower than 20% in the best case ($US_{0.02}$/CSHOT) considering the AUC results. Furthermore, in many situations, we get AUC lower than 1%, particularly concerning the FPFH and PFHRGB descriptors.

The proposed boosting represents an undeniable gain regarding the improvement in accuracy, performing better for every trial, except the $US_{0.04}$/PFHRGB pair. Despite that, we do not report this % improvement w.r.t. the traditional pipeline as in Table 5.4. The reason is that as the performances are too low in some cases, reporting such improvement could not reflect a fair comparison, e.g., for the $US_{0.05}$/CSHOT pair, the AUC of the boosted is $61\times$ higher than the traditional pipeline, but still around 1%. Performing a more rational analysis and considering

only the best cases of each pipeline, we see a massive improvement of nearly 100% for all the descriptors.

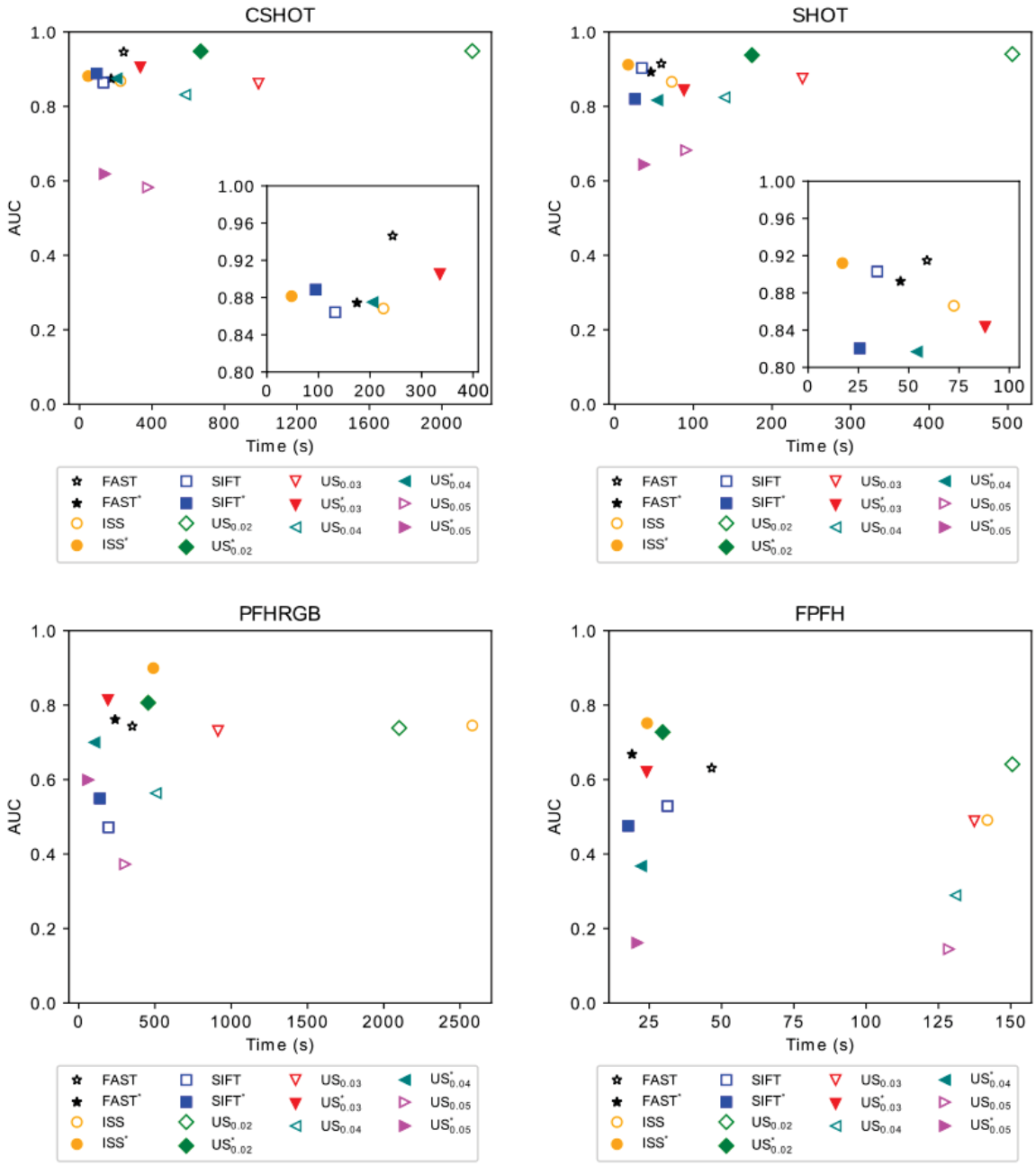


Figure 5.2: AUC $\times$ Time Results for the descriptors on the Bologna dataset. Boosted pipeline denoted by an asterisk (*) next of the keypoint name and a filled marker.

Finally, in Figure 5.2, we report a Pareto analysis of the data for all descriptors, considering the Bologna Kinect dataset. We can see how the boosted pipeline's execution obtains points, i.e., detector/descriptor pairs closer to the ideal position (that is $AUC = 1$ and time as low as possible). In this analysis, the CSHOT, SHOT, and FPFH obtained the best performance when paired with the boosted ISS (ISS*), while PFHRGB when paired with the Boosted US at $r = 3cm$ ($\text{US}^*_{0.03}$). Hence, the boosting pipeline outperforms the traditional one for all tested descriptors when considering the combined effect of processing time and recognition performance.

For the Washington RGB-D Scenes dataset, the Pareto analysis is not required because it clear that the boosted pipeline outperforms the traditional one. As reported in Tables 5.3 and 5.5, we can see a substantial improvement in the processing time as well the accuracy, represented by the AUC, for almost all pairs detector/descriptor of the executed trials.

5.4 FINAL REMARKS AND OVERVIEW

This chapter presented an approach based on saliency detection to boost the traditional local descriptor pipeline in terms of processing time and eventually on accuracy. Results regarding the application of the boosted pipeline on the Bologna Kinect dataset were previously published in Marcon et al. (2019).

We evaluated our proposal in two object recognition datasets and verified that all the tested cases had a significant processing time reduction, from 22 to 88%. Interestingly, the processing time reduction did not generally decrease the object recognition performance, as measured by the AUC of the precision-recall curves. Regarding the Bologna Kinect dataset, we found consistent improvements in the performance recognition for all descriptors in at least one pairing, up to 5% for SHOT and CSHOT, and more than 50% for FPFH and PFHRGB. Considering the Washington RGB-D Scenes dataset, the improvements are even better, with more than a 70% reduction in time processing and the double accuracy performance achieved by our proposal. However, this particular dataset presents challenging situations, and the results are coarse, but still endorses the benefits of employing our approach on object recognition tasks.

Despite the improvements in processing time, the whole processing time is not suitable for real-time applications yet. However, the proposed approach offers a considerable speed-up without impacting recognition performance negatively, which brings us a step closer to creating a compelling and real-time local feature pipeline for 3D object recognition.

6 THE COLOR AND THE SHAPE

Deep learning strategies for object recognition and classification problems have been extensively studied for RGB images. As the demand for good quality labeled data increases, large datasets are becoming available, serving not only as a significant benchmark of methods (deep or not) but also as training data for real applications. ImageNet (Deng et al., 2009) is, undoubtedly, the most studied dataset, and the *de-facto* standard on such recognition tasks. This dataset presents more than 20,000 categories, but a subset with 1,000 categories, known as ImageNet Large Scale Visual Recognition Challenge (ILSVRC), is mostly used.

Training a model on ImageNet is quite a challenging task in terms of computational resources and time consumption. Fortunately, transferring its models offer efficient solutions in different contexts, acting as a blackbox feature extractor. Agrawal et al. (2014) and Huh et al. (2016) explore and corroborate with this high capacity of transferring models trained on ImageNet.

In consonance with the results attained on RGB images only, many works have explored pre-trained models on ImageNet on RGB-D images, achieving state-of-the-art results on challenging object recognition datasets. Bui et al. (2016) by using AlexNet, Zia et al. (2017) with VGG, (Caglayan and Burak Can, 2018), and finally, Caglayan et al. (2020) performed a comparison with several architectures associated with Recursive Neural Networks (RNN). Despite the performance of pre-trained descriptors on RGB images, exploring depth information is also valuable (Lai et al., 2011a), and studying learning shape[1] feature-extractors on object recognition is crucial to evolve such applications.

Based on the above-mentioned, the contribution of this chapter is threefold. First, we evaluate traditional to modern architectures of deep-learning-based networks applied in the Washington RGB-D Object dataset. We evaluate these architectures in terms of category detection and instance recognition. Furthermore, we explore the capability of local learned descriptors, acting in a global context. Finally, we propose using the RGB pre-trained models associated with global descriptors of the shape. To extract shape features, we employ unsupervised approaches and consider models trained from scratch and fine-tuned. Results show that this association is beneficial, increasing the accuracy concerning color-only features.

6.1 RELATED WORKS

6.1.1 Color feature extraction

As a mark on the deep learning history, Krizhevsky et al. (2012) presented the first deep convolutional architecture employed on the ILSVRC, an 8-layer architecture dubbed AlexNet. This network was the first to prove that deep learning could beat hand-crafted methods when trained on a large scale. After that, convolutional networks (ConvNets for short) became more accurate, deeper, and bigger in terms of parameters. (Simonyan and Zisserman, 2015) propose VGG, a network that doubled the depth of AlexNet, but exploring tiny filters (3×3), and became the runner-up on the ILSVRC, one step back the GoogLeNet (Szegedy et al., 2015), with 22 layers. GoogLeNet relies on the Inception architecture, and for this reason, it is also named

[1]Throughout this chapter, we adopt the term shape referring to geometrical information regarding 3D models. We do not perform shape feature extraction on RGB images.

Inception v1. After GoogLeNet come the Inception v2 and v3 (Szegedy et al., 2016), and v4 (Szegedy et al., 2017) architectures, always deeper and bigger.

Another type of ConvNets, called ResNets, uses the concept of residual blocks that use skip-connection blocks that learn residual functions regarding the input. Given an input $\mathbf{x}$ and a block output $\mathcal{F}(\mathbf{x})$ the output of a block that uses skip-connections will be $\mathcal{F}(\mathbf{x}) + \mathbf{x}$. In practice, the residual mapping facilitates the optimization process and extracts more high-level features, exploring deeper architectures. Based on these findings, many architectures have been proposed, such as ResNet with 50, 101, 152 (He et al., 2016a), and 200 layers (He et al., 2016b). Also, based on developments regarding the residual blocks, Xie et al. (2017) developed the ResNeXt architecture. The basis upon ResNeXt blocks resides on the use of parallel ResNet-like blocks that have the output summed before the residual calculation. Another characteristic of such blocks is to reduce the bottleneck of each sub-block to a size of d. ResNeXt architectures nomenclature refers to the number of layers, number of concurrent ResNet blocks, and bottleneck output. For instance, the best accurate network of this family reported on the ImageNet leaderboard[2] is the ResNeXt-101 32×48d with 101 layers. Each ResNeXt block in such architecture has 32 ResNet concurrent blocks and a bottleneck of size 48, totaling 829M parameters.

Some architectures propose using deep learning features on resource-limited devices, such as smartphones and embedded systems. The most prominent architecture is the MobileNet, with the v1 (Howard et al., 2017), v2 (Sandler et al., 2018), and v3 (Howard et al., 2019). Another family of leading networks is the EfficientNet (Tan and Le, 2019). Relying on the use of these lighter architectures, EfficientNet proposes profound networks without compromise resource efficiency. The authors propose eight architectures starting from 237 to 813 layers (EfficientNet-B0 and B7, respectively). Table 6.1 presents a comparison between some ConvNets regarding the depth (number of layers), amount of parameters, and accuracy on the ImageNet benchmark.

Table 6.1: Deep architectures performance on ImageNet. The column Accuracy refers to the ILSVRC. Year refers to the oficial availability of the referred publication.

Architecture	Layers	Accuracy (%)	Parameters	Year
AlexNet (Krizhevsky et al., 2012)	8	63.3	60M	2012
GoogLeNet (Szegedy et al., 2015)	22	69.8	5M	2014
VGG16 (Simonyan and Zisserman, 2015)	16	74.4	138M	2014
ResNet 50 (He et al., 2016a)	50	77.15	25.6M	2015
ResNet 101 (He et al., 2016a)	101	78.3	40M	2015
ResNet 200 (He et al., 2016b)	200	79.9	63M	2016
Inception v2 (Szegedy et al., 2016)	22	74.8	11.2M	2016
Inception v3 (Szegedy et al., 2016)	48	78.8	24M	2016
ResNeXt-101 32×8d (Xie et al., 2017)	101	82.2	88M	2017
ResNeXt-101 32×48d (Mahajan et al., 2018)	101	85.4	829M	2018
MobileNet v1 (Howard et al., 2017)	28	70.6	4.2M	2017
MobileNet v2 (Sandler et al., 2018)	53	74.7	6.9M	2018
MobileNet v3 (Howard et al., 2019)	53	75.2	5.4M	2019
EfficientNet B0 (Tan and Le, 2019)	237	76.3	5.3M	2019
EfficientNet B7 (Tan and Le, 2019)	813	84.4	66M	2019

[2]Available on `https://paperswithcode.com/sota/image-classification-on-imagenet`

6.1.2 RGB-D object recognition

Many works address the problem of object recognition in RGB-D images. State-of-the-art methods explore two main trends: based on pre-trained CNNs and using covariance descriptors. This section explores both families of methods, pointing to the most successful techniques used for object classification and instance recognition. Starting from the pivotal work of Lai et al. (2011a) that proposes the RGB-D Object and Scenes dataset, providing as the baseline, recognition results using descriptors for color, based on the use of SIFT (Lowe, 1999), and for shape information, by using the Spin Image (Johnson and Hebert, 1999) descriptor.

Methods that explore covariance descriptors provide compact feature vectors for visual and geometric information, found in point clouds. One may consider the approaches of Fehr et al. (2014), Beksi and Papanikolopoulos (2015), and Zhang et al. (2017) on the object recognition task. Despite efficiency, these methods introduce hand-crafted approaches and depend on arbitrary assumptions, not directly related to the data.

We present some approaches applied to the object categorization task on the RGB-D Object Dataset, regarding the use of pre-trained CNN models attained from the ImageNet dataset. Some approaches handle the object recognition task by a multi-modal strategy based on the combination of RNN and explore high-level features from pre-trained CNNs. As example we have Bui et al. (2016) that combine RNN and AlexNet, Caglayan and Burak Can (2018) with VGG pre-trained models, and Caglayan et al. (2020) that perform an evaluation on several popular architectures, such as AlexNet, VGG, ResNet, and DenseNet. Zia et al. (2017) propose a new CNN architecture, which is an RGB-D extension of the VGG16 network, based on the projection of the features maps 2D on the 3D domain.

6.2 PROPOSED APPROACH

As previously pointed, this work proposes a joint adoption of color and shape feature extractors. For color descriptors, we explore the following networks: AlexNet, VGG16, ResNet101, Inception v3, MobileNet v2, ResNeXt101 32×8d, and EfficientNet B7. We extracted only the bottleneck feature map for all these architectures, which corresponds to 1000 bins descriptors. For shape descriptors, we consider three approaches based on plane folding: the LEAD and LEAD-PN, presented in Chapter 4 and a fine-tuning on the model presented by Spezialetti et al. (2019), all of them 512 bins long. Despite these approaches being originally conceived to describe local features, they were adapted to a global context in this work.

As depicted in Figure 6.1, this work proposes concatenating color and shape descriptors and verifying if this joint adoption is beneficial to the object recognition task. Before passing through the classifier, we preprocess both features to perform scaling on the training set by subtracting the mean and scaling to unit variance, following Equation 6.1. The obtained scaler for training is then used on the test set, i.e., using the training set's mean and variance. We employed in the context of this work, the Gaussian Naïve Bayes (GNB), Support-Vector Classifier (SVC), Random-Forest (RF), Logistic Regression (LR), and a Multi-layer perceptron (MLP) classifiers.

$$\hat{\mathbf{X}} = \frac{\mathbf{X} - \mu}{\sigma} \tag{6.1}$$

The training process of shape descriptors for the LEAD descriptor followed the methods presented in Chapter 4. For LEAD-PN, we adopted a training process proposed by Groueix et al. (2018), and for the fine-tuned descriptor, we follow Spezialetti et al. (2019). However, as these approaches are designed for local patches, we had to globalize the local patches fed as input. We searched the centroid's nearest point on the object surface to emulate the keypoint, the whole

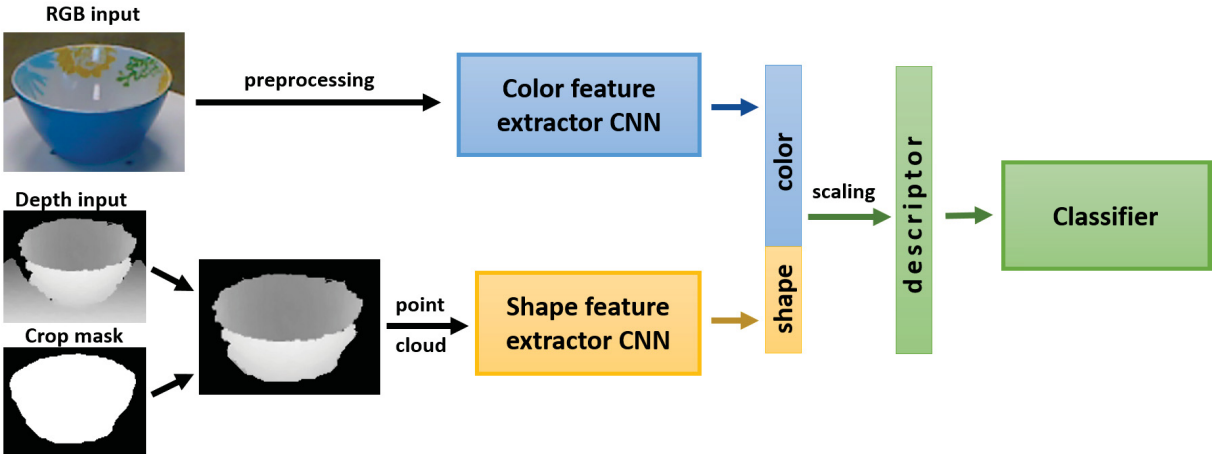


Figure 6.1: Combination of color and shape descriptors scheme for proposed approach.

cloud as the patch, and the full-cloud radius as support. We perform this computation process individually, and trained in a canonical representation, transforming the input by the inverse of the LRF transformation, extracted from the patches, using FLARE (Petrelli and Di Stefano, 2012).

