A Study of 3D Point Cloud Features for Shape Retrieval

Hoang Justin Lev

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Hoang Justin LEV

Thèse dirigée par **Mounir MOKHTARI**, Maître de conférences, Institut Mines Télécom et
codirigée par **Joo-Hwee LIM**, Associate Professor, ASTAR et **Nizar OUARTI**, Associate Professor, Sorbonne UPMC
préparée au sein du Laboratoire **Image and Pervasive Access Lab**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Étude des propriétés des nuages de points 3D pour reconnaissance de forme

A Study of 3D Point Cloud Features for Shape Retrieval

Thèse soutenue publiquement le **11 mai 2020**
devant le jury composé de :

Monsieur JEAN-MARIE BONNIN
PROFESSEUR, IMT ATLANTIQUE BRETAGNE-PAYS DE LA LOIRE, Président du jury
Monsieur VINCENT CHARVILLAT
PROFESSEUR, INP TOULOUSE - ENSEEIHT, Rapporteur
Monsieur NICOLAS LOMENIE
MAITRE DE CONFERENCES HDR, UNIVERSITE PARIS-DESCARTES (PARIS-V), Rapporteur
Monsieur JOO-HWEE LIM
Principal Scientist, A*STAR - SINGAPORE, Examinateur
Monsieur NIZAR OUARTI
MAITRE DE CONFERENCES, SORBONNE UNIVERSITE - PARIS, Examinateur
Monsieur MOUNIR MOKHTARI
PROFESSEUR, INSTITUT MINES TELECOM, Directeur de thèse

UNIVERSITÉ GRENOBLE ALPES

Mathematics, Information Sciences and Technologies, and

Computer Science (MSTII) Doctoral School.

THESIS

in preparation for the title of Doctor of Philosophy

A STUDY OF 3D POINT CLOUD FEATURES FOR SHAPE RETRIEVAL

JUSTIN LEV

Thesis directed by Mounir Mokhtari

Prepared at Image & Perversive Access Lab (IPAL)

Co-directed by Nizar Ouarti and Joo-Hwee Lim

Que diable allait-il faire dans cette galère ?

Les Fourberies de Scapin, Molière.

Acknowledgments

To begin with I would like to express my gratitude to the members of my jury for taking the time to read and give feedback on my thesis. Vincent Charvillat, Nicolas Lomeni, I am grateful that you accepted to be my reviewers. Thanks to my thesis director, Mounir Mokhtari and my both supervisors, Nizar Ouarti and Joo-Hwee Lim. Joo-Hwee, I really appreciated your advice, your patience and your professionalism.

Je tiens bien sûr à remercier toutes les autres personnes, pas officiellement impliquées dans cette thèse, et qui ont pourtant été essentielles à a son aboutissement. La route était droite, mais la pente était bien forte. C'est vous qui m'avez le plus écouté, le plus supporté (et cela, dans les deux sens du terme). Merci les Amis. Anne C., pour avoir toujours été là. Ysaline, Lilian, les anciens colocs. Chloé K. (la visite du Tate était *passionnante*). Sophie, Pierre, nos repas à 4000 calories me manquent mais les pâtes sans gluten un peu moins. Céline, Anne V., Valentin ("le meilleur tonton du monde") et tous les autres.

Merci aux personnes de Singapour à commencer par les incontournables stagiaires dont l'IPAL n'a pas manqué. En particulier Nico, 10 fois plus intelligent que tout le monde. Alex, l'acolyte des soirées. Et Bastien, le Parrain de toute cette mafia. Vous avez été très efficace en tant que chaudière de la *student room* dans laquelle on avait la chance d'être tous entassés. Merci aux compagnons de galère: Thomas, Matthieu et Ana. Enfin, Ariane et Flo, merci pour votre gentillesse et votre générosité. Ariane : je te suis infiniment redevable d'avoir pris le temps de relire tous mes articles et surtout de m'avoir supporté sur tant de kilomètres. Certains n'ont pas tenu 35 bornes

avant de s'évanouir quand d'autres ont carrément simulé la blessure de surentraînement après un petit aller-retour à Stadium... Sans vous, les mardis auraient été moins sportifs, moins gourmands, mais surtout bien moins enflammés sans les parties de Pérudo, Monopoly deal ou Catan.