6.2.1 Evaluation protocol

We evaluate the proposed method on the Washington RGB-D Object dataset (Lai et al., 2011a), a dataset containing around 45,000 RGB-D images of 300 object instances of 51 different household categories objects. Instances are stored as three sequences of contiguous frames captured from a rotating table at $30°$, $45°$, and $60°$ concerning the camera and the object. We evaluate object recognition performance on the category (e.g., soda can or coffee mug) and instance (e.g., Pepsi can or Mountain Dew can) levels. To do so, we follow the standard protocol presented by the authors (Lai et al., 2011a).

In the category level, we trained the system using the ten splits provided by the authors. On each split, for each category, one instance is randomly selected to compound the test set, and the others are for training. At the instance level, the protocol considers two distinct scenarios: alternating contiguous frames (ACF) and leave-sequence-out (LSO). In the former, we divide each sequence into three subparts of equal length, totalizing nine sequences, so we randomly select seven of them for training and two for testing. The latter consists of using the sequences captured at $30°$ and $60°$ for training and $45°$ for testing. After running the trials, we compute the average accuracy for category level and instance recognition with ACF. There is no randomness on the LSO scenario, so we report a single trial accuracy. All the RGB inputs are resized to 224×224 pixels, and normalized by a mean of $(0.485, 0.456, 0.406)$ and standard deviation of $(0.229, 0.224, 0.225)$, following the expected values on the torchvision pre-trained models[3]. To preprocess depth input, we cropped the object's background, applying the provided masks, and after that, we convert it to a point cloud representation.

6.2.2 Implementation details

We used a Linux Ubuntu 18.04 LTS machine to perform tests, with a CPU Ryzen 7 2700X eight-core processor, 32GB of RAM, and a GPU RTX 2070 Super. We performed all the implementations regarding deep architectures on the Pytorch framework and acquired the pre-trained models from the Torchvision library. We use the FLARE (Petrelli and Di Stefano, 2012)

[3]Available on `https://pytorch.org/docs/stable/torchvision/models.html`

implementation from PCL library 1.8.1, using default parameters, and the ML classifiers are from the Scikit-learn library.

6.3 EXPERIMENTAL RESULTS

In this section, we show the results achieved in our experiments. We carried out all trials on the Washington RGB-D Object dataset, regarding category and instance recognition. Firstly we tested the accuracy performance of features extracted by CNN pre-trained models. We also compared these features by variating the ML classifier. For this particular trial, the LR classifier has outperformed the others for all architectures. Hence, we report only the best results in Table 6.2, and a complete comparison is provided in Appendix A, highlighting the six classifiers analyzed.

Table 6.2: Comparison of color features from CNN architectures on the Washington RGB-D Object dataset. The best result reported in **blue**, the second best in **green**, and the third in **red**.

Method	Category	Instance (LSO)	Instance (ACF)
Lai et al. (2011a)	74.7 ± 3.6	60.7	91.0 ± 0.5
AlexNet	73.0 ± 2.6	89.8	93.9 ± 0.4
ResNet101	**83.4 ± 2.3**	**94.1**	95.3 ± 0.3
VGG16	77.5 ± 2.6	88.8	91.0 ± 0.6
Inception v3	81.0 ± 2.4	88.1	90.3 ± 0.4
MobileNet v2	82.4 ± 2.4	**93.8**	**95.8 ± 0.3**
ResNeXt101 32×8d	**85.0 ± 2.1**	**93.9**	**95.7 ± 0.4**
EfficientNet B7	**86.3 ± 3.1**	**93.8**	**95.6 ± 0.5**

According to Table 6.2, we have four architectures that present at least one result on the top three. Regarding category recognition, the best model is the EfficientNet-B7 (Tan and Le, 2019) architecture, which presents 86.3% accuracy. Attending the instance recognition task, we have the ResNet101 (He et al., 2016a) leading when considering the LSO with 94.1%, and MobileNet v2 (Sandler et al., 2018), with 95.8% accuracy on the ACF scenario. The ResNext101 (Xie et al., 2017) architecture deserves attention, starring as runner-up on the three recognition situations.

We also report the performance of the global version of the proposed descriptors (Chapter 4) and the fine-tuned version of Spezialetti et al. (2019). The results shown in Table 6.3 demonstrate that this approach may not be so useful the way it is, performing worst than the baseline proposal of Lai et al. (2011a). Among our proposals, Spezialetti et al. (2019) reaches the best results by its global fine-tuned version, for all experimented scenarios.

Table 6.3: Comparison of shape features on the Washington RGB-D Object dataset. Best result of each column in **bold**.

Method	Category	Instance (LSO)	Instance (ACF)
Lai et al. (2011a)	**66.8 ± 2.5**	**46.5**	**52.7 ± 1.0**
LEAD-PN	36.1 ± 1.2	13.0	17.5 ± 0.5
LEAD	50.4 ± 1.0	23.4	30.3 ± 0.6
Spezialetti et al. (2019)	51.2 ± 1.4	25.3	31.7 ± 0.8

Table 6.4: Category recognition on the Washington RGB-D Object dataset. The best result reported in **blue**, the second best in **green**, and the third in **red**.

	Category		
Method	Color	Shape	Color + Shape
Lai et al. (2011a) (RF)	74.7 ± 3.6	66.8 ± 2.5	79.6 ± 4.0
Lai et al. (2011a) (kSVC)	74.5 ± 3.1	64.7 ± 2.2	83.8 ± 3.5
Subset-RNN (Bai et al., 2015)	82.8 ± 3.4	81.8 ± 2.6	88.5 ± 3.1
Fusion 2D/3D CNNs (Zia et al., 2017)	89.0 ± 2.1	**89.0** ± 2.1	91.8 ± 0.9
MM-LRF-ELM (Liu et al., 2018)	84.3 ± 3.2	82.9 ± 2.5	89.6 ± 2.5
VGG f-RNN (Caglayan and Burak Can, 2018)	**89.9 ± 1.6**	84.0 ± 1.8	92.5 ± 1.2
MDSI-CNN (Asif et al., 2017)	**89.9 ± 1.8**	84.9 ± 1.7	**92.8 ± 1.2**
HP-CNN (Zaki et al., 2019)	87.6 ± 2.2	85.0 ± 2.1	91.1 ± 1.4
RCFusion (Loghmani et al., 2019)	**89.6 ± 2.2**	**85.9** ± 2.7	**94.4 ± 1.4**
ResNet101-RNN (Caglayan et al., 2020)	**92.3 ± 1.0**	**87.2** ± 2.5	**94.1 ± 1.0**
ResNet101 + LEAD (Ours)	83.3 ± 2.3	50.4 ± 1.0	86.4 ± 1.8
MobileNet v2 + LEAD (Ours)	83.4 ± 2.4		85.4 ± 2.2
ResNeXt101 32× 8d + LEAD (Ours)	85.0 ± 2.1		88.2 ± 1.9
EfficientNet B7 + LEAD (Ours)	86.3 ± 3.1		88.7 ± 2.1

Notwithstanding the results of shape-only approaches on the recognition task, we evaluate if the proposed scheme, as in Figure 6.1, presents a valid method in such applications, i.e., concatenate color and shape features. Tables 6.4, 6.5, and 6.6 summarize performance results regarding category and instance recognition (LSO and ACF scenarios). We compare accuracies considering color features (obtained by the CNN pre-trained networks), shape features (global versions of proposed methods), and the joint approach (concatenation of color and shape features). Considering the four best color feature extractors previously selected, three shape feature extractors, and six machine learning classifiers, we have performed a combination of them, resulting in 72 trials for each recognition scenario. However, we report the best combination of each color descriptor. We present a complete evaluation in Appendix A.

Table 6.4 reports a comparison of our proposal with state-of-the-art methods concerning category detection. We observe that despite achieving a combined performance (Column *Color + Shape*) of over 85% in all situations, our proposals are insufficient compared to state-of-the-art approaches. However, we must point out that the jointly introduced method presented slightly better results regarding color-only strategies (Column *Color*), with improvements from 1.4 to 3.1%. Finally, we point that the LEAD descriptor provided the best improvements for the category recognition task when associated with the color feature extractors.

Next, we evaluate the recognition task when considering an instance detection circumstance. In Tables 6.5 and 6.6, we report such results regarding the LSO and ACF methodologies (Lai et al., 2011a). First, we observe that differently from the category scenario, for recognition of instances, Spezialetti et al. (2019) fine-tuned descriptor is better suitable to our approach combination in most situations, with an exception on the ResNet101 for LSO.

In an LSO evaluation scenario, our proposal becomes more competitive with state-of-the-art approaches. We observe that ResNet101 and LEAD's combination brings us in the top-three considering color-only approaches, and in the fourth position, only 0.1% behind the third best.

Table 6.5: Instance recognition on the Washington RGB-D Object dataset (Leave-sequence-out). The best result reported in **blue**, the second best in **green**, and the third in **red**.

Instance (leave-sequence-out)			
Method	Color	Shape	Color + Shape
Lai et al. (2011a) (RF)	59.9	45.5	73.1
Lai et al. (2011a) (kSVC)	60.7	46.2	74.8
Kernel descriptor (Bo et al., 2011)	90.8	**54.7**	91.2
SP+HMP (Bo et al., 2013)	92.1	51.7	92.8
Multi-Modal (Schwarz et al., 2015)	92.0	-	94.1
CDDL (Beksi and Papanikolopoulos, 2015)	-	-	93.7
PCC Desc. (Zhang et al., 2017)	92.9	**53.7**	**94.6**
MDSI-CNN (Asif et al., 2017)	**97.7**	**57.6**	**97.9**
MM-LRF-ELM (Liu et al., 2018)	91.0	50.9	92.5
HP-CNN (Zaki et al., 2019)	**95.5**	50.2	**97.2**
ResNet101 + LEAD (Ours)	**94.1**	23.4	94.5
MobileNet v2 + Spezialetti (Ours)	93.8		93.8
ResNeXt101 32× 8d + Spezialetti (Ours)	93.9	25.3	93.9
EfficientNet B7 + Spezialetti (Ours)	93.8		93.8

When we analyze the ACF scenario (Table 6.6), we face a predominance of our proposed techniques. Considering a color-only feature vector, the pre-trained CNN models outperform the state-of-the-art methods, featuring the first four positions among the competitors. Surprisingly, the best accuracy was attained by the MobileNet v2 (Sandler et al., 2018), which is an architecture focused on limited resources devices. Such results encourage more studies on real-time instance recognition applications. Considering the joint proposal, again, the MobileNet v2 network combined with Spezialetti et al. (2019) descriptor, reaches the third position in a joint feature condition.

To examine more deeply our results, we plot the confusion matrices of our best networks regarding category and instance recognition scenarios. We depict such products in Figures 6.2 and 6.4.

From the category perspective, we observe a highly apparent principal diagonal. However, two categories have discrepant performance. We see that our proposal fails to predict the *mushroom* and *food_jar* categories, returning the *garlic* and *shampoo* classes, respectively. As we can catch in Figure 6.3, these misclassifications are plausible. First, when we observe samples of the *garlic* category, some are easily confounded with *mushrooms*, including by a human being. The second example is more related to the preprocessing step we perform on data. *Food_jar* and *shampoo* containers are very different in terms of height and width. However, the networks trained on ImageNet expect a squared image (224×224 pixels), and when we prepare them to apply to the system, this resizing process does not maintain the aspect ratio. In Figure 6.3, we present all the examples after performing this cited resizing step, and it is clear the similarity between them. Another detail that could complicate the features' discriminability is the background similarity between them, affecting especially the *food_jar* and *shampoo* categories.

Table 6.6: Instance recognition on the Washington RGB-D Object dataset (Alternating Contiguous Frame). The best result reported in **blue**, the second best in **green**, and the third in **red**. *Standard deviation doesn't reported by the authors.

Instance (alternating contiguous frames)			
Method	Color	Shape	Color + Shape
Lai et al. (2011a) (RF)	90.1 ± 0.8	**52.7 ± 1.0**	90.5 ± 0.4
Lai et al. (2011a) (kSVC)	91.0 ± 0.5	51.2 ± 0.8	90.6 ± 0.6
IDL (Lai et al., 2011b)	54.8 ± 0.6	**89.8 ± 0.2**	91.3 ± 0.3
CKM Desc. (Blum et al., 2012)	-	-	92.1 ± 0.4
CDSVM (Fehr et al., 2014)	-	-	94.4 ± 2.0
CDDL (Beksi and Papanikolopoulos, 2015)	-	-	**96.9 ± 0.5**
PCC Desc. (Zhang et al., 2017)*	92.9	**53.7**	**97.1 ± 1.8**
ResNet101 + Spezialetti (Ours)	95.3 ± 0.3	30.3 ± 0.7	95.6 ± 0.4
MobileNet v2 + Spezialetti (Ours)	**95.9 ± 0.3**		**95.9 ± 0.2**
ResNeXt101 32× 8d + Spezialetti (Ours)	**95.7 ± 0.4**		95.7 ± 0.2
EfficientNet B7 + Spezialetti (Ours)	**95.6 ± 0.5**		95.0 ± 0.4

Figure 6.3: Examples of misclassified categories of our proposed method. Our method infer wrongly the garlic category as being mushroom, and the food jar as being shampoo.

Now, looking at instance recognition results, we face a different kind of problem. The main diagonal is almost a straight line on the confusion matrix in our method. However, focusing on the details, we observe some intraclass misclassifications that impact the final accuracy results. We depict some of them that are most obvious in Figure 6.4. At the beginning of the diagonal line, we face an intraclass problem case by the *banana* category. Going further, near the middle of the line, we have the most visible problems, regarding the *lemon* and *lime* classes, and, almost at the end of the diagonal, we face issues caused by the *staples* category. In Figure 6.5, we perceive that these particular cases are very challenging, and mainly on the lemon and lime samples, the discriminability is impaired.

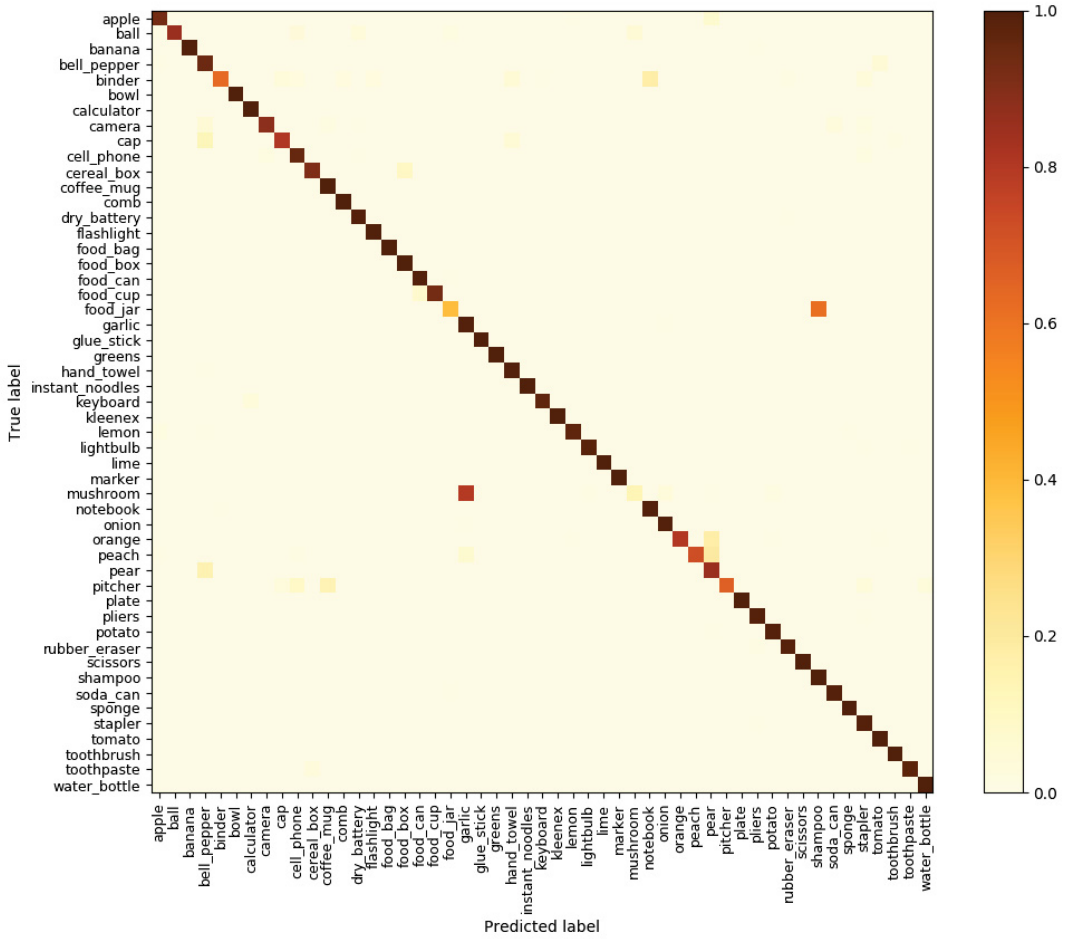


Figure 6.2: Confusion matrix of category recognition of the proposed method. This matrix corresponds to the EfficientNet-B7+LEAD method, trained on the *Split 2*, according to Lai et al. (2011a).

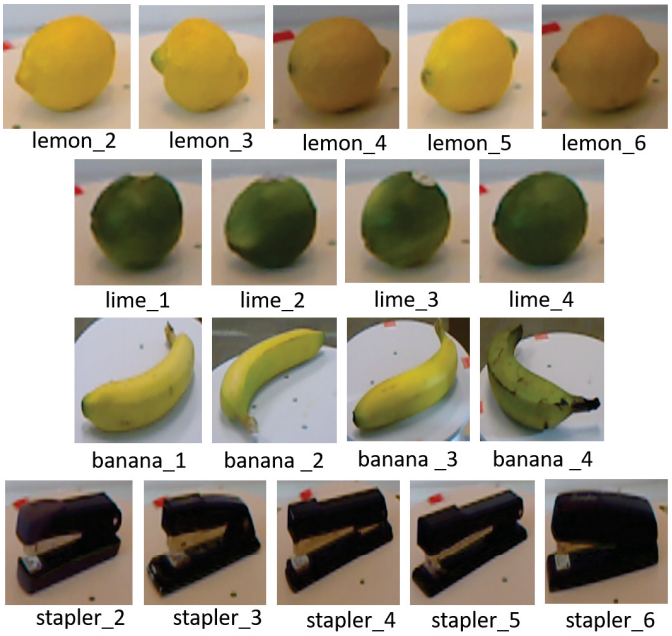


Figure 6.5: Examples of misclassified instances of our proposed method. We see that there is a high intra-class similarity in some categories of the dataset. Our method fails in discriminate such instances, as presented in Figure 6.4

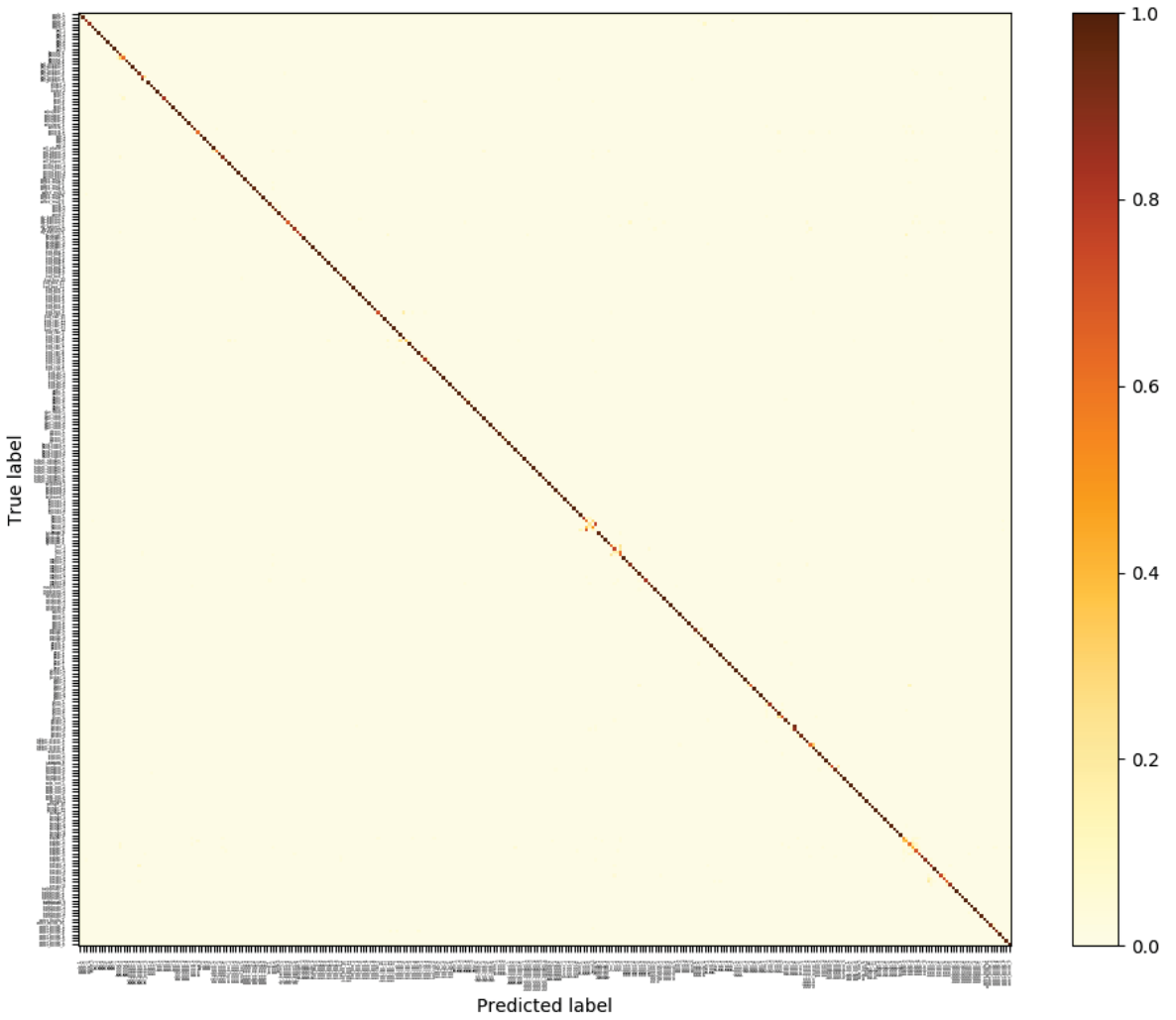


Figure 6.4: Confusion matrix of instance recognition of the proposed method. This matrix corresponds to the ResNet101-B7+LEAD method, that presented our best results on the leave-sequence-out scenario. For a high-quality image please visit: `http://bit.ly/cm-resnet101`

6.4 FINAL REMARKS AND OVERVIEW

In this chapter, we have presented a simple approach to combining off-the-shelf pre-trained models. We apply the weights of state-of-the-art architectures proposed and trained for the ImageNet dataset. We then merge them with the global versions of unsupervised descriptors for point cloud applications. The results show that this approach, despite the simplicity, is efficient principally in instance recognition applications, achieving over 95% accuracy for such task. Shape features have shown a low performance but do not compromise the efficiencies, on instance recognition, inclusively, their joint use has improved category recognition accuracy, from 1.4 to 3.1%. Following most state-of-the-art studies, maybe working directly on the depth information could bring more useful results, instead of processing point clouds, as our proposal. Another hint could be to use ensembles of local descriptors as performed by Lai et al. (2011a) with Spin Images.

The results we faced in our trials give us support to exploit more applied approaches in real-world environments. The next section offers a way to level up the comprehension of using pre-trained models in real-time (or nearly) object recognition applications.

7 TOWARDS REAL-TIME OBJECT RECOGNITION AND POSE ESTIMATION IN POINT CLOUDS

Object recognition and pose estimation represent a central role in a broad spectrum of applications, such as object grasping and manipulation, bin picking tasks, and verification of industrial assemblies (Wang et al., 2013; Vock et al., 2019). Successful object recognition, highly reliable pose estimation, and near real-time operation are essential capabilities and operational challenges for robot perception systems.

A methodology usually employed to estimate rigid transformations between scenes and objects is centered on a template matching approach. Starting from a known item or a part of an object, this technique involves searching all the occurrences in a larger, and usually, cluttered scene (Vock et al., 2019). However, due to natural occlusions, such occurrences may be represented only by a partial view of an object. The template is often another point cloud, and the main challenge of the template matching approach is to maintain the runtime feasibility and preserve the robustness.

Template matching approaches rely on RANSAC-based feature-matching algorithms, following the pipeline previously presented in Figure 2.11. RANSAC has proven to be one of the most versatile and robust to a wide range of applications. Unfortunately, for large or dense point clouds, its runtime becomes a significant limitation in several of the example applications mentioned above (Vock et al., 2019).

Very recent works propose to work on RGB images to estimate the 3D pose of objects in real-time for particular kinds of items, such as shoes (Hou et al., 2020) and human poses (Silva et al., 2019). When we seek a 6DoF estimation pose, performing in real-time is a more challenging task. Hodan et al. (2018) present an extensive benchmark of full cloud object detection and pose estimation and found a runtime of about a second per test target on average.

In this chapter, we introduce a novel pipeline to deal with point cloud pose estimation on uncontrolled environments and cluttered scenes. Our proposed pipeline recognizes the object using color feature descriptors, crops the selected bounding-box, reduces the search surface of the scene point cloud, and finally estimates the object's pose in a traditional local feature-based approach.

7.1 BACKGROUND

As presented in Figure 2.12, a comprehensive registration process usually consists of two steps: coarse and fine registrations (Guo et al., 2014a). We can produce a coarse registration transformation by performing a manual alignment, motion tracking, or the most common, using the local feature-matching (Mian et al., 2006). Local feature-matching-based algorithms automatically obtain corresponding points from two or multiple point clouds, coarsely registering by minimizing the distance between them. These methods have been extensively studied and have confirmed to be compliant and computer efficient (Johnson and Hebert, 1999; Tombari et al., 2010; Guo et al., 2013b; Salti et al., 2014). After coarsely register the point clouds, a fine-registration algorithm is applied to refine the initial coarse registration iteratively. Examples of fine-registration algorithms include the ICP algorithm that perform point-to-point alignment (Besl and McKay, 1992), or point-to-plane (Chen and Medioni, 1992). These algorithms are suitable to perform matching between point clouds of isolated scenes (3D registration) or between a scene and a model (3D object recognition). This proposal adopted two approaches to generate

the initial alignment: a traditional feature-based RANSAC and the Fast Global Registration (FGR).

Initially introduced by Fischler and Bolles (1981), RANSAC is an iterative and very versatile algorithm. Several applications enjoy this characteristic. To cite a few, we have outlier detection, approximate functions in a noisy set of points, and of course, for estimating a projection of a group of points into another group, and thus, estimate this projection's transformation.

In a feature-based strategy, RANSAC selects n random points from the source in each iteration and their corresponding points on the target. The algorithm rejects false matches, computes a transformation, and validates on the entire point cloud. This process is executed until satisfying some criteria or the maximum number of iterations. RANSAC and also ICP have some drawbacks regarding the iterations and the NN search to find matches. FGR algorithm (Zhou et al., 2016) proposes a feature-based approach that saves processing time by optimizing a global objective. Results shown that this technique provides accuracy comparable to ICP at a lower computational cost.

Despite the possibility of adopting only a Fast registration, in an application scenario, as we have only partial views of the objects, we must select the best-suited view between them. The feature-matching-based approach outputs a coarse transformation between each model's view and the scene. So, we choose the view with the highest number of inlier correspondences and submit it to an ICP fine alignment.

7.2 PROPOSED APPROACH

In this section, we explain in detail our proposed approach. We start from an RGB image and its corresponding point cloud, generated from RGB and depth images. These inputs are submitted to our pipeline, composed of three main steps: color feature classification, feature-based registration, and fine adjustment. We depict our proposal in Figure 7.1 and present these steps in the next sections.

7.2.1 Color feature classification

Our proposal starts with the detection of the desired object and bounding box estimation of it. After this detection, we can crop and preprocess the image and finally submit to a deep-learning-based color feature extractor. The preprocessing step includes image resizing to adjust to the network input dimensions. The deep network architecture outputs a feature vector, used to predict the object's instance, by a pre-trained ML classifier.

In our trials, we explored the achievements of Chapter 6, and employed models of networks, pre-trained on ImageNet, which presented satisfactory results on the Washington RGB-D Object dataset, as previously pointed. We tested the networks most accurate on the trials, according to Table 6.2, to name: ResNet101 (He et al., 2016a), MobileNet v2 (Sandler et al., 2018), ResNeXt101 32×8d (Xie et al., 2017), and EfficientNet-B7 (Tan and Le, 2019). These networks input a 224×244 pixels image and output a 1000 bins feature vector. We employed the LR classifier, with two variants: a pre-trained on the Object dataset's full set, and a distinct model trained on the subset of objects annotated on the Scenes dataset. To verify the best accurate classifier, we do not perform object detection. Instead, we get the ground-truth bounding boxes, hence verifying for each ML system, which is the best feasible performance.

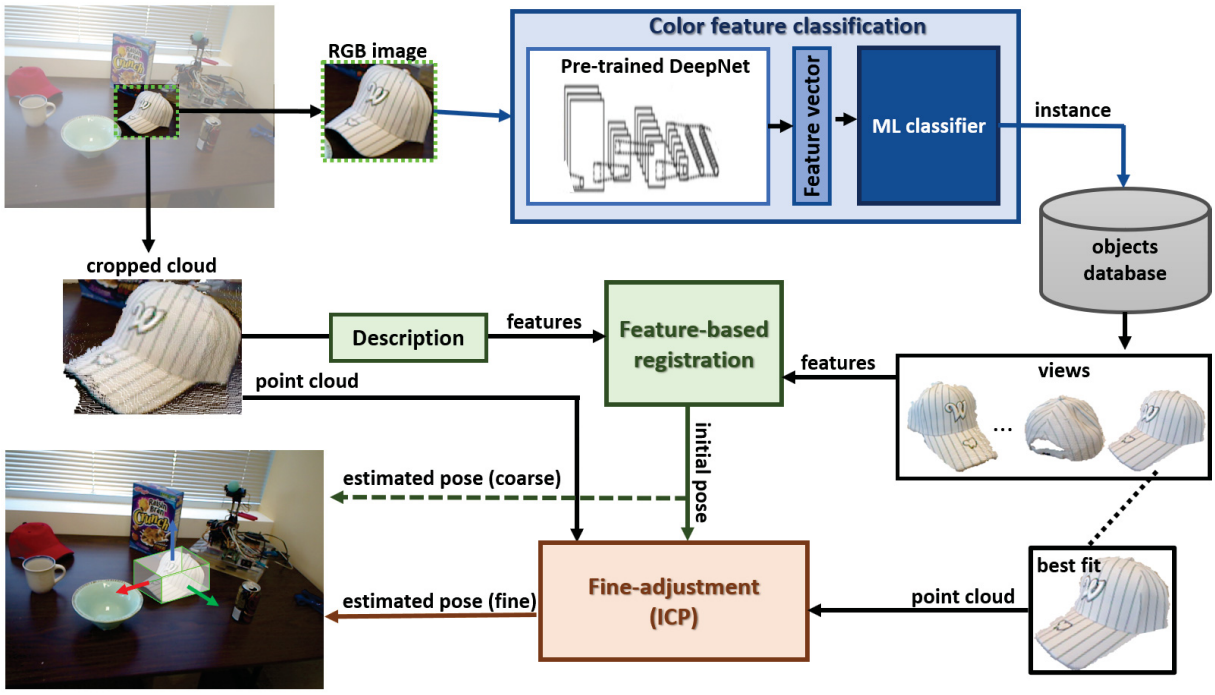


Figure 7.1: Pipeline of the proposed approach to pose estimation. To estimate the pre-segmented object's instance, we extract its features by a deep learning color-based extractor and a pre-trained ML classifier. After we select from the objects dataset, the view with the highest number of correspondences resulting from a feature-based registration algorithm. Finally, we apply an ICP dense registration algorithm to estimate the position and pose of the object.

7.2.2 Feature-based registration

Before scene processing, we extract and store information about the objects. The database is composed of information concerning each item, as well as the extracted features of them. We choose a local-descriptors-based approach to estimate the object's pose. For each instance of an object, we store several partial views of it. Between these views, our method will select the most likely to the correspondent object on the scene.

Based on the predicted objects' classes, we can select a set of described views from the model database. We then perform a feature-based registration between these views and the point cloud of the scene's object (previously cropped based on the detected bounding box). This method will estimate a transformation based on the correspondences between a scene and a partial view of an object. Then, the view with the highest number of inliers, and at least three correspondences is selected. The estimated affine transformation will input to the ICP algorithm, performing a dense registration.

We detect keypoint from each cloud with a uniform sampling, using a leaf size of 1 cm after we describe each keypoint using the FPFH (Rusu et al., 2009) descriptor with a radius of 5 cm. We choose this descriptor due to its processing time and size (33 bins) well-suited for real-time applications. To perform the coarse registration step, we evaluate two methods, previously presented: The RANSAC and FGR. We considered for both techniques an inlier correspondence distance lower than 1 cm between scene and models. We set the convergence criteria for RANSAC to 4M iterations and 500 validation steps, and for FGR to 100 iterations, following Choi et al. (2015) and Zhou et al. (2016).

7.2.3 Fine-adjustment

The previous step outputs an affine transformation that could be used as a final pose of the object concerning the scene. However, to guarantee a fine-adjustment, we employ an additional step to the process. We adopt the ICP algorithm to perform a dense registration, using as input the transformation resultant from the registration step, the scene, and best-fitted view clouds. We adopt an ICP based on the Point-to-plane approach (Chen and Medioni, 1992), with a max correspondence distance set to 1 cm.

7.3 EXPERIMENTAL RESULTS

7.3.1 Evaluation Protocol

We evaluate our proposal quantitative and qualitatively on the Washington RGB-D Scenes (Lai et al., 2011a), a challenging dataset, as pointed in Chapter 5 in our tests. First, we consider CNN feature extraction and classification accuracy based on the models trained in Chapter 6. We also verify the entire dataset's processing time, looking at the frame processing rate in classification and pose estimation scenarios.

The Scenes dataset does not provide ground-truth annotations concerning the objects' pose, so we had to find a plausible metric to evaluate the registration results. We adopted two different metrics: the Root mean squared error (RMSE) and an inlier ratio measurement. The latter represents the overlapping area between the source (model) and the target (scene). It is calculated based on the ratio between inlier correspondences and the number of points on the target.

7.3.2 Implementation details

Tests were performed on a Linux Ubuntu 18.04 LTS machine, equipped with a CPU Ryzen 7 2700X, 32GB of RAM, and a GPU Geforce RTX 2070 Super. To process the point clouds, perform keypoint extraction, description with FPFH, and registration with RANSAC and FGR, we used the Open3D Library, Version 0.4.0. We preprocess images using pillow 5.3.0 and OpenCV 3.4.2. The deep learning models implementations are from PyTorch 1.6.0, and the pre-trained networks are from Torchvision 0.7.0. To run the tests, we used GPU processing, powered by Cuda 10.1 and CudNN 7.6.3 libraries. The ML classifiers employed are from Scikit-learn 0.23.0.

7.3.3 Dataset

We validate our proposal on the Washington RGB-D Scenes, which has proved to be such a challenging dataset, as observed in Chapter 5 employing the proposed boosted pipeline. This dataset presents eight indoor sequences of scenes in household environments. The objects placed in such locations are related to the Washington RGB-D Objects dataset (Lai et al., 2011a), with categories and instances annotated by a bidimensional bounding box circumscribing each item. Table 7.1 gives some details regarding the size of the sequences and their average number of objects.

Table 7.1: Details regarding the RGB-D Scenes datasets

Scene	**Number of frames**	**Models per frame**
desk_1	98	1.89
desk_2	190	1.85
desk_3	228	2.56
kitchen_small_1	180	3.55
meeting_small_1	180	8.79
table_1	125	5.92
table_small_1	199	3.68
table_small_2	234	2.89
Average	**179.25**	**3.89**

7.3.4 Results

We summarize the experimental evaluation results on the Washington RGB-D Scenes in terms of accuracy and processing time. We opposed the selected CNN architectures examining only a classification based on the RGB information, taking the annotated bounding box, and submitting to the *Color Feature Classification* stage of our pipeline (as in Section 7.2.1). Table 7.2 relates to category and Table 7.3 to an instance-level recognition.