J'ai bien sûr un mot pour ma famille : Maman, Papa, Chloé, cette fois c'est terminé, je ne peux plus continuer les études, il va falloir (vraiment) commencer à travailler. Je vous remercie de m'avoir toujours laisser faire mes choix. Je sais que me voir partir à l'autre bout de Paris, puis à l'autre bout de la France et enfin quasiment à l'autre bout du monde n'a pas été facile. Mais je vous rassure, on ne peut pas *encore* aller à l'autre bout du système solaire.

Enfin, mes derniers mots seront pour Yoyo : merci pour tout, la thèse et la vie à Singapour auraient été bien différentes sans toi.

Abstract

With the improvement and proliferation of 3D sensors, the price cut and enhancement of computational power, the usage of 3D data intensifies for the last few years. The 3D point cloud is one type amongst the others for 3D representation. This particularly representation is the direct output of sensors, accurate and simple. As a non-regular structure of unordered list of points, the analysis on point cloud is challenging and hence the recent usage only.

This PhD thesis focuses on the use of 3D point cloud representation for three-dimensional shape analysis. More particularly, the geometrical shape is studied through the curvature of the object. Descriptors marking out the distribution of the principal curvature is proposed: *Semantic Point Cloud* (SPC), and *Multi-Scale Principal Curvature Point Cloud* (MPC2). *Global Local Point Cloud* (GLPC) is another descriptor using the curvature but in combination with other features. These three descriptors are robust to typical 3D scan error like noisy data or occlusion. They outperform state-of-the-art algorithms in instance retrieval task with more than 90% of accuracy.

The thesis also studies deep learning algorithms on 3D point cloud which emerges during the three years of this PhD. The first approach tested, uses curvature-based descriptor as the input of a multi-layer perceptron network. Their accuracy cannot catch state-of-the-art performances. However, the study shows that ModelNet, the standard dataset for 3D shape classification is not a good picture of the reality. Indeed, the dataset does not reflect wealthness of the curvature of true objects scans.

Ultimately, a new neural network architecture is proposed. Inspired by the state-of-the-art deep learning network, *Multi-scale PointNet* computes the

feature on multiple scales and combines them all to describe an object. Still under development, the performances are still to be improved.

In summary, tackling the challenging use of 3D point clouds but also the quick evolution of the domain, the thesis contributes to the state-of-the-art in three major aspects: (i) Design of new algorithms, relying on geometrical curvature of the object for instance retrieval task. (ii) Study and exhibition of the need to build a new standard classification dataset with more realistic objects. (iii) Proposition of a new deep neural network for 3D point cloud analysis.

Résumé

Grâce à l'amélioration et la multiplication des capteurs 3D, la diminution des prix et l'augmentation des puissances de calculs, l'utilisation de donnée 3D s'est intensifiée ces dernières années. Les nuages de points 3D (*3D point cloud*) sont une des représentations possibles pour de telles données. Elle à l'avantage d'être simple et précise, ainsi que le résultat immédiat de la capture. En tant que structure non-régulière sous forme de liste de points, l'analyse des nuages de points est complexe d'où leur récente utilisation.

Cette thèse se concentre sur l'utilisation de nuages de points 3D pour une analyse tridimensionnelle de leur forme. La géométrie des nuages est plus particulièrement étudiée via les courbures des objets. Des descripteurs représentant la distribution des courbures principales sont proposés: *Semantic Point Cloud* (SPC) et *Multi-Scale Principal Curvature Point Cloud* (MPC2). *Global Local Point Cloud* (GLPC) est un autre descripteur basé sur les courbures mais en combinaison d'autres propriétés. Ces trois descripteurs sont robustes aux erreurs communes lors d'une capture 3D comme par exemple le bruit ou bien les occlusions. Leurs performances sont supérieures à ceux de l'état de l'art en ce qui concerne la reconnaissance d'instance avec plus de 90% de précision.