Table 7.2: Category classification performance on the RGB-D Scenes datasets.

	MobileNet v2		**Resnet101**		**ResNeXt101 32x8d**		**EfficientNet-B7**	
Scene	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS
desk_1	85.95%	**17.80**	85.41%	13.53	**94.05%**	10.35	90.27%	7.74
desk_2	51.42%	**18.86**	56.25%	13.90	64.77%	10.72	**82.67%**	7.82
desk_3	92.12%	**14.73**	79.79%	10.93	63.70%	7.90	**97.09%**	5.65
kitchen_small_1	53.68%	**10.54**	59.94%	7.90	59.62%	5.76	**78.25%**	4.26
meeting_small_1	59.92%	**4.68**	53.41%	3.38	49.75%	2.42	**65.87%**	1.75
table_1	71.35%	**6.76**	61.22%	5.02	53.38%	3.50	**73.11%**	2.51
table_small_1	**91.95%**	**10.20**	84.45%	7.77	67.39%	5.46	79.13%	4.10
table_small_2	63.37%	**12.37**	72.38%	9.82	56.72%	6.82	**77.10%**	5.16
Average	71.22%	**9.85**	69.11%	7.34	63.67%	5.27	**80.44%**	3.85

Table 7.3: Instance classification performance on the RGB-D Scenes datasets.

	MobileNet v2		**Resnet101**		**ResNeXt101 32x8d**		**EfficientNet-B7**	
Scene	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS
desk_1	42.70%	**13.03**	**51.89%**	9.66	48.11%	7.63	49.73%	6.55
desk_2	41.76%	**12.95**	38.92%	9.31	55.40%	7.93	**76.42%**	6.35
desk_3	72.77%	**9.84**	52.57%	7.09	52.91%	5.78	**90.58%**	4.60
kitchen_small_1	36.31%	**7.97**	34.74%	5.29	48.20%	4.12	**56.81%**	3.25
meeting_small_1	41.40%	**3.29**	38.05%	2.35	42.92%	1.74	**50.63%**	1.33
table_1	56.76%	**4.62**	38.11%	3.43	31.08%	2.49	**61.49%**	2.00
table_small_1	75.03%	**7.50**	63.30%	5.33	65.35%	3.89	**83.36%**	3.16
table_small_2	55.39%	**9.13**	45.35%	6.88	49.34%	5.04	**65.88%**	4.10
Average	52.77%	**6.99**	45.37%	5.03	49.16%	3.80	**66.86%**	3.02

The first outcome of this evaluation is the dominance of two networks over the other competitors considering different aspects. EfficientNet (Tan and Le, 2019) architecture

outperforms in terms of accuracy, and MobileNet v2 (Sandler et al., 2018) in processing time w.r.t. the others in almost all scenes, for category and instance.

Considering a category recognition scenario, EfficientNet reaches an average accuracy of more than 80%, followed by MobileNet v2. For instance recognition, the performance is almost 67%. However, when we aim a processing time efficiency, EfficientNet does not perform so well, being the slowest network with a frame-rate of 3.85 per second. On the other hand, the MobileNet v2 fulfills the network's main proposal to be time-efficient and light for embedded applications, and present the second-best accuracy and the best frame-rate, with almost 10 FPS for the category and 7 FPS for instance.

The full-set of the Object dataset contains 51 categories and 300 distinct instances. Concerning the Scenes dataset, this number drops to 6 categories and 22 instances. When we use a model trained on the full-set, most categories or instances will never be detected. Thus, we learned a lighter classifier that considers only achievable classes/instances. We show such results in Table 7.4.

Table 7.4: Performance comparison between a full and a specific training set. Best result for each column in **bold**

DeepNet	Category				Instance			
	Full		Scenes		Full		Scenes	
	Acc	FPS	Acc	FPS	Acc	FPS	Acc	FPS
MobileNet v2	71.22%	**9.85**	90.65%	**25.02**	52.77%	**6.99**	67.35%	**24.62**
Resnet101	69.11%	7.34	88.82%	14.59	45.37%	5.03	61.41%	13.94
ResNeXt101 32x8d	63.67%	5.27	83.71%	9.20	49.16%	3.80	59.04%	8.86
EfficientNet-B7	**80.44%**	3.85	**92.79%**	6.17	**66.86%**	3.02	**82.94%**	5.88

After this change on the model specificity, we distinguish a noticeable improvement in accuracy and the processing time, achieving MobileNet v2 a near real-time performance regarding an average scenario. A significant accuracy gain was established, with over 12% for categories and 10% for instances, pulling the best results to respectively 92% and 83%.

Regarding frame processing rate, it is essential to notice that the average number of models varies from 1.85 to 8.79 over the scenes (Table 7.1), with almost four objects per frame on average. Thus, we can infer that our proposal can deliver a near-real-time FPS, inclusive in a multi-classification problem. When we consider only a single target, the performance is almost four times faster, as presented in Table 7.6, on the *Color only* column.

7.3.4.1 Pose estimation results

Based on the assumption that we mapped the objects we aim, and we could detect in a real-world scenario, we adopted those models trained on the subset of the RGB-D Object dataset. We also considered only an instance detection situation. The reason for working only on the instance-level is that we could have intra-class misclassifications, which could corrupt the pose alignment step. For each instance detected by the *color feature classification* stage, we take ten views of the referred object from the models' database.

In Table 7.5 we report an evaluation concerning the Feature-based registration and Fine-adjustment stages of our pipeline. Getting a set of ten views of the same object, we perform a coarse estimation by using RANSAC or FGR. We evaluate quantitatively such methods concerning the inlier ratio, RMSE, and execution time. We apply the resulting affine transformation as the input of an ICP dense registration and evaluate if this input can imply differences in the processing time.

Table 7.5: Comparison between feature-based registration methods. The times reported for the feature-based methods consider the whole execution for ten views of the same object and select the best one. Processing time listed in seconds. Best result for each column in **bold**

Methods	Feature-based registration time (↓)	ICP time (↓)	Inlier ratio (↑)	RMSE (↓)
RANSAC	0.7688	**0.0061**	**0.2689**	**0.0055**
Fast Global Registration	**0.0580**	0.0075	0.1895	0.0059

Indeed, the FGR method is much faster than RANSAC. However, we observe that for both metrics, RANSAC outperforms it. The Inlier ratio presented by the latter is around 50% higher than the faster method, and also show an RMSE more consistent. The transformation generated by the coarse alignment algorithm also impacts the ICP execution, and we see that a better estimation can speed up the fine-adjustment process. To evaluate more deeply if the ICP, after the feature-matching application, can surpass problems like a more rough estimation, we must assess an annotated pose. Unfortunately, the adopted dataset does not offer such data, and further studies may verify such affirmation on a pose-annotated dataset.

Now we report the processing rate regarding executing all the stages of our proposed pipeline. Table 7.6 presents the frame processing rate based on a single target object scenario. We evaluate referring to the first stage execution (*Color only*), the early two stages (Columns *RANSAC*, and *FGR*), and a pipeline' full execution (*+ICP*).

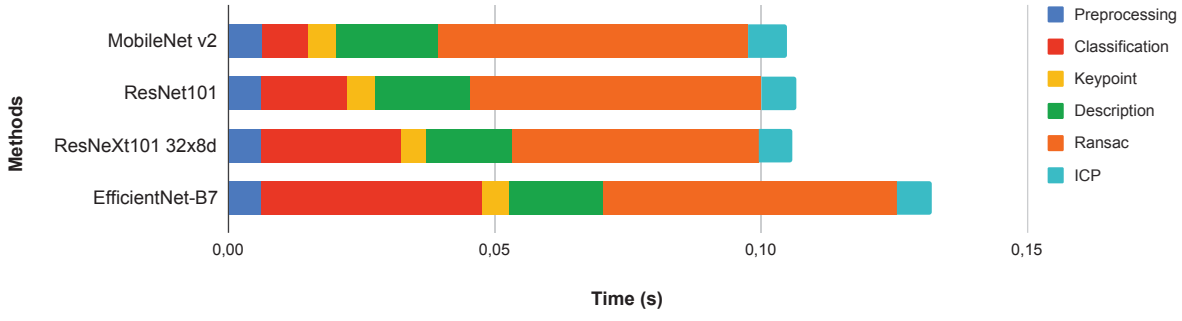
Table 7.6: Frame processing rate based on a single target pose estimation. *Color only* refers to object classification only, other columns refer to the pose aligment step, coarse (RANSAC and FGR) or fine (by adding ICP).

	Color only	RANSAC	FGR	RANSAC + ICP	FGR + ICP
MobileNet v2	89.49	1.89	13.89	1.82	13.57
ResNet101	52.45	1.96	13.83	1.81	13.39
ResNeXt101 32x8d	33.73	2.03	14.18	2.09	13.32
EfficientNet-B7	22.51	1.43	8.94	1.40	8.55

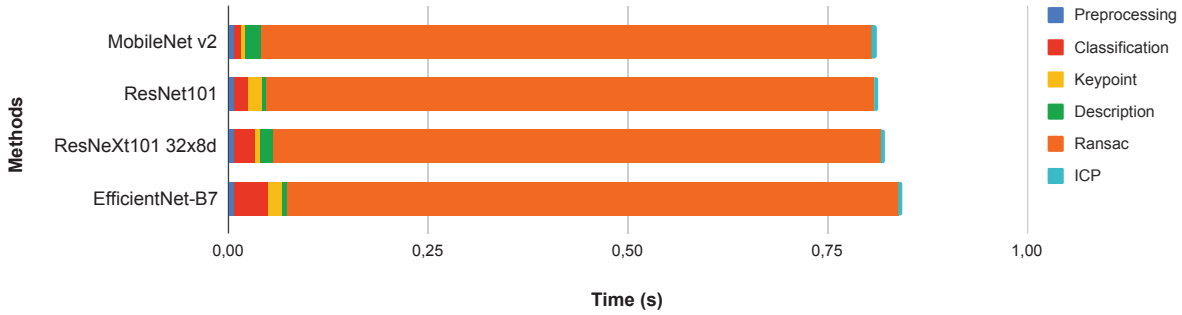
At first sight, one can conjecture that a RANSAC-based approach is unpromising when presenting around 2 FPS. However, considering an FGR-based process, the results are encouraging, with 8 FPS for the best accurate method, and more than 13 for the others. For many applications that deal with real-time, a frame rate around eight or more is acceptable. We agree that *the facto* standard for real-time is to process at least 30 FPS, however, due to the modularity of our proposed pipeline, the stages are independent, and we could let a full execution to indispensable situations.

An application situation may include a target object's location and recovery of the pose, for instance, by a robot. The system could execute a scheduled procedure, firstly, localizing this object by using only the first stage of the pipeline, in real-time. Then, as the subject approaches the objective, we could execute the second stage, estimating a rough transformation, e.g., once a second. Finally, when the object is next to the user, we can run the full pipeline, with the fine-adjustment stage.

To investigate more deeply the processing time of a successfully detected object of our pipeline, we summarize how much time takes each substep in Figure 7.2. We can infer two main steps that negatively impact the time processing: classification and feature-based estimation. Regarding the former, the correct selection of the network to extract color features is fundamental to speed-up the whole process, presenting a significant difference between the faster, i.e., MobileNet v2 (Sandler et al., 2018), and the slower, i.e., EfficientNet-B7 (Tan and Le,



(a) FGR



(b) RANSAC

Figure 7.2: Processing time of each step on the proposed approach. We consider only succesfully detected objects on this comparison. (a) presents times referring to the Fast Global Registration (Zhou et al., 2016) method, and (b) to RANSAC. Times are given in seconds

2019). We perceive a considerable impact on the time processing when using RANSAC for the feature-based stage, despite having better results than FGR. In this implementation, we do not use any concurrent processing, which could significantly improve such time for both coarse pose estimation methods. Our pipeline is highly flexible, and the use of recent proposals may enhance our results, for instance, Deep Global Registration (Choy et al., 2020) to coarse estimation, and the ColoredICP (Park et al., 2017) to fine-adjustment.

7.3.4.2 Qualitative results

We provide qualitative visualizations of our proposed method (RANSAC + ICP) regarding pose estimation in Figure 7.3. Our method succeeds in aligning several different shaped models, such as planes (*cereal box*), cylinders (*soda can*, *coffee mugs*, and *flashlights*), and free form models (*caps*). As we perform a rigid transformation to align objects and scenes, it is fundamental to the model choice. Examples like the *red cap* that present a crumple on top, harming the alignment estimation. Otherwise, we confirm the robustness of the combination of coarse and fine alignments on the bowl object, that despite being partially cropped on the scene cloud, still has inferred the pose correctly.

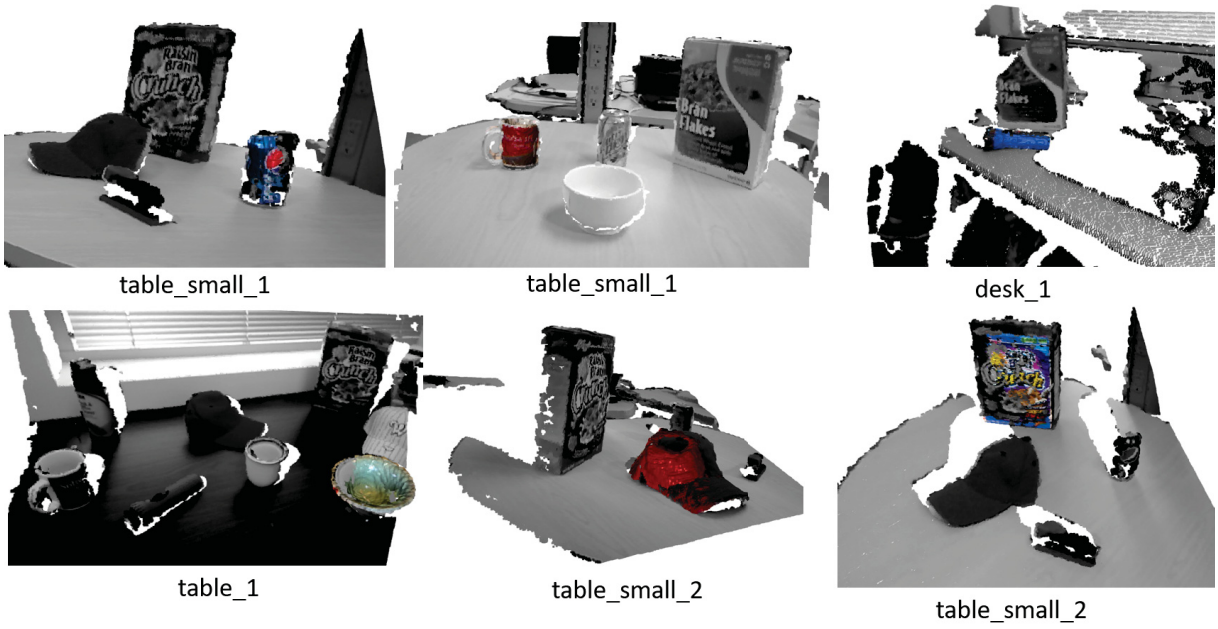


Figure 7.3: Qualitative visualizations of successful pose alignment

In Figure 7.4, we present some wrong alignments of our proposals. We can observe that the main shape of the object weights a lot on the alignment. For instance, the *mugs* had the body (cylinder) well aligned but a misalignment on the handle. We also perceive a flip on the cereal box due to the large plane at the front. The *bowl* in the rightmost example fails in aligning, though, differently from the previous figure, where the method robustly handled a partial view of a *bowl*, in this case, we have about 50% only of the object visible. Despite the misalignments verified, as we reduce the surface search on the scene cloud, we always have an estimation next or even inside the 3D projection of the 2D bounding box outputted by the detection stage.

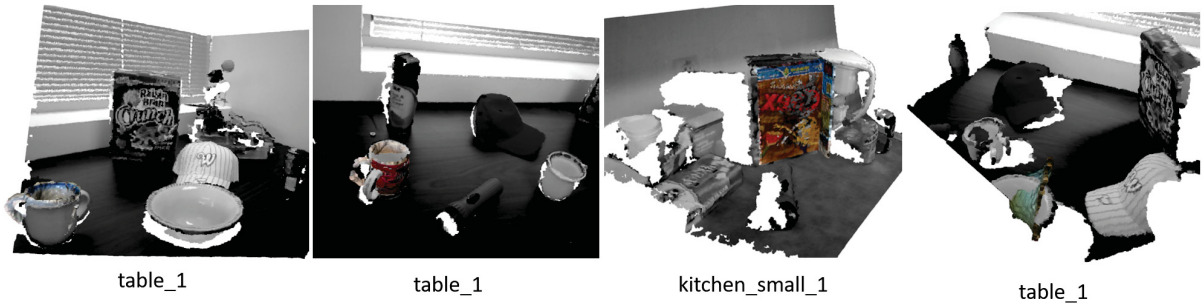


Figure 7.4: Qualitative visualizations of wrong pose alignment. From left to right: two examples of coffee mugs with a misoriented handles, flipped cereal box, and a rotated bowl

7.4 FINAL REMARKS AND OVERVIEW

3D pose estimation is a challenging task, mainly for real-time applications. Sometimes developers must surrender on the precision, aiming response time. This chapter introduced a novel pipeline that combines the power of deep-learning-based color features extractors with a local descriptors pipeline to pose estimation in point clouds. We evaluated detection of objects and achieved almost 93% accuracy on a category recognition scenario, and 83% on an instance situation, in the best case. This precision is also accompanied by a high frame processing rate, arriving up to

90 FPS. The pose estimation rate is plausible for some applications, and by scheduling the stages of our pipeline, we can reach standard real-time processing.

Parallelization strategies can improve even more time results, and also, different local descriptors and keypoint extractors could support this. Further studies include evaluating our proposal in a pose-annotated dataset to verify the estimation precision. Additionally, these findings on the deep-learning-based architectures can help develop an integrated region proposal and object detection algorithm based on them. State-of-the-art deep learning methods such as SSD (Liu et al., 2016) and YOLO (Redmon et al., 2016) enable such potentiality.

8 CONCLUSIONS AND FUTURE DIRECTIONS

This work's main objective was to propose, evaluate, and validate strategies to optimize feature-based applications on 3D point clouds. Our main contributions are two-folded and include the development of local descriptors based on the deep learning paradigm and the proposal of improvements' strategies of existing object recognition and pose estimation standards in 3D-based applications.

We proposed some novel findings on this doctoral thesis. We presented an efficient equivariant local descriptor, named LEAD, that presents state-of-the-art accuracy performance on standard benchmarks. It is the best method by far when considering a transfer learning scenario. At the best of our knowledge, we also introduced the first fully-data-driven approach to extract orientation from 3D patches, an LRF named Compass. We compared our proposed LRF with state-of-the-art hand-crafted methods, and the results show the supremacy of our proposal over the competitors in three different standard datasets. We also validate Compass in a full-object orientation condition using it as a transformation network attached to a Pointnet in classification mode. The results again show the robustness of this LRF in distinct evaluation plots.

Based on the findings of LEAD and Compass, we also bestowed SOUND, the first end-to-end self-orienting 3D local descriptor learned strictly from data. SOUND network, leveraged by the Spherical CNNs framework, extracts discriminative features and orientation from 3D patches in a single forward pass. Evaluation results have proven the proposal's efficiency regarding relative rotation and translation errors, presenting very competing results in the studied datasets.

On the object recognition and pose estimation scenarios, we started developing an update on the standard pipeline proposed by Aldoma et al. (2012b). We found that, by adding a simple prior segmentation step, the whole pipeline processing time, and surprisingly the overall accuracy, are significantly improved. This previous step, called Saliency Boost, utilizes a saliency object detection method. We reach an accuracy gain of up to 60% and a processing time five times faster through this modified pipeline. Continuing in the same line of combining visual and shape features, we assessed deep-architecture-based feature extractors on the object recognition task. The discriminative capacity of produced color features pulls such networks in state-of-the-art regarding instance recognition. We also verified a noteworthy improvement on category recognition, when color and shape features are combined, concerning color-only approaches.

Despite the improvements in the proposed boosted pipeline, we verified an underperformance in real-world scenes environments. Such results incited us to propose an even more valuable methodology to detect and estimate objects' 6DoF pose. Results from the comprehensive evaluation of color feature extractors also supported our choices. Our proposed pipeline uses a 2D object recognition stage, then collects a set of partial views (2.5D) of objects, submits to a coarse alignment, and finally fine-estimates the object's pose using an ICP dense registration stage. We found that this pipeline is suitable for real-time applications, as discussed, and competitive with prior literature results regarding the processing frame rate.

8.1 SUMMARY OF CONTRIBUTIONS

This section summarizes the contributions considered in the introduction of this doctoral thesis. At this point, we realize the contributions of the proposed approaches we made throughout this document.

- Our LEAD descriptor presents a progression of the previous work of Spezialetti et al. (2019), regarding accuracy in terms of feature match recall (1.64%) and in time performance (37%).
- LEAD features as runner-up in the 3DMatch Benchmark dataset and performs in the first place on the challenging ETH dataset, with a significant margin of more than 17% about the other methods.
- We introduced Compass, the first full-data-driven LRF. Despite the novelty, our proposal outperforms the existing hand-crafted techniques in three different datasets in terms of repeatability, which is the standard metric for such judgment.
- Feeding a PointNet architecture with a canonical orientation provided by Compass, we achieve state-of-the-art performance on a rotation-invariant shape classification problem.
- We present, at the best of our knowledge, the first self-orienting local descriptors, named SOUND. On direct measurement errors, SOUND outperforms the other compared techniques presenting significative reductions regarding RRE and RTE.
- About object recognition tasks, we propose an update on the standard local descriptors pipeline, named Saliency Boost. This method not only speeds up the whole process by almost $5\times$ but confer an accuracy gain of the tested methods.
- We performed a comprehensive evaluation of established state-of-the-art networks, evaluating models trained on the ImageNet and transferred to the RGB-D object recognition's context.
- A combination of color and shape features was proposed, and despite the simplicity, presented convincing results in category and instance recognition tasks.
- We presented and evaluated an effective pipeline for object detection and 6DoF pose estimation that enables real-time processing for point clouds.

8.2 FUTURE WORKS

This dissertation enabled the identification of some future directions based on our observations. This section presents and discusses some efforts that could bring valuable results, as emerging studies of those granted on this document.

- Based on our thesis' findings, one future direction could be toward the proposition of an end-to-end pose estimation network, by using a SOUND-like architecture for RGB-D. A similar preceding work of the Frustum Networks, presented by Qi et al. (2018), relies on the estimation of amodal bounding cubes from lidar-like data, not properly 6DoF estimation.

- The Saliency Boost pipeline results encourage an investigation of the reliance on applying a similar approach to the registration pipeline.
- We explored the combination of color and shape features, considering the global objects' embedding. One future direction could test the use of local descriptors' ensembles associated with color features, as performed by Lai et al. (2011a) with Spin Images.
- Spherical CNNs (Cohen et al., 2018) architecture presents limitations regarding processing time. The calculation of the discrete Fourier transform on the framework is quite heavy in terms of computational cost. Some newer initiatives can better handle computer resources, such as the Icosahedral CNNs (Cohen et al., 2019) and Spin-Weighted Spherical CNNs (Esteves et al., 2020) should be considered.

8.3 FINAL REMARKS

As stated in the objectives, this doctoral thesis has presented strategies to deal better with 3D information. As always in science, our achieved results are only the iceberg's tip, and much more could be done. We expect that our studies instigate and inspire other researches in working in 3D CV applications, a challenging but at the same time rewarding field.

REFERENCES

Agrawal, P., Girshick, R. and Malik, J. (2014). Analyzing the performance of multilayer neural networks for object recognition. Em *European conference on computer vision*, páginas 329–344. Springer.

Aldoma, A., Marton, Z., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B. and Vincze, M. (2012a). Three-dimensional object recognition and 6 DoF pose estimation. *IEEE Robotics & Automation Magazine*, páginas 80–91.

Aldoma, A., Marton, Z.-C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R. B., Gedikli, S. and Vincze, M. (2012b). Tutorial: Point cloud library: Three-dimensional object recognition and 6 DoF pose estimation. *IEEE Robotics & Automation Magazine*, 19(3):80–91.

Aldoma, A., Tombari, F., Rusu, R. B. and Vincze, M. (2012c). OUR-CVFH–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6DoF pose estimation. Em *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, páginas 113–122. Springer.

Alexandre, L. A. (2012). 3D descriptors for object and category recognition: a comparative evaluation. Em *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*, página 7.

Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D. and Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795.

Asif, U., Bennamoun, M. and Sohel, F. A. (2017). A multi-modal, discriminative and spatially invariant CNN for RGB-D object labeling. *IEEE transactions on pattern analysis and machine intelligence*, 40(9):2051–2065.

Aytekin, C., Iosifidis, A. and Gabbouj, M. (2018). Probabilistic saliency estimation. *Pattern Recognition*, 74:359–372.

Bai, J., Wu, Y., Zhang, J. and Chen, F. (2015). Subset based deep learning for RGB-D object recognition. *Neurocomputing*, 165:280–292.

Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L. and Tai, C.-L. (2020). D3Feat: Joint learning of dense detection and description of 3D local features. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, páginas 6359–6367.

Beksi, W. J. and Papanikolopoulos, N. (2015). Object classification using dictionary learning and RGB-D covariance descriptors. Em *2015 IEEE international conference on robotics and automation (ICRA)*, páginas 1880–1885. IEEE.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-D shapes. Em *Sensor fusion IV: control paradigms and data structures*, volume 1611, páginas 586–606. International Society for Optics and Photonics.

Blum, M., Springenberg, J. T., Wülfing, J. and Riedmiller, M. (2012). A learned feature descriptor for object recognition in RGB-D data. Em *2012 IEEE International Conference on Robotics and Automation*, páginas 1298–1303. IEEE.

Bo, L., Ren, X. and Fox, D. (2011). Depth kernel descriptors for object recognition. Em *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 821–826. IEEE.

Bo, L., Ren, X. and Fox, D. (2013). Unsupervised feature learning for RGB-D based object recognition. Em *Experimental robotics*, páginas 387–402. Springer.

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Em *Proceedings of the fifth annual workshop on Computational learning theory*, páginas 144–152.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

Bui, H. M., Lech, M., Cheng, E., Neville, K. and Burnett, I. S. (2016). Object recognition using deep convolutional features transformed by a recursive network structure. *IEEE Access*, 4:10059–10066.

Caglayan, A. and Burak Can, A. (2018). Exploiting multi-layer features using a CNN-RNN approach for RGB-D object recognition. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, páginas 0–0.

Caglayan, A., Imamoglu, N., Can, A. B. and Nakamura, R. (2020). When CNNs meet random RNNs: Towards multi-level analysis for RGB-D object and scene recognition. *arXiv preprint arXiv:2004.12349*.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H. et al. (2015). Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*.

Chapelle, O. and Wu, M. (2010). Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval*, 13(3):216–235.

Chen, H. and Bhanu, B. (2007). 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262.

Chen, H., Li, Y. and Su, D. (2019). Multi-modal fusion network with multi-scale multi-path and cross-modal interactions for RGB-D salient object detection. *Pattern Recognition*, 86:376–385.

Chen, X., Ma, H., Wan, J., Li, B. and Xia, T. (2017). Multi-view 3D object detection network for autonomous driving. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1907–1915.

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155.

Choi, S., Zhou, Q.-Y. and Koltun, V. (2015). Robust reconstruction of indoor scenes. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 5556–5565.

Chopra, S., Hadsell, R. and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. Em *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, páginas 539–546. IEEE.

Choy, C., Dong, W. and Koltun, V. (2020). Deep global registration. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, páginas 2514–2523.

Choy, C., Gwak, J. and Savarese, S. (2019a). 4D spatio-temporal convnets: Minkowski convolutional neural networks. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 3075–3084.

Choy, C., Park, J. and Koltun, V. (2019b). Fully convolutional geometric features. Em *ICCV*.

Chua, C. S. and Jarvis, R. (1997). Point signatures: A new representation for 3D object recognition. *International Journal of Computer Vision*, 25(1):63–85.

Cohen, T., Weiler, M., Kicanaoglu, B. and Welling, M. (2019). Gauge equivariant convolutional networks and the icosahedral CNN. Em Chaudhuri, K. and Salakhutdinov, R., editores, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 de *Proceedings of Machine Learning Research*, páginas 1321–1330, Long Beach, California, USA. PMLR.

Cohen, T. S., Geiger, M., Köhler, J. and Welling, M. (2018). Spherical CNNs. Em *International Conference on Learning Representations*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. Em *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, páginas 303–312.

Dai, A., Nießner, M., Zollöfer, M., Izadi, S. and Theobalt, C. (2017). BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics 2017 (TOG)*.

Deng, H., Birdal, T. and Ilic, S. (2018a). PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, páginas 602–618.

Deng, H., Birdal, T. and Ilic, S. (2018b). PPFNet: Global context aware local features for robust 3D point matching. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 195–205.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. Em *2009 IEEE conference on computer vision and pattern recognition*, páginas 248–255. Ieee.

Driscoll, J. R. and Healy, D. M. (1994). Computing Fourier transforms and convolutions on the 2-sphere. *Advances in applied mathematics*, 15(2):202–250.

Drost, B., Ulrich, M., Navab, N. and Ilic, S. (2010). Model globally, match locally: Efficient and robust 3D object recognition. Em *2010 IEEE computer society conference on computer vision and pattern recognition*, páginas 998–1005. Ieee.

Elbaz, G., Avraham, T. and Fischer, A. (2017). 3D point cloud registration for localization using a deep neural network auto-encoder. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 4631–4640.

Esteves, C., Allen-Blanchette, C., Makadia, A. and Daniilidis, K. (2018). Learning SO(3) equivariant representations with spherical CNNs. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, páginas 52–68.

Esteves, C., Makadia, A. and Daniilidis, K. (2020). Spin-weighted spherical CNNs. *arXiv preprint arXiv:2006.10731*.

Fehr, D., Beksi, W. J., Zermas, D. and Papanikolopoulos, N. (2014). RGB-D object classification using covariance descriptors. Em *2014 IEEE International Conference on Robotics and Automation (ICRA)*, páginas 5467–5472. IEEE.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Gojcic, Z., Zhou, C., Wegner, J. D. and Wieser, A. (2019). The perfect match: 3D point cloud matching with smoothed densities. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 5545–5554.

Gomes, R. B., da Silva, B. M. F., de Medeiros Rocha, L. K., Aroca, R. V., Velho, L. C. P. R. and Gonçalves, L. M. G. (2013). Efficient 3D object recognition using foveated point clouds. *Computers & Graphics*, 37(5):496–508.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep learning*. MIT press.

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C. and Aubry, M. (2018). A papier-mâché approach to learning 3D surface generation. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 216–224.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M. and Wan, J. (2014a). An integrated framework for 3-D modeling, object detection, and pose estimation from point-clouds. *IEEE Transactions on Instrumentation and Measurement*, 64(3):683–693.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J. and Kwok, N. M. (2016). A comprehensive performance evaluation of 3D local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J. and Zhang, J. (2014b). Performance evaluation of 3D local feature descriptors. Em *Asian Conference on Computer Vision*, páginas 178–194. Springer.

Guo, Y., Sohel, F., Bennamoun, M., Lu, M. and Wan, J. (2013a). Rotational projection statistics for 3D local surface description and object recognition. *International journal of computer vision*, 105(1):63–86.

Guo, Y., Sohel, F. A., Bennamoun, M., Wan, J. and Lu, M. (2013b). RoPS: A local feature descriptor for 3D rigid objects based on rotational projection statistics. Em *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, páginas 1–6. IEEE.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. and Bennamoun, M. (2020). Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hartley, R., Trumpf, J., Dai, Y. and Li, H. (2013). Rotation averaging. *International journal of computer vision*, 103(3):267–305.

He, K., Zhang, X., Ren, S. and Sun, J. (2016a). Deep residual learning for image recognition. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 770–778.

He, K., Zhang, X., Ren, S. and Sun, J. (2016b). Identity mappings in deep residual networks. Em *European conference on computer vision*, páginas 630–645. Springer.

Hermans, A., Beyer, L. and Leibe, B. (2017). In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

Hodan, T., Michel, F., Brachmann, E., Kehl, W., GlentBuch, A., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X. et al. (2018). Bop: Benchmark for 6D object pose estimation. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, páginas 19–34.

Holz, D., Ichim, A. E., Tombari, F., Rusu, R. B. and Behnke, S. (2015). Registration with the point cloud library: A modular framework for aligning in 3-D. *IEEE Robotics & Automation Magazine*, 22(4):110–124.

Honari, S., Molchanov, P., Tyree, S., Vincent, P., Pal, C. and Kautz, J. (2018). Improving landmark localization with semi-supervised learning. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1546–1555.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W. (1992). Surface reconstruction from unorganized points. Em *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, páginas 71–78.

Hornik, K., Stinchcombe, M., White, H. et al. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hou, Q., Cheng, M.-M., Hu, X., Borji, A., Tu, Z. and Torr, P. H. (2017). Deeply supervised salient object detection with short connections. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 3203–3212.

Hou, T., Ahmadyan, A., Zhang, L., Wei, J. and Grundmann, M. (2020). Mobilepose: Real-time pose estimation for unseen objects with weak shape supervision. *arXiv preprint arXiv:2003.03522*.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. et al. (2019). Searching for mobilenetv3. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 1314–1324.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Huh, M., Agrawal, P. and Efros, A. A. (2016). What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*.

Huynh, D. Q. (2009). Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164.

Jansen, P. and Kellner, J. (2015). The role of rotational hand movements and general motor ability in children's mental rotation performance. *Frontiers in Psychology*, 6:984.

Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449.

Kadambi, A., Bhandari, A. and Raskar, R. (2014). 3D depth cameras in vision: Benefits and limitations of the hardware. Em *Computer Vision and Machine Learning with RGB-D Sensors*, páginas 3–26. Springer.

Kastner, S. and Ungerleider, L. G. (2000). Mechanisms of visual attention in the human cortex. *Annual review of neuroscience*, 23(1):315–341.

Khoury, M., Zhou, Q.-Y. and Koltun, V. (2017). Learning compact geometric features. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 153–161.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klokov, R. and Lempitsky, V. (2017). Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 863–872.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em *Advances in neural information processing systems*, páginas 1097–1105.

Lai, K., Bo, L. and Fox, D. (2014). Unsupervised feature learning for 3D scene labeling. Em *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, páginas 3050–3057. IEEE.

Lai, K., Bo, L., Ren, X. and Fox, D. (2011a). A large-scale hierarchical multi-view RGB-D object dataset. Em *2011 IEEE international conference on robotics and automation*, páginas 1817–1824. IEEE.

Lai, K., Bo, L., Ren, X. and Fox, D. (2011b). Sparse distance learning for object recognition combining rgb and depth information. Em *2011 IEEE International Conference on Robotics and Automation*, páginas 4007–4013. IEEE.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Leo, M., Furnari, A., Medioni, G. G., Trivedi, M. and Farinella, G. M. (2018). Deep learning for assistive computer vision. Em *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

Li, J., Chen, B. M. and Hee Lee, G. (2018a). So-net: Self-organizing network for point cloud analysis. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 9397–9406.

Li, L., Zhu, S., Fu, H., Tan, P. and Tai, C.-L. (2020a). End-to-end learning local multi-view descriptors for 3D point clouds. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, páginas 1919–1928.

Li, Y., Mo, K., Shao, L., Sung, M. and Guibas, L. (2020b). Learning 3d part assembly from a single image. *European conference on computer vision (ECCV 2020)*.

Li, Z., Lang, C., Feng, S. and Wang, T. (2018b). Saliency ranker: A new salient object detection method. *Journal of Visual Communication and Image Representation*, 50:16–26.

Liu, H., Li, F., Xu, X. and Sun, F. (2018). Multi-modal local receptive field extreme learning machine for object recognition. *Neurocomputing*, 277:4–11.

Liu, J.-J., Hou, Q., Cheng, M.-M., Feng, J. and Jiang, J. (2019a). A simple pooling-based design for real-time salient object detection. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 3917–3926.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016). SSD: Single shot multibox detector. Em *European conference on computer vision*, páginas 21–37. Springer.

Liu, X., Han, Z., Liu, Y.-S. and Zwicker, M. (2019b). Point2sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network. Em *Thirty-Third AAAI Conference on Artificial Intelligence*.