La thèse étudie également les récents algorithmes de deep learning qui concernent les nuages de points 3D qui sont apparus au cours de ces trois ans de thèse. Une première approche utilise des descripteurs basé sur les courbures en tant que données d'entrée pour un réseau de perceptron multicouche (MLP). Les résultats ne sont cependant pas au niveau de l'état de l'art mais cette étude montre que ModelNet, la base de données de référence pour la

classification d'objet 3D, n'est pas optimale. En effet, la base de données n'est pas une bonne représentation de la réalité en ne reflétant pas la richesse de courbures des objets réels.

Enfin, l'architecture d'un réseau neuronal artificiel est présenté. Inspiré par l'état de l'art en deep learning, *Multi-scale PointNet* détermine les propriétés d'un objet à différente échelle et les combine afin de le décrire. Encore en développement, le modèle requiert encore des ajustements pour obtenir des résultats concluants.

Pour résumer, en s'attaquant au problème complexe de l'utilisation des nuages de points 3D mais aussi à l'évolution rapide du domaine, la thèse contribue à l'état de l'art sur trois aspects majeurs: (i) L'élaboration de nouveaux algorithmes se basant sur les courbures géométrique des objets pour la reconnaissance d'instance. (ii) L'étude qui montre que la construction d'une nouvelle base de données plus réaliste est nécessaire pour correctement poursuivre les études dans le domaine. (iii) La proposition d'une nouvelle architecture de réseau de neurones artificiels pour l'analyse de nuage de points 3D.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

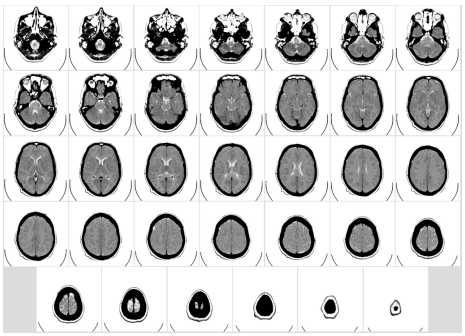
1.1.1 3D Data and Application

In the past few years, thanks to the enhancement of the sensors and computational power, 3D data became easier to handle. The acquisition system became smaller, more accurate and cheaper while the enhancement of computational power makes the processing time shorter. Therefore the use of 3D data representation steps up throughout many applications.

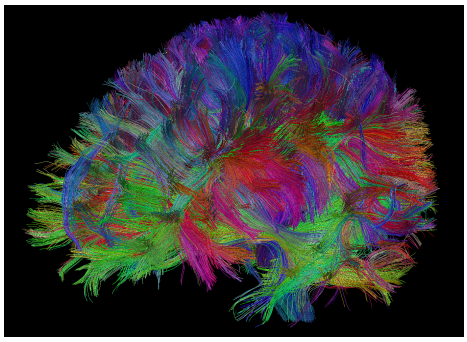
3D in Medical Usage

3D data are commonly used in medical fields. Indeed, some medical imaging technics offer 3D visualisations for the physicians. For instance: a computed tomography scan (CT scan) is a method which generates 3D images of the inside of an object from an extensive series of 2D X-ray images [55]. CT scan was introduced in the 1971s and has been used increasingly since then. The primary use is to scan the head, the lungs, the arterial and venous vessels (angiography) but also the heart or abdominal and pelvic region. A CT scan consists of a collection of 2D images, and each is the visualisation of the object for a given depth. Usually, a 2D acquisition is made every millimetre.

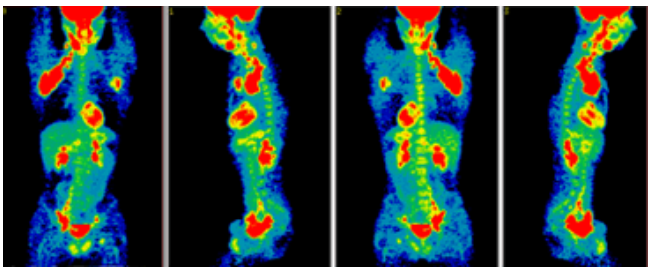
Another medical imaging technic which also came out in the 1970s is the



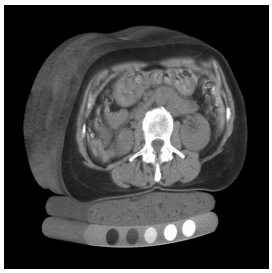
(a) Computed tomography (CT) scan of a human brain. A subset of the collection of 2D images is represented. Each image is the X-ray acquisition for one depth. Source: Department of Radiology, Uppsala University.