Loghmani, M. R., Planamente, M., Caputo, B. and Vincze, M. (2019). Recurrent convolutional fusion for RGB-D object recognition. *IEEE Robotics and Automation Letters*, 4(3):2878–2885.

Long, J., Shelhamer, E. and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 3431–3440.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. Em *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, páginas 1150–1157. Ieee.

Luo, X., Huang, J., Szeliski, R., Matzen, K. and Kopf, J. (2020). Consistent video depth estimation. *Tranactions on Graphics*, 39(4).

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A. and van der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. Em *Proceedings of the European Conference on Computer Vision (ECCV)*, páginas 181–196.

Mahendran, S., Ali, H. and Vidal, R. (2017). 3D pose regression using convolutional neural networks. Em *Proceedings of the IEEE International Conference on Computer Vision Workshops*, páginas 2174–2182.

Manhardt, F., Arroyo, D. M., Rupprecht, C., Busam, B., Birdal, T., Navab, N. and Tombari, F. (2019). Explaining the ambiguity of object detection and 6D pose from visual data. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 6841–6850.

Marcon, M., Spezialetti, R., Salti, S., Silva, L. and Di Stefano, L. (2019). Boosting object recognition in point clouds by saliency detection. Em *International Conference on Image Analysis and Processing*, páginas 321–331. Springer.

Masci, J., Boscaini, D., Bronstein, M. and Vandergheynst, P. (2015). Geodesic convolutional neural networks on riemannian manifolds. Em *Proceedings of the IEEE international conference on computer vision workshops*, páginas 37–45.

Maturana, D. and Scherer, S. (2015). Voxnet: A 3D convolutional neural network for real-time object recognition. Em *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, páginas 922–928. IEEE.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Melzi, S., Spezialetti, R., Tombari, F., Bronstein, M. M., Di Stefano, L. and Rodola, E. (2019). Gframes: Gradient-based local reference frame for 3D shape matching. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 4629–4638.

Mian, A., Bennamoun, M. and Owens, R. (2010). On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361.

Mian, A. S., Bennamoun, M. and Owens, R. A. (2006). A novel representation and feature matching algorithm for automatic pairwise registration of range images. *International Journal of Computer Vision*, 66(1):19–40.

Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2.

Murali, A., Mousavian, A., Eppner, C., Paxton, C. and Fox, D. (2020). 6-dof grasping for target-driven object manipulation in clutter. Em *2020 IEEE International Conference on Robotics and Automation (ICRA)*, páginas 6232–6238. IEEE.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Ng, A. (2020). Deep learning. `http://cs229.stanford.edu/materials/CS229-DeepLearning.pdf`. Acessed in: 30-7-2020.

Novatnack, J. and Nishino, K. (2008). Scale-dependent/invariant local 3D shape descriptors for fully automatic registration of multiple sets of range images. Em *European conference on computer vision*, páginas 440–453. Springer.

Pan, H., Guan, T., Luo, Y., Duan, L., Tian, Y., Yi, L., Zhao, Y. and Yu, J. (2016). Dense 3D reconstruction combining depth and RGB information. *Neurocomputing*, 175:644–651.

Park, J., Zhou, Q.-Y. and Koltun, V. (2017). Colored point cloud registration revisited. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 143–152.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Petrelli, A. and Di Stefano, L. (2011). On the repeatability of the local reference frame for partial shape matching. Em *2011 International Conference on Computer Vision*, páginas 2244–2251. IEEE.

Petrelli, A. and Di Stefano, L. (2012). A repeatable and efficient canonical reference for surface matching. Em *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, páginas 403–410. IEEE.

Pomerleau, F., Liu, M., Colas, F. and Siegwart, R. (2012). Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14):1705–1711.

Qi, C. R., Liu, W., Wu, C., Su, H. and Guibas, L. J. (2018). Frustum pointnets for 3D object detection from RGB-D data. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 918–927.

Qi, C. R., Su, H., Mo, K. and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3D classification and segmentation. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 652–660.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M. and Guibas, L. J. (2016). Volumetric and multi-view CNNs for object classification on 3D data. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 5648–5656.

Qi, C. R., Yi, L., Su, H. and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Em *Advances in neural information processing systems*, páginas 5099–5108.

Rao, Y., Lu, J. and Zhou, J. (2019). Spherical fractal convolutional neural networks for point cloud recognition. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 452–460.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You only look once: Unified, real-time object detection. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 779–788.

Risbo, T. (1996). Fourier transform summation of Legendre series and D-functions. *Journal of Geodesy*, 70(7):383–396.

Rosebrock, A. (2017). *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. Em *European conference on computer vision*, páginas 430–443. Springer.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. Em *Proceedings third international conference on 3-D digital imaging and modeling*, páginas 145–152. IEEE.

Russell, S. and Norvig, P. (2013). *Inteligência Artificial*. Rio de Janeiro: Campus Elsevier.

Rusu, R. B., Blodow, N. and Beetz, M. (2009). Fast point feature histograms (FPFH) for 3D registration. Em *2009 IEEE international conference on robotics and automation*, páginas 3212–3217. IEEE.

Rusu, R. B., Blodow, N., Marton, Z. C. and Beetz, M. (2008). Aligning point cloud views using persistent feature histograms. Em *2008 IEEE/RSJ international conference on intelligent robots and systems*, páginas 3384–3391. IEEE.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point cloud library (pcl). Em *2011 IEEE international conference on robotics and automation*, páginas 1–4. IEEE.

Salti, S., Tombari, F. and Di Stefano, L. (2014). SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 4510–4520.

Schultz, M. and Joachims, T. (2004). Learning a distance metric from relative comparisons. Em *Advances in neural information processing systems*, páginas 41–48.

Schwarz, M., Schulz, H. and Behnke, S. (2015). RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. Em *2015 IEEE international conference on robotics and automation (ICRA)*, páginas 1329–1335. IEEE.

Sedaghat, N., Zolfaghari, M., Amiri, E. and Brox, T. (2016). Orientation-boosted voxel nets for 3D object recognition. *arXiv preprint arXiv:1604.03351*.

Shepard, R. N. and Metzler, J. (1971). Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703.

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A. and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in RGB-D images. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 2930–2937.

Silva, L. J. S., da Silva, D. L. S., Raposo, A. B., Velho, L. and Lopes, H. C. V. (2019). Tensorpose: Real-time pose estimation for interactive applications. *Computers & Graphics*, 85:1–14.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Em *International Conference on Learning Representations*.

Song, S. and Xiao, J. (2014). Sliding shapes for 3D object detection in depth images. Em *European conference on computer vision*, páginas 634–651. Springer.

Song, S. and Xiao, J. (2016). Deep sliding shapes for amodal 3D object detection in RGB-D images. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 808–816.

Spezialetti, R., Salti, S. and Stefano, L. D. (2019). Learning an effective equivariant 3D descriptor without supervision. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 6401–6410.

Su, H., Maji, S., Kalogerakis, E. and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. Em *Proceedings of the IEEE international conference on computer vision*, páginas 945–953.

Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. Em *Thirty-first AAAI conference on artificial intelligence*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015). Going deeper with convolutions. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 2818–2826.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. Em *International Conference on Machine Learning*, páginas 6105–6114.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F. and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 6411–6420.

Tombari, F. and Di Stefano, L. (2010). Object recognition in 3D scenes with occlusions and clutter by hough voting. Em *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*, páginas 349–355. IEEE.

Tombari, F., Salti, S. and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. Em *European conference on computer vision*, páginas 356–369. Springer.

Tombari, F., Salti, S. and Di Stefano, L. (2013). Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision*, 102(1-3):198–220.

Valentin, J., Dai, A., Nießner, M., Kohli, P., Torr, P., Izadi, S. and Keskin, C. (2016). Learning to navigate the energy landscape. Em *3D Vision (3DV), 2016 Fourth International Conference on*, páginas 323–332. IEEE.

Vandenberg, S. G. and Kuse, A. R. (1978). Mental rotations, a group test of three-dimensional spatial visualization. *Perceptual and motor skills*, 47(2):599–604.

Vock, R., Dieckmann, A., Ochmann, S. and Klein, R. (2019). Fast template matching and pose estimation in 3D point clouds. *Computers & Graphics*, 79:36–45.

Wang, W., Chen, L., Chen, D., Li, S. and Kühnlenz, K. (2013). Fast object recognition and 6D pose estimation using viewpoint oriented color-shape histogram. Em *2013 IEEE International Conference on Multimedia and Expo (ICME)*, páginas 1–6. IEEE.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2).

Wu, C.-Y., Manmatha, R., Smola, A. J. and Krahenbuhl, P. (2017). Sampling matters in deep embedding learning. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 2840–2848.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. and Xiao, J. (2015). 3D shapenets: A deep representation for volumetric shapes. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 1912–1920.

Xiao, J., Owens, A. and Torralba, A. (2013). SUN3D: A database of big spaces reconstructed using SfM and object labels. Em *Proceedings of the IEEE International Conference on Computer Vision*, páginas 1625–1632.

Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K. (2017). Aggregated residual transformations for deep neural networks. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 1492–1500.

Yan, W., Xu, Z., Zhou, X., Su, Q., Li, S. and Wu, H. (2020). Fast object pose estimation using adaptive threshold for bin-picking. *IEEE Access*, 8:63055–63064.

Yang, J., Xiao, Y. and Cao, Z. (2018a). Toward the repeatability and robustness of the local reference frame for 3D shape matching: An evaluation. *IEEE Transactions on Image Processing*, 27(8):3766–3781.

Yang, J., Zhang, Q., Xiao, Y. and Cao, Z. (2017). Toldi: An effective and robust approach for 3D local shape description. *Pattern Recognition*, 65:175–187.

Yang, Y., Feng, C., Shen, Y. and Tian, D. (2018b). FoldingNet: Point cloud auto-encoder via deep grid deformation. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 206–215.

You, Y., Lou, Y., Liu, Q., Ma, L., Wang, W., Tai, Y. and Lu, C. (2018). PRIN: Pointwise rotation-invariant network. *arXiv preprint arXiv:1811.09361*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R. and Smola, A. J. (2017). Deep sets. Em *Advances in neural information processing systems*, páginas 3391–3401.

Zaki, H. F., Shafait, F. and Mian, A. (2019). Viewpoint invariant semantic object and scene categorization with RGB-D sensors. *Autonomous Robots*, 43(4):1005–1022.

Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J. and Funkhouser, T. (2017a). 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1802–1811.

Zeng, A., Yu, K.-T., Song, S., Suo, D., Walker, E., Rodriguez, A. and Xiao, J. (2017b). Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. Em *2017 IEEE international conference on robotics and automation (ICRA)*, páginas 1386–1383. IEEE.

Zhang, H., Zhuang, B. and Liu, Y. (2017). Object classification based on 3D point clouds covariance descriptor. Em *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 2, páginas 234–237. IEEE.

Zhang, K., Hao, M., Wang, J., de Silva, C. W. and Fu, C. (2019a). Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features. *arXiv preprint arXiv:1904.10014*.

Zhang, Z., Hua, B.-S., Rosen, D. W. and Yeung, S.-K. (2019b). Rotation invariant convolutions for 3D point clouds deep learning. Em *2019 International Conference on 3D Vision (3DV)*, páginas 204–213. IEEE.

Zhao, Y., Birdal, T., Deng, H. and Tombari, F. (2019). 3D point capsule networks. Em *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3D object recognition. Em *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, páginas 689–696. IEEE.

Zhou, Q.-Y., Park, J. and Koltun, V. (2016). Fast global registration. Em *European Conference on Computer Vision*, páginas 766–782. Springer.

Zhou, Q.-Y., Park, J. and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*.

Zhou, Y., Barnes, C., Lu, J., Yang, J. and Li, H. (2019). On the continuity of rotation representations in neural networks. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 5745–5753.

Zhou, Y.-T. and Chellappa, R. (1988). Computation of optical flow using a neural network. Em *ICNN*, páginas 71–78.

Zhu, A., Yang, J., Zhao, C., Xian, K., Cao, Z. and Li, X. (2020). LRF-Net: Learning local reference frames for 3D local shape description and matching. *arXiv preprint arXiv:2001.07832*.

Zia, S., Yuksel, B., Yuret, D. and Yemez, Y. (2017). RGB-D object recognition using deep convolutional neural networks. Em *Proceedings of the IEEE International Conference on Computer Vision Workshops*, páginas 896–903.

APPENDIX A – DETAILED DATA OF CHAPTER 6

A.1 CATEGORY RECOGNITION DATA

Table A.1: Results on category recogntion combining color feature extractors and ML classifiers

	AlexNet						Resnet101						VGG_16						Inception v3						MobileNet v2						ResNext-101 32x8d						EfficientNet-B7					
Split	RF	SVC	GNB	kSVC	MLP	LR	RF	SVC	GNB	kSVC	MLP	LR	RF	SVC	GNB	kSVC	MLP	LR	RF	SVC	GNB	kSVC	MLP	LR	RF	SVC	GNB	kSVC	MLP	LR	RF	SVC	GNB	kSVC	MLP	LR	RF	SVC	GNB	kSVC	MLP	LR
1	0.616	0.676	0.501	0.667	0.632	0.681	0.761	0.795	0.726	0.814	0.811	0.809	0.656	0.740	0.569	0.728	0.682	0.733	0.682	0.739	0.678	0.773	0.766	0.776	0.724	0.797	0.688	0.791	0.778	0.805	0.767	0.817	0.723	0.829	0.839	0.843	0.770	0.810	0.737	0.804	0.795	0.813
2	0.645	0.748	0.553	0.717	0.671	0.757	0.811	0.844	0.770	0.849	0.768	0.852	0.733	0.788	0.653	0.792	0.763	0.803	0.745	0.824	0.718	0.842	0.842	0.834	0.800	0.834	0.764	0.864	0.825	0.862	0.830	0.861	0.813	0.879	0.873	0.877	0.849	0.883	0.829	0.898	0.899	0.906
3	0.603	0.677	0.559	0.656	0.607	0.702	0.749	0.779	0.777	0.808	0.784	0.803	0.684	0.751	0.603	0.740	0.676	0.753	0.710	0.764	0.699	0.797	0.765	0.792	0.726	0.762	0.701	0.783	0.753	0.803	0.762	0.817	0.752	0.825	0.805	0.825	0.781	0.829	0.725	0.841	0.850	0.853
4	0.648	0.739	0.534	0.711	0.699	0.740	0.798	0.818	0.772	0.840	0.835	0.850	0.691	0.755	0.619	0.743	0.737	0.752	0.741	0.828	0.726	0.839	0.833	0.835	0.749	0.811	0.707	0.803	0.797	0.835	0.776	0.825	0.757	0.822	0.805	0.832	0.781	0.850	0.751	0.833	0.830	0.855
5	0.623	0.732	0.520	0.689	0.631	0.720	0.784	0.810	0.731	0.813	0.789	0.833	0.723	0.766	0.622	0.768	0.702	0.771	0.723	0.785	0.710	0.797	0.817	0.810	0.755	0.797	0.704	0.800	0.772	0.811	0.787	0.835	0.769	0.831	0.835	0.833	0.779	0.830	0.756	0.823	0.833	0.839
6	0.619	0.704	0.510	0.661	0.650	0.711	0.764	0.783	0.715	0.804	0.798	0.808	0.678	0.746	0.596	0.734	0.730	0.776	0.725	0.801	0.694	0.810	0.797	0.814	0.733	0.779	0.665	0.773	0.692	0.785	0.781	0.824	0.749	0.816	0.801	0.828	0.810	0.868	0.767	0.866	0.876	0.883
7	0.667	0.733	0.569	0.710	0.730	0.749	0.787	0.829	0.750	0.837	0.840	0.842	0.730	0.806	0.641	0.795	0.776	0.820	0.746	0.803	0.716	0.831	0.810	0.822	0.757	0.814	0.718	0.813	0.819	0.839	0.815	0.868	0.779	0.858	0.855	0.879	0.837	0.876	0.784	0.886	0.887	0.896
8	0.684	0.740	0.607	0.754	0.709	0.746	0.836	0.857	0.819	0.873	0.844	0.874	0.730	0.773	0.656	0.788	0.766	0.779	0.731	0.799	0.742	0.826	0.791	0.804	0.799	0.826	0.755	0.860	0.834	0.855	0.813	0.848	0.781	0.867	0.868	0.858	0.813	0.831	0.797	0.853	0.847	0.851
9	0.664	0.731	0.531	0.727	0.671	0.737	0.779	0.825	0.726	0.845	0.823	0.848	0.707	0.759	0.609	0.751	0.723	0.767	0.706	0.795	0.685	0.805	0.798	0.801	0.767	0.819	0.718	0.826	0.802	0.821	0.828	0.868	0.778	0.880	0.859	0.872	0.844	0.886	0.779	0.894	0.894	0.897
10	0.676	0.754	0.605	0.738	0.742	0.761	0.793	0.816	0.778	0.826	0.805	0.822	0.714	0.776	0.632	0.771	0.751	0.792	0.708	0.774	0.692	0.777	0.789	0.790	0.763	0.832	0.705	0.816	0.815	0.824	0.804	0.841	0.787	0.840	0.848	0.857	0.781	0.832	0.732	0.824	0.823	0.840
Avg	0.645	0.724	0.549	0.703	0.674	0.730	0.786	0.816	0.756	0.831	0.810	0.834	0.704	0.766	0.620	0.761	0.731	0.775	0.722	0.791	0.706	0.810	0.801	0.808	0.757	0.807	0.713	0.813	0.789	0.824	0.796	0.840	0.769	0.845	0.839	0.850	0.804	0.850	0.766	0.852	0.853	0.863
Std	0.028	0.028	0.037	0.034	0.045	0.026	0.025	0.025	0.032	0.022	0.026	0.023	0.026	0.020	0.027	0.025	0.035	0.026	0.020	0.027	0.020	0.024	0.026	0.019	0.027	0.023	0.029	0.030	0.042	0.024	0.025	0.020	0.025	0.024	0.027	0.021	0.030	0.027	0.032	0.033	0.034	0.031

Table A.2: Results on category recogntion combining shape feature extractors and ML classifiers

	LEAD						**Spezialetti et al.**						**LEAD-PN**					
Split	**RF**	**SVC**	**GNB**	**MLP**	**kSVC**	**LR**	**RF**	**SVC**	**GNB**	**MLP**	**kSVC**	**LR**	**RF**	**SVC**	**GNB**	**MLP**	**kSVC**	**LR**
1	0.4514	0.3478	0.3248	0.5068	0.4574	0.2891	0.4578	0.3280	0.3083	0.5023	0.4561	0.2245	0.3404	0.3071	0.1240	0.3657	0.2653	0.2989
2	0.4807	0.3555	0.3211	0.5027	0.4859	0.2864	0.4817	0.3513	0.3451	0.5078	0.4893	0.2492	0.3412	0.3082	0.1380	0.3570	0.2556	0.2949
3	0.4396	0.3317	0.2814	0.4859	0.4583	0.2591	0.4382	0.3187	0.2796	0.4881	0.4500	0.2293	0.3300	0.3033	0.1196	0.3508	0.2654	0.2931
4	0.4875	0.3637	0.3242	0.5079	0.4800	0.2889	0.4901	0.3432	0.3249	0.5066	0.4835	0.2490	0.3449	0.3186	0.1289	0.3548	0.2673	0.3001
5	0.4672	0.3372	0.3359	0.4915	0.4629	0.2662	0.4730	0.3236	0.3265	0.5012	0.4637	0.2267	0.3398	0.3132	0.1486	0.3654	0.2740	0.3004
6	0.4749	0.3491	0.3325	0.5187	0.4850	0.2828	0.4858	0.3410	0.3370	0.5197	0.4848	0.2633	0.3511	0.3269	0.1390	0.3839	0.2761	0.3198
7	0.4870	0.3632	0.3564	0.5038	0.4982	0.2832	0.4955	0.3466	0.3256	0.5191	0.4932	0.2585	0.3348	0.3073	0.1405	0.3583	0.2660	0.3082
8	0.4837	0.3674	0.3730	0.5055	0.4828	0.2835	0.4908	0.3508	0.3646	0.5216	0.4807	0.2481	0.3230	0.2918	0.1385	0.3465	0.2525	0.2987
9	0.4631	0.3436	0.3358	0.5012	0.4685	0.2571	0.4683	0.3313	0.3452	0.5078	0.4697	0.2348	0.3289	0.3157	0.1472	0.3507	0.2578	0.3052
10	0.4738	0.3470	0.3473	0.5168	0.4882	0.2602	0.4856	0.3286	0.3479	0.5408	0.4954	0.2380	0.3575	0.3307	0.1489	0.3750	0.2783	0.3269
Avg	0.4709	0.3506	0.3332	**0.5041**	0.4767	0.2757	0.4767	0.3363	0.3305	**0.5115**	0.4766	0.2421	0.3392	0.3123	0.1373	**0.3608**	0.2658	0.3046
Std	0.0158	0.0118	0.0243	0.0100	0.0140	0.0133	0.0178	0.0117	0.0238	0.0144	0.0158	0.0135	0.0105	0.0114	0.0102	0.0117	0.0087	0.0109

Table A.3: Results on category recogntion combining color + shape feature extractors and ML classifiers

	resnext101_32x8d+LEAD-PN						EfficientNet-B7 + LEAD-PN						Resnet101+LEAD-PN						MobileNet_v2+LEAD-PN					
Runs	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**	**RF**	**SVC**	**GNB**	**kSVC**	**MLP**	**LR**	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**
1	0.7697	0.8198	0.5695	0.8420	0.8383	0.8368	0.7832	0.8032	0.5757	0.8072	0.8208	0.8217	0.7584	0.8100	0.5979	0.8204	0.7991	0.8110	0.7281	0.8170	0.5422	0.8143	0.7981	0.7893
2	0.8522	0.8874	0.5935	0.8931	0.8748	0.8839	0.8644	0.8873	0.6020	0.9004	0.9112	0.9087	0.8258	0.8598	0.5839	0.8732	0.8596	0.8623	0.8104	0.8602	0.5646	0.8759	0.8666	0.8643
3	0.7700	0.8336	0.5828	0.8392	0.8219	0.8336	0.7864	0.8356	0.5390	0.8425	0.8516	0.8552	0.7606	0.8055	0.6011	0.8262	0.8148	0.8110	0.7305	0.7913	0.5347	0.8068	0.7951	0.7953
4	0.8014	0.8386	0.5479	0.8413	0.8194	0.8351	0.7931	0.8611	0.5532	0.8366	0.8599	0.8682	0.8101	0.8399	0.5662	0.8625	0.8558	0.8458	0.7714	0.8179	0.5197	0.8399	0.8390	0.8294
5	0.8100	0.8365	0.5871	0.8444	0.8263	0.8461	0.7870	0.8308	0.5778	0.8200	0.8388	0.8433	0.7936	0.8492	0.5743	0.8489	0.8339	0.8214	0.7629	0.8259	0.5361	0.8177	0.7887	0.8106
6	0.7935	0.8266	0.5654	0.8362	0.8069	0.8136	0.8163	0.8699	0.5794	0.8657	0.8797	0.8817	0.7791	0.8133	0.5818	0.8146	0.8021	0.8066	0.7487	0.7994	0.5394	0.8047	0.7702	0.7809
7	0.8309	0.8719	0.6172	0.8821	0.8711	0.8564	0.8444	0.8787	0.6149	0.8732	0.8859	0.8925	0.8002	0.8610	0.6207	0.8538	0.8505	0.8371	0.7735	0.8546	0.5695	0.8525	0.8411	0.8267
8	0.8350	0.8480	0.6067	0.8774	0.8724	0.8606	0.8082	0.8340	0.5941	0.8504	0.8619	0.8579	0.8440	0.8630	0.6374	0.8829	0.8666	0.8694	0.8065	0.8492	0.5671	0.8600	0.8578	0.8527
9	0.8389	0.8832	0.5896	0.8862	0.8744	0.8791	0.8494	0.8697	0.6018	0.8925	0.8970	0.8939	0.7938	0.8439	0.5842	0.8574	0.8474	0.8488	0.7858	0.8355	0.5691	0.8262	0.8009	0.8274
10	0.8283	0.8529	0.6165	0.8709	0.8460	0.8558	0.7892	0.8240	0.5887	0.8234	0.8458	0.8473	0.8064	0.8312	0.6274	0.8437	0.8336	0.8362	0.7725	0.8375	0.5577	0.8392	0.8241	0.8233
Avg	0.8130	0.8499	0.5876	**0.8613**	0.8452	0.8501	0.8122	0.8494	0.5827	0.8512	0.8653	**0.8670**	0.7972	0.8377	0.5975	**0.8484**	0.8363	0.8350	0.7690	0.8288	0.5500	**0.8337**	0.8182	0.8200
Std	0.0289	0.0236	0.0225	**0.0226**	0.0263	0.0216	0.0302	0.0275	0.0231	0.0313	0.0280	**0.0270**	0.0268	0.0218	0.0239	**0.0225**	0.0240	0.0221	0.0280	0.0229	0.0177	**0.0239**	0.0322	0.0267

	resnext101_32x8d+LEAD						EfficientNet-B7 + LEAD						Resnet101+LEAD						MobileNet_v2+LEAD					
Runs	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**	**RF**	**SVC**	**GNB**	**kSVC**	**MLP**	**LR**	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**
1	0.8102	0.8365	0.7651	0.8800	0.8787	0.8671	0.8243	0.8217	0.7999	0.8385	0.8430	0.8524	0.7933	0.8362	0.7739	0.8489	0.8392	0.8416	0.7753	0.8368	0.7492	0.8443	0.8282	0.8200
2	0.8728	0.9095	0.8507	0.9160	0.9171	0.9097	0.8842	0.9007	0.8554	0.9210	0.9271	0.9234	0.8547	0.8751	0.8293	0.8850	0.8882	0.8816	0.8518	0.8697	0.8263	0.8961	0.8706	0.8851
3	0.8045	0.8549	0.7603	0.8609	0.8495	0.8629	0.8043	0.8498	0.7331	0.8716	0.8687	0.8820	0.7877	0.8234	0.7890	0.8429	0.8177	0.8488	0.7838	0.8208	0.7356	0.8409	0.8228	0.8374
4	0.8261	0.8670	0.7801	0.8720	0.8690	0.8632	0.8206	0.8766	0.7682	0.8648	0.8452	0.8825	0.8365	0.8534	0.7934	0.8782	0.8772	0.8658	0.8132	0.8308	0.7544	0.8546	0.8607	0.8582
5	0.8327	0.8691	0.7972	0.8649	0.8618	0.8654	0.8198	0.8524	0.7909	0.8517	0.8581	0.8598	0.8156	0.8591	0.7870	0.8590	0.8175	0.8392	0.8054	0.8409	0.7686	0.8315	0.8219	0.8378
6	0.8132	0.8383	0.7796	0.8540	0.8477	0.8469	0.8557	0.8849	0.7937	0.8911	0.8716	0.8957	0.7925	0.8398	0.7609	0.8429	0.8171	0.8345	0.7803	0.8241	0.7355	0.8219	0.8106	0.8164
7	0.8471	0.8783	0.8095	0.8834	0.8699	0.8774	0.8666	0.8826	0.8202	0.8849	0.8987	0.9006	0.8150	0.8800	0.8035	0.8644	0.8689	0.8569	0.7993	0.8594	0.7581	0.8643	0.8496	0.8490
8	0.8393	0.8724	0.8124	0.8897	0.8896	0.8789	0.8465	0.8448	0.8139	0.8824	0.8845	0.8839	0.8566	0.8727	0.8375	0.8959	0.8495	0.8892	0.8357	0.8542	0.7964	0.8805	0.8811	0.8788
9	0.8614	0.8971	0.7982	0.9016	0.9018	0.8991	0.8656	0.8844	0.7990	0.9056	0.9071	0.9051	0.8274	0.8619	0.7733	0.8697	0.8708	0.8707	0.8301	0.8550	0.7727	0.8525	0.8510	0.8545
10	0.8514	0.8735	0.8146	0.8943	0.8949	0.8804	0.8311	0.8448	0.7673	0.8593	0.8755	0.8802	0.8334	0.8546	0.8095	0.8575	0.8592	0.8498	0.8122	0.8552	0.7617	0.8560	0.8270	0.8385
Avg	0.8359	0.8697	0.7968	**0.8817**	0.8780	0.8751	0.8419	0.8643	0.7942	0.8771	0.8780	**0.8866**	0.8213	0.8556	0.7957	**0.8644**	0.8505	0.8578	0.8087	0.8447	0.7659	**0.8543**	0.8423	0.8476
Std	0.0227	0.0230	0.0269	**0.0194**	0.0227	0.0185	0.0257	0.0249	0.0334	0.0251	0.0270	**0.0210**	0.0249	0.0182	0.0246	**0.0179**	0.0266	0.0185	0.0252	0.0164	0.0278	**0.0221**	0.0236	0.0225

	resnext101_32x8d + Spezialetti et al.						EfficientNet-B7 + Spezialetti et al.						Resnet101 + Spezialetti et al.						MobileNet_v2 + Spezialetti et al.					
Runs	**RF**	**SVC**	**GNB**	**kSVC**	**MLP**	**LR**	**RF**	**SVC**	**GNB**	**kSVC**	**MLP**	**LR**	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**	**RF**	**SVC**	**GNB**	**LR**	**MLP**	**kSVC**
1	0.7897	0.8339	0.7540	0.8287	0.8137	0.8651	0.8137	0.8153	0.7836	0.8206	0.8376	0.8339	0.7827	0.8376	0.7657	0.8349	0.8160	0.8102	0.7581	0.8312	0.7389	0.8309	0.7925	0.7904
2	0.8671	0.9163	0.8309	0.8971	0.9139	0.9140	0.8825	0.8981	0.8522	0.9149	0.8997	0.9168	0.8505	0.8717	0.8160	0.8835	0.8495	0.8713	0.8435	0.8805	0.8013	0.8958	0.8612	0.8758
3	0.7967	0.8645	0.7613	0.8548	0.8105	0.8583	0.8072	0.8486	0.7272	0.8569	0.8669	0.8726	0.7930	0.8212	0.7811	0.8438	0.8432	0.8369	0.7773	0.8005	0.7252	0.8364	0.8234	0.8271
4	0.8190	0.8570	0.7680	0.8421	0.8511	0.8635	0.8158	0.8724	0.7582	0.8490	0.8666	0.8734	0.8306	0.8459	0.7877	0.8693	0.8245	0.8561	0.8008	0.8272	0.7477	0.8477	0.7987	0.8345
5	0.8372	0.8539	0.7786	0.8570	0.8462	0.8565	0.8163	0.8487	0.7873	0.8441	0.8542	0.8532	0.8089	0.8624	0.7703	0.8565	0.8370	0.8256	0.7961	0.8430	0.7458	0.8292	0.8071	0.8226
6	0.8087	0.8474	0.7729	0.8401	0.8341	0.8515	0.8547	0.8862	0.7928	0.8866	0.8460	0.8939	0.7951	0.8410	0.7614	0.8415	0.8414	0.8300	0.7770	0.8224	0.7362	0.8278	0.8132	0.8125
7	0.8443	0.8889	0.7984	0.8817	0.8712	0.8837	0.8526	0.8891	0.8012	0.8833	0.9064	0.9045	0.8159	0.8830	0.7966	0.8620	0.8530	0.8595	0.7947	0.8682	0.7441	0.8721	0.8572	0.8528
8	0.8443	0.8647	0.8014	0.8645	0.8798	0.8768	0.8411	0.8406	0.7998	0.8760	0.8731	0.8696	0.8534	0.8632	0.8313	0.8883	0.8376	0.8716	0.8288	0.8498	0.7904	0.8701	0.8693	0.8620
9	0.8514	0.8970	0.7887	0.8830	0.8574	0.8977	0.8678	0.8882	0.7913	0.8975	0.9004	0.9001	0.8252	0.8667	0.7652	0.8667	0.8626	0.8528	0.8231	0.8452	0.7647	0.8576	0.8441	0.8414
10	0.8501	0.8704	0.8031	0.8646	0.8451	0.8881	0.8366	0.8375	0.7627	0.8517	0.8620	0.8685	0.8311	0.8555	0.7952	0.8589	0.8302	0.8473	0.8189	0.8534	0.7457	0.8519	0.8368	0.8265
Avg	0.8308	0.8694	0.7857	0.8614	0.8523	**0.8755**	0.8389	0.8625	0.7856	0.8680	0.8713	**0.8786**	0.8186	0.8548	0.7871	**0.8605**	0.8395	0.8461	0.8018	0.8421	0.7540	**0.8520**	0.8303	0.8346
Std	0.0258	0.0248	0.0233	0.0214	0.0309	**0.0203**	0.0255	0.0279	0.0328	0.0285	0.0238	**0.0252**	0.0240	0.0184	0.0233	**0.0174**	0.0138	0.0202	0.0267	0.0231	0.0243	**0.0223**	0.0273	0.0248

A.2 INSTANCE RECOGNITION LSO

Table A.4: Results on instance recognition (LSO) combining feature extractors and ML classifiers

	Color					
Classifier	RF	SVC	GNB	MLP	LR	kSVC
AlexNet	0.8173	0.8955	0.7522	0.8499	0.8977	0.8565
Resnet101	0.8865	0.9383	0.8716	0.9011	0.9414	0.9180
VGG_16	0.7869	0.8766	0.7280	0.8349	0.8875	0.8413
Inception v3	0.7582	0.8690	0.7772	0.8698	0.8806	0.8664
MobileNet v2	0.8848	0.9341	0.8721	0.9098	0.9381	0.9235
ResNext-101 32x8d	0.8965	0.9376	0.8776	0.9197	0.9393	0.9242
EfficientNet-B7	0.8739	0.9370	0.8366	0.9285	0.9383	0.9234

	Shape					
Classifier	RF	SVC	GNB	MLP	LR	kSVC
LEAD-PN	0.1267	0.1226	0.0488	0.1302	0.1060	0.0863
LEAD	0.2201	0.1728	0.1715	0.2341	0.1107	0.2176
Spezialetti et al.	0.2410	0.1784	0.1750	0.2527	0.1036	0.2304

	Color + Shape					
Classifier	RF	SVC	GNB	MLP	LR	kSVC
ResNet101 + LEAD-PN	0.8901	0.9358	0.6705	0.9244	0.9408	0.9230
ResNet101 + Spezialetti et al.	0.8910	0.9420	0.8699	0.9155	0.9370	0.9283
ResNet101 + LEAD	0.8922	0.9445	0.8686	0.9268	0.9415	0.9254
MobileNet v2 + LEAD-PN	0.8853	0.9243	0.6535	0.9274	0.9359	0.9252
MobileNet v2 + Spezialetti et al.	0.8949	0.9363	0.8696	0.9285	0.9384	0.9298
MobileNet v2 + LEAD	0.8893	0.9296	0.8662	0.9207	0.9329	0.9328
ResNeXt101 32x8d + LEAD-PN	0.8979	0.9277	0.6712	0.9175	0.9371	0.9228
ResNeXt101 32x8d + Spezialetti et al.	0.8997	0.9378	0.8732	0.9277	0.9393	0.9282
ResNeXt101 32x8d + LEAD	0.8950	0.9350	0.8716	0.9202	0.9337	0.9290
EfficientNet-B7 + LEAD-PN	0.8724	0.9256	0.6318	0.9174	0.9300	0.9143
EfficientNet-B7 + Spezialetti et al.	0.8824	0.9315	0.8367	0.8972	0.9376	0.9215
EfficientNet-B7 + LEAD	0.8748	0.9311	0.8320	0.9210	0.9301	0.9199

A.3 INSTANCE RECOGNITION ACF

Table A.5: Results on instance recognition (ACF) combining color feature extractors and ML classifiers