(b) 3D modelling of the nerve tracts using the data collected by diffusion-weighted magnetic resonance imaging (dMRI). Source: Martinos Center for Brain Imaging, Massachusetts General Hospital, Boston, Massachusetts.

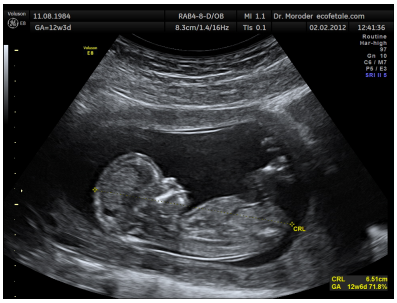


(c) Positron emission tomography (PET) scans. The tracer product can be seen in red. Source: Nuclear Medecine Department, Clinical Center, National Institutes of Health.



(d) 3D volumetric reconstruction of a Quantitative CT scan. Source: MindwaysCT QCT Pro brochure.

(e) 3D ultrasound of a 20 weeks fetus. Source: Commons.wikimedia.org



(f) 2D ultrasound of a 12 weeks fetus. The details are less visible on this 2D ultrasound than the 3D ultrasound. Source: Commons.wikimedia.org

Figure 1.1: 3D Visualisation in Medical Imaging. Examples of computed tomography, magnetic resonance image, positron emission tomography scans, 2D and 3D ultrasound.

magnetic resonance imaging (MRI) [22]. In this case, strong magnetic fields and radio waves are used to map the location of water and fat in the body. The magnetic field applied provokes the emission of a radio-frequency signal by the hydrogen atoms in the body which are then detected by antennas. The principal used are for neuroimaging (better visualization than a CT scan), cardiac MRI (as a complement to cardiac CT), spinal imaging and soft tissue tumours but also for liver and gastrointestinal exams and angiography.

Positron-emission tomography (PET) [6] is another 3D visualisation technique. PET allows visualisation of the metabolism of cells by using a tracer molecule. The gamma rays emitted by the tracer molecule is detected by depth layers, and a 3D model can be reconstructed. The primary use of PET is for oncology to detect cancer, heart disease by visualizing the areas of decreased blood flow and also neuroimaging.

3D ultrasound is also another medical 3D imaging [40]. Mainly known for the obstetrics application, 3D ultrasound is also used to visualize the heart (cardiology) or the blood vessels and arteries (vascular imaging). Because ultrasound can provide real-time visualisation, it can be used for surgical guidance or to see a fetus in motion (4D ultrasound scans). 3D ultrasound is an extension of the 2D ultrasound: the transducer (scanning device) is translated, tilt or rotate in order to get several views of the object.

These technics are the main tomography technics used in medical imaging. Tomography from Ancian Greek *tomos*, "slice, section" and *grapho*, "to write" consists in imaging the object by section [55]. Once the collection of two-dimensional images over, volume rendering techniques are used to generate a 3D volumetric grid model. Thanks to the regular acquisition pattern, each sampled value of the 2D image is matched to an opacity and a colour. The two values are then converted into RGBA (Red, Green, Blue, Alpha) components, and the volumetric grid is the result of this process on the whole 2D images collection. Current devices allow a high resolution, up to 800000 voxels.

The 3D medical imaging can represent medical data in more accurate details than usual 2D imaging. Allowing to get a better understanding, a better diagnosis and thus a better treatment. For instance, with a 3D imaging,

a surgeon can directly see a picture of the anatomy, there is no need to create a mental image of the patient anatomy. This limits the risk in during the operation and increases the efficiency of it.

However, one of the most promising usages of 3D medical imaging is the 3D printing of organs. Indeed, 3D scans can be used to print organ with the good size and shape to perfectly fit the patient's own. In the first time, 3D printers were used to produce low-cost and highly customizable prosthetic parts for patients [84, 85]. But they can also be used for the development of artificial hearts [24]. A silicone artificial heart can be created, using a 3D scan of the real heart to adjust its shape quickly and without additional cost.