	ResNet101						ResNext-101 32x8d						EfficientNet-B7						MobileNet v2						AlexNet						VGG16						inception v3					
Split	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC
1	0.905	0.956	0.845	0.926	0.952	0.929	0.908	0.951	0.856	0.934	0.961	0.930	0.893	0.957	0.810	0.941	0.953	0.939	0.912	0.954	0.833	0.921	0.957	0.934	0.862	0.928	0.741	0.915	0.948	0.894	0.833	0.910	0.706	0.876	0.909	0.857	0.794	0.888	0.747	0.880	0.904	0.887
2	0.897	0.951	0.845	0.923	0.956	0.925	0.910	0.962	0.868	0.936	0.954	0.933	0.893	0.950	0.813	0.929	0.959	0.933	0.912	0.951	0.851	0.932	0.956	0.942	0.873	0.934	0.744	0.907	0.933	0.885	0.819	0.908	0.707	0.861	0.910	0.865	0.786	0.901	0.745	0.882	0.900	0.888
3	0.906	0.950	0.849	0.919	0.953	0.924	0.914	0.954	0.880	0.930	0.959	0.932	0.896	0.953	0.814	0.939	0.953	0.929	0.903	0.950	0.850	0.928	0.954	0.932	0.866	0.938	0.739	0.906	0.938	0.875	0.826	0.904	0.692	0.871	0.905	0.866	0.782	0.886	0.746	0.872	0.897	0.885
4	0.908	0.952	0.858	0.922	0.954	0.927	0.913	0.958	0.858	0.936	0.949	0.935	0.893	0.958	0.804	0.946	0.959	0.928	0.912	0.950	0.849	0.935	0.962	0.939	0.871	0.935	0.759	0.911	0.937	0.891	0.828	0.905	0.705	0.865	0.911	0.861	0.784	0.895	0.740	0.867	0.900	0.886
5	0.900	0.952	0.851	0.923	0.956	0.922	0.903	0.955	0.865	0.941	0.955	0.936	0.895	0.959	0.814	0.939	0.958	0.920	0.911	0.954	0.846	0.922	0.959	0.929	0.870	0.943	0.747	0.896	0.942	0.886	0.834	0.900	0.709	0.873	0.907	0.842	0.805	0.896	0.735	0.861	0.910	0.878
6	0.902	0.946	0.859	0.923	0.949	0.928	0.910	0.955	0.859	0.918	0.960	0.933	0.890	0.958	0.817	0.944	0.962	0.933	0.904	0.951	0.850	0.935	0.958	0.935	0.871	0.939	0.748	0.890	0.937	0.888	0.820	0.906	0.702	0.867	0.909	0.876	0.783	0.895	0.741	0.888	0.904	0.886
7	0.907	0.952	0.850	0.933	0.954	0.928	0.910	0.956	0.859	0.916	0.961	0.933	0.894	0.952	0.813	0.945	0.955	0.936	0.908	0.954	0.852	0.924	0.960	0.930	0.866	0.925	0.745	0.896	0.940	0.887	0.827	0.901	0.698	0.862	0.924	0.871	0.794	0.891	0.731	0.893	0.910	0.881
8	0.904	0.945	0.837	0.924	0.954	0.924	0.912	0.952	0.860	0.947	0.957	0.945	0.880	0.949	0.803	0.946	0.951	0.924	0.895	0.956	0.827	0.933	0.965	0.930	0.855	0.936	0.737	0.907	0.935	0.881	0.839	0.900	0.699	0.853	0.917	0.862	0.785	0.901	0.744	0.875	0.901	0.874
9	0.901	0.958	0.843	0.927	0.948	0.924	0.911	0.955	0.867	0.927	0.956	0.931	0.903	0.952	0.802	0.942	0.946	0.933	0.900	0.955	0.846	0.940	0.958	0.943	0.844	0.937	0.753	0.904	0.939	0.888	0.837	0.910	0.695	0.879	0.901	0.859	0.798	0.896	0.754	0.860	0.904	0.881
10	0.906	0.959	0.841	0.928	0.954	0.925	0.914	0.953	0.854	0.932	0.956	0.937	0.888	0.961	0.810	0.942	0.960	0.926	0.895	0.953	0.834	0.932	0.957	0.934	0.872	0.942	0.749	0.899	0.936	0.890	0.819	0.914	0.696	0.877	0.909	0.858	0.791	0.904	0.745	0.879	0.902	0.887
Avg	0.904	0.952	0.848	0.925	0.953	0.925	0.911	0.955	0.863	0.932	0.957	0.934	0.893	0.955	0.810	0.941	0.956	0.930	0.905	0.953	0.844	0.930	0.959	0.935	0.865	0.936	0.746	0.903	0.939	0.886	0.828	0.906	0.701	0.868	0.910	0.862	0.790	0.895	0.743	0.876	0.903	0.883
Std	0.003	0.005	0.007	0.004	0.003	0.002	0.003	0.003	0.008	0.009	0.004	0.004	0.006	0.004	0.005	0.005	0.005	0.006	0.007	0.002	0.009	0.006	0.003	0.005	0.009	0.006	0.007	0.008	0.004	0.005	0.008	0.005	0.006	0.008	0.006	0.009	0.008	0.006	0.006	0.011	0.004	0.005

Table A.6: Results on instance recognition (ACF) combining shape feature extractors and ML classifiers

	LEAD-PN						LEAD						Spezialetti et al.					
Split	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC
1	0.1729	0.1422	0.0488	0.1744	0.1355	0.1084	0.2616	0.1802	0.1745	0.3104	0.1185	0.2482	0.3080	0.1843	0.1796	0.3111	0.1117	0.2425
2	0.1674	0.1498	0.0527	0.1788	0.1256	0.1030	0.2664	0.1870	0.1832	0.3041	0.1107	0.2350	0.2857	0.1779	0.1886	0.3109	0.0986	0.2564
3	0.1638	0.1487	0.0643	0.1748	0.1330	0.1058	0.2640	0.1769	0.1733	0.3092	0.1102	0.2522	0.2955	0.1774	0.1796	0.3312	0.1102	0.2580
4	0.1751	0.1543	0.0606	0.1742	0.1273	0.1014	0.2821	0.1712	0.1803	0.2983	0.1150	0.2428	0.3079	0.1838	0.1833	0.3220	0.1146	0.2629
5	0.1771	0.1453	0.0535	0.1701	0.1262	0.1037	0.2623	0.1857	0.1751	0.2929	0.1165	0.2472	0.2945	0.1942	0.1861	0.3174	0.1073	0.2645
6	0.1626	0.1541	0.0527	0.1705	0.1285	0.1038	0.2832	0.1809	0.1682	0.3000	0.1192	0.2431	0.3014	0.1944	0.1796	0.3058	0.1206	0.2638
7	0.1680	0.1414	0.0603	0.1765	0.1318	0.1019	0.2704	0.1814	0.1834	0.2961	0.1062	0.2465	0.2988	0.1903	0.1719	0.3201	0.1009	0.2486
8	0.1738	0.1500	0.0579	0.1847	0.1282	0.1090	0.2777	0.1884	0.1708	0.3023	0.1109	0.2312	0.3031	0.1899	0.1899	0.3101	0.1069	0.2540
9	0.1683	0.1506	0.0598	0.1692	0.1234	0.1003	0.2909	0.1845	0.1887	0.3006	0.1178	0.2458	0.2924	0.1860	0.1799	0.3234	0.1114	0.2679
10	0.1724	0.1477	0.0537	0.1811	0.1279	0.0992	0.2866	0.1713	0.1831	0.3128	0.1087	0.2543	0.2985	0.1768	0.1858	0.3171	0.1049	0.2626
Avg	0.1702	0.1484	0.0564	0.1754	0.1287	0.1037	0.2745	0.1808	0.1781	0.3027	0.1134	0.2446	0.2986	0.1855	0.1824	0.3169	0.1087	0.2581
Std	0.0049	0.0044	0.0048	0.0050	0.0037	0.0033	0.0109	0.0061	0.0066	0.0065	0.0046	0.0071	0.0069	0.0067	0.0054	0.0076	0.0065	0.0079

Table A.7: Results on instance recognition (ACF) combining color + shape feature extractors and ML classifiers

	Resnet+LEAD						Resnext+LEAD						MobileNet+LEAD						EfficientNet+LEAD					
Split	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC
1	0.8970	0.9547	0.8386	0.9418	0.9561	0.9414	0.9019	0.9574	0.8559	0.9476	0.9546	0.9379	0.9087	0.9550	0.8311	0.9504	0.9481	0.9439	0.8885	0.9538	0.7887	0.9408	0.9515	0.9277
2	0.8980	0.9545	0.8398	0.9456	0.9495	0.9425	0.9099	0.9578	0.8493	0.9389	0.9531	0.9409	0.9057	0.9546	0.8383	0.9459	0.9556	0.9443	0.8836	0.9440	0.8165	0.9334	0.9483	0.9300
3	0.9038	0.9594	0.8393	0.9315	0.9536	0.9382	0.9028	0.9571	0.8440	0.9352	0.9557	0.9493	0.8982	0.9533	0.8403	0.9488	0.9555	0.9356	0.8898	0.9464	0.8031	0.9328	0.9469	0.9296
4	0.9071	0.9517	0.8407	0.9397	0.9481	0.9280	0.9164	0.9575	0.8450	0.9505	0.9533	0.9411	0.9110	0.9557	0.8348	0.9397	0.9567	0.9408	0.8896	0.9478	0.7813	0.9349	0.9561	0.9319
5	0.9022	0.9536	0.8431	0.9453	0.9552	0.9323	0.9076	0.9531	0.8433	0.9372	0.9531	0.9440	0.8939	0.9576	0.8248	0.9355	0.9476	0.9354	0.8827	0.9431	0.7891	0.9279	0.9508	0.9285
6	0.9037	0.9548	0.8397	0.9342	0.9553	0.9367	0.9076	0.9535	0.8547	0.9373	0.9532	0.9411	0.8950	0.9574	0.8388	0.9455	0.9489	0.9426	0.8991	0.9444	0.8025	0.9182	0.9530	0.9289
7	0.9130	0.9570	0.8449	0.9346	0.9445	0.9393	0.9115	0.9514	0.8566	0.9379	0.9512	0.9408	0.9093	0.9586	0.8568	0.9369	0.9530	0.9426	0.8986	0.9483	0.7891	0.9243	0.9424	0.9244
8	0.9181	0.9574	0.8332	0.9470	0.9502	0.9356	0.9039	0.9604	0.8423	0.9464	0.9477	0.9452	0.9003	0.9585	0.8380	0.9483	0.9579	0.9387	0.8850	0.9481	0.8033	0.9332	0.9511	0.9229
9	0.9124	0.9487	0.8344	0.9348	0.9521	0.9360	0.9084	0.9542	0.8642	0.9485	0.9582	0.9426	0.8999	0.9532	0.8319	0.9450	0.9538	0.9416	0.8902	0.9460	0.7979	0.9273	0.9444	0.9333
10	0.9156	0.9542	0.8445	0.9471	0.9513	0.9418	0.9057	0.9532	0.8551	0.9478	0.9505	0.9467	0.8984	0.9584	0.8338	0.9465	0.9518	0.9434	0.8907	0.9539	0.7841	0.9366	0.9525	0.9356
Avg	0.9071	0.9546	0.8398	0.9401	0.9516	0.9372	0.9076	0.9556	0.8510	0.9427	0.9531	0.9429	0.9020	0.9562	0.8369	0.9443	0.9529	0.9409	0.8898	0.9476	0.7956	0.9309	0.9497	0.9293
Std	0.0074	0.0030	0.0039	0.0060	0.0037	0.0045	0.0044	0.0028	0.0073	0.0059	0.0029	0.0034	0.0062	0.0021	0.0084	0.0051	0.0037	0.0033	0.0056	0.0037	0.0109	0.0066	0.0042	0.0038

	Resnet+Spezialetti et al.						Resnext+Spezialetti et al.						MobileNet+Spezialetti et al.						EfficientNet+Spezialetti et al.					
Split	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC
1	0.9026	0.9572	0.8359	0.9435	0.9465	0.9371	0.9122	0.9549	0.8317	0.9469	0.9545	0.9399	0.9085	0.9564	0.8461	0.9334	0.9557	0.9399	0.8936	0.9454	0.7861	0.9280	0.9550	0.9299
2	0.9074	0.9536	0.8497	0.9413	0.9532	0.9424	0.9065	0.9565	0.8555	0.9508	0.9609	0.9440	0.9074	0.9608	0.8483	0.9503	0.9544	0.9404	0.8823	0.9443	0.7892	0.9246	0.9478	0.9373
3	0.9094	0.9611	0.8377	0.9422	0.9527	0.9382	0.9036	0.9541	0.8484	0.9311	0.9537	0.9369	0.9017	0.9590	0.8329	0.9443	0.9545	0.9383	0.8941	0.9495	0.8052	0.9336	0.9515	0.9335
4	0.9119	0.9573	0.8430	0.9326	0.9507	0.9347	0.9198	0.9600	0.8389	0.9278	0.9549	0.9392	0.9113	0.9549	0.8420	0.9462	0.9547	0.9416	0.8945	0.9491	0.7962	0.9445	0.9431	0.9331
5	0.9090	0.9555	0.8365	0.9389	0.9536	0.9335	0.8987	0.9614	0.8566	0.9394	0.9566	0.9395	0.9046	0.9582	0.8343	0.9489	0.9537	0.9406	0.9022	0.9462	0.8101	0.9483	0.9459	0.9250
6	0.9132	0.9544	0.8486	0.9367	0.9525	0.9397	0.9145	0.9577	0.8445	0.9409	0.9509	0.9423	0.9040	0.9622	0.8422	0.9483	0.9493	0.9406	0.9037	0.9492	0.8088	0.9442	0.9572	0.9297
7	0.9099	0.9517	0.8535	0.9328	0.9422	0.9375	0.9135	0.9564	0.8661	0.9427	0.9568	0.9354	0.9041	0.9605	0.8390	0.9534	0.9582	0.9356	0.8904	0.9503	0.8029	0.9329	0.9503	0.9313
8	0.9106	0.9595	0.8349	0.9381	0.9505	0.9351	0.9056	0.9581	0.8385	0.9500	0.9528	0.9451	0.9155	0.9593	0.8318	0.9544	0.9600	0.9372	0.9049	0.9520	0.7933	0.9265	0.9479	0.9220
9	0.9157	0.9496	0.8357	0.9267	0.9530	0.9395	0.9018	0.9575	0.8479	0.9535	0.9585	0.9391	0.9066	0.9575	0.8431	0.9537	0.9562	0.9335	0.8912	0.9531	0.8012	0.9367	0.9491	0.9294
10	0.8926	0.9576	0.8517	0.9373	0.9587	0.9311	0.9166	0.9555	0.8463	0.9337	0.9569	0.9281	0.9124	0.9618	0.8357	0.9418	0.9530	0.9389	0.8891	0.9507	0.7952	0.9411	0.9515	0.9362
Avg	0.9082	0.9558	0.8427	0.9370	0.9514	0.9369	0.9093	0.9572	0.8474	0.9417	0.9556	0.9389	0.9076	0.9591	0.8396	0.9475	0.9550	0.9387	0.8946	0.9490	0.7988	0.9360	0.9499	0.9307
Std	0.0065	0.0035	0.0074	0.0051	0.0044	0.0033	0.0070	0.0023	0.0100	0.0088	0.0029	0.0048	0.0043	0.0024	0.0057	0.0065	0.0029	0.0025	0.0071	0.0028	0.0081	0.0083	0.0042	0.0047

	Resnet+LEAD-PN						Resnext+LEAD-PN						MobileNet+LEAD-PN						EfficientNet+LEAD-PN					
Split	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC	RF	SVC	GNB	MLP	LR	kSVC
1	0.9042	0.9500	0.6971	0.9435	0.9476	0.9340	0.8973	0.9586	0.6923	0.9531	0.9521	0.9396	0.9014	0.9516	0.6821	0.9468	0.9535	0.9263	0.8873	0.9400	0.6534	0.9431	0.9411	0.9260
2	0.9009	0.9466	0.6998	0.9340	0.9607	0.9320	0.9080	0.9544	0.7012	0.9459	0.9584	0.9338	0.9076	0.9556	0.6883	0.9466	0.9527	0.9420	0.8816	0.9424	0.6529	0.9361	0.9515	0.9145
3	0.9056	0.9460	0.6923	0.9365	0.9577	0.9269	0.9116	0.9533	0.6749	0.9497	0.9638	0.9335	0.9129	0.9503	0.6687	0.9512	0.9522	0.9270	0.8849	0.9406	0.6412	0.9383	0.9542	0.9265
4	0.9040	0.9490	0.6857	0.9303	0.9542	0.9364	0.9143	0.9538	0.6990	0.9564	0.9503	0.9403	0.8949	0.9480	0.6711	0.9361	0.9518	0.9431	0.8947	0.9446	0.6364	0.9388	0.9452	0.9281
5	0.9061	0.9464	0.6836	0.9452	0.9522	0.9351	0.9084	0.9527	0.7096	0.9400	0.9565	0.9330	0.8991	0.9492	0.6755	0.9389	0.9513	0.9442	0.8841	0.9454	0.6416	0.9332	0.9502	0.9250
6	0.9015	0.9405	0.7015	0.9365	0.9507	0.9327	0.9063	0.9531	0.6994	0.9526	0.9599	0.9392	0.9091	0.9498	0.6682	0.9417	0.9503	0.9467	0.8955	0.9462	0.6539	0.9013	0.9462	0.9324
7	0.9035	0.9470	0.6844	0.9441	0.9491	0.9381	0.9183	0.9480	0.6985	0.9136	0.9524	0.9331	0.9057	0.9536	0.6689	0.8885	0.9526	0.9410	0.8961	0.9433	0.6511	0.9369	0.9461	0.9242
8	0.9115	0.9475	0.6983	0.9458	0.9511	0.9377	0.9095	0.9499	0.6988	0.9449	0.9549	0.9319	0.9100	0.9471	0.6726	0.9465	0.9533	0.9448	0.8937	0.9432	0.6409	0.9357	0.9451	0.9244
9	0.9071	0.9478	0.7022	0.9452	0.9507	0.9365	0.9096	0.9501	0.6946	0.9477	0.9561	0.9372	0.9139	0.9508	0.6863	0.9398	0.9551	0.9334	0.8907	0.9502	0.6342	0.9338	0.9462	0.9221
10	0.9071	0.9491	0.6882	0.9392	0.9536	0.9265	0.9137	0.9505	0.6888	0.9524	0.9547	0.9314	0.9103	0.9485	0.6682	0.9428	0.9596	0.9350	0.8847	0.9467	0.6411	0.9332	0.9503	0.9222
Avg	0.9052	0.9470	0.6933	0.9400	0.9528	0.9336	0.9097	0.9524	0.9456	0.9456	0.9559	0.9353	0.9065	0.9505	0.6750	0.9379	0.9532	0.9384	0.8893	0.9443	0.6447	0.9330	0.9476	0.9245
Std	0.0031	0.0026	0.0073	0.0055	0.0040	0.0041	0.0056	0.0092	#REF!	0.0122	0.0040	0.0034	0.0062	0.0026	0.0078	0.0179	0.0026	0.0074	0.0054	0.0030	0.0074	0.0115	0.0038	0.0046