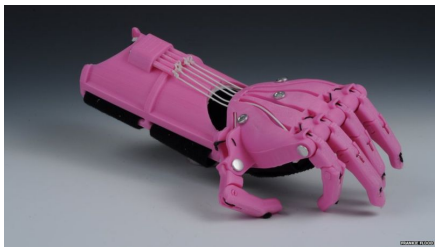
Nowadays, with advances in 3D printer, organs and tissues with blood vessels can even be printed [88]. The examples of usage are multiple: bone, heart valve, ear cartilage, cranium replacement or synthetic skin. Research for printing more complex organs like liver, kidney or lung tissue is also promising. All of them being a replica of the patient's own thanks to the use of 3D models. The main challenge is to make the organs viable, but using printed organ will solve the issue of transplant rejection and lack of organ donors.

Simply put, the use of 3D data makes the medicine better. Thanks to advances in 3D medical imaging physicians have additional tools and with better accuracy to diagnose patient. While, in the meantime, 3D printer improved and are now able to make organs which will be used for transplant in the foreseeable future.

Usage of 3D for Autonomous Vehicle

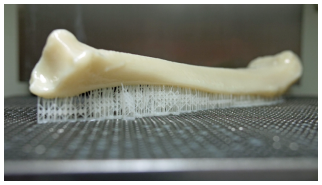
Autonomous vehicles are another example of 3D data usage. The 3D information for autonomous vehicle is vital and make it possible to get the distance from the environment objects. 3D data are also used by unmanned aerial vehicles to scan the ground.

These two utilizations are possible thanks to the use of Lidar, terrestrial Lidar for the self-driving cars and aerial for the unmanned aerial vehicle.

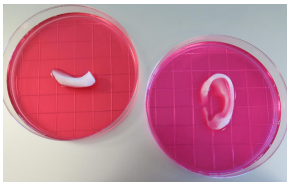


(a) Prosthetic hand. CAD software was used to design the hand and 3D printer to manufacture it. Source: [85]

(b) 3D printed artificial heart. Functional monoblock silicone heart with complex inner structure [24]. Source: ETH Zurich



(c) 3D printed bone model to allow the surgical team to see and study before the operation. A CT scan is used to create the 3D model, which is then printed. Source: Today's Medical Development



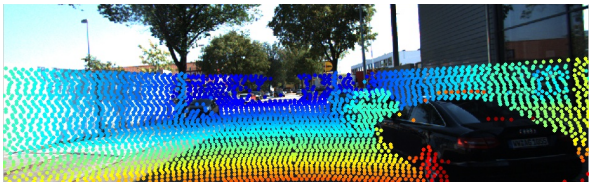
(d) 3D printed jaw bone and ear. Printed living tissue to replace injured or diseased tissue in patients [62] Source: Wake Forest Baptist Medical Center.

(e) 3D printed bionic ear. The electronic part would be able to enhance functionalities over human [81].

Figure 1.2: 3D printing for medical usage

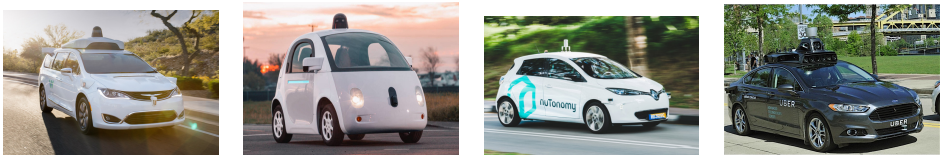
A Lidar from "Laser Imaging, Detection and Ranging" is a technology to measure distance by light lasers pulse. A laser pulse is emitted on the target and the time until the reception of the reflected pulse is measured, then a simple calculation returns the distance to the measured point. Hence the representation of the acquired data as a list of coordinate: the point cloud. Another type of Lidar used to measure speed, relies on the Doppler effect: the offset of frequency is measured, and the speed can be computed.

The research to build autonomous car relies on the use of multiple sensors, including critical Lidar sensors.



(a) Point cloud generated by the Lidar set on a car for KITTI dataset [42]. The coloured points are the projection of the point cloud on a 2D image.

(b) The recording car uses to create KITTI dataset [42]. The Lidar can be seen on top of the car.



(c) Examples of autonomous vehicles. In all model, the Lidar sensor can be seen on the roof of the car. From left to right, these prototypes are developed by Waymo, Waymo, Nutonomy and Uber.

Figure 1.3: Usage of 3D by Autonomous Vehicle.

3D in Entertainment

3D became popular for most people with the Kinect: an RGB-D camera produced in 2010 by Microsoft the home video game console Xbox 360. The accessory provides a new way to interact with the game and without a traditional game controller. For instance, in a car racing game, the user could pilot the car by mimic holding a virtual wheel in front of the camera: to steer left, the gamer would make the same move than if he was in its car. In a tennis game, the player would hold an invisible racket and hit the ball on the screen as a tennis player would do in the field.

The Kinect features a simple RGB camera, a depth sensor and some microphones and can provide a full-body 3D motion capture and facial recognition. The depth sensor uses infrared lasers with a CMOS sensor to capture the scene. The upgraded version for the Xbox One offers a better accuracy by using a wide-angle time-of-flight camera. The launch price was $150 only for an accurate RGB-D camera.

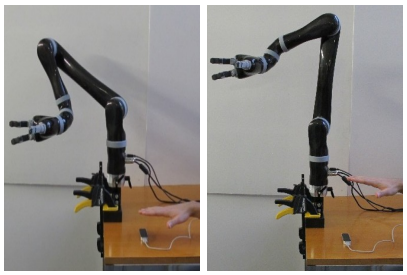
Thanks to the low-cost price and the release of an SDK (software development kit) by Microsoft, researchers were able to develop their own application, hence the popularity of the RGB-D in research. The applications then multiplied rapidly. For instance, the Kinect is used for 3D mapping of indoor environments for robot navigation or manipulation (Henry et al. [54]), but human detection is also possible (Xia et al. [133]). Another usage is the possibility to use hand gestures to browse with Google Chrome with DepthJS (Zinman et al. [146]). 3D tracking of hand articulations is also a research topic (Oikonomidis et al. [92]).

The Leap Motion is another device which is particularly adapted for hand tracking The technology is simple, with two infrared cameras and three infrared LEDs. Devote to hand tracking, the observation area is about 1 meter. The data acquired are then a deep-map of the hand, the Leap Motion software (not public) then uses models to interpret the motion of the hand and return a 3D position. By using the device, it is possible to navigate on a website, use a pinch gesture to zoom in or even manipulate virtual 3D object [90].

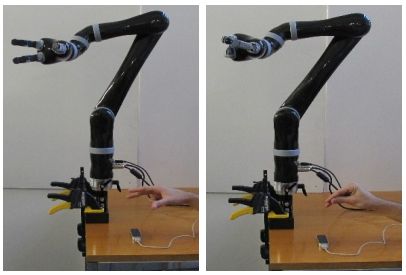
Just like Microsoft Kinect, the popularity of the Leap Motion is due to the release of a development kit. For instance, Bassily et al. [7] try to enslave the motion of a robotic arm to the user's hand movement. While Jin et al. [58] use the two Leap Motion sensors to perform a tele-operative manipulation on a controlled robot.

Another usage of the 3D in entertainment is the motion capture, which is the process of recording movements of people or objects. When the recording is more accurate and includes the facial expressions and hand, the term of performance capture is used. In video game development, motion capture is used to give a dash of more realism to virtual characters. For instance, movement of footballers is recorded and then implemented into the game to mimic their behaviours. For the most famous of them, their face is also recorded, giving the programmers the possibility to create a more realistic response. Motion capture also became popular in the cinema: movies like Avatar, The Lord of The Rings trilogy or King Kong would not be possible without. Indeed, the Na'vi, Gollum or King Kong characters are the result of

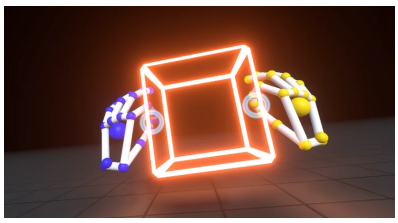
(a) RGB-D image captured by a Kinect. Left: RGB image. Right: Depth information, the darker, the closer. Illustration from [54]



(b) Control of the robotic arm's Z-direction with the Leap Motion controller. The higher the hand, the higher the robotic arm [7].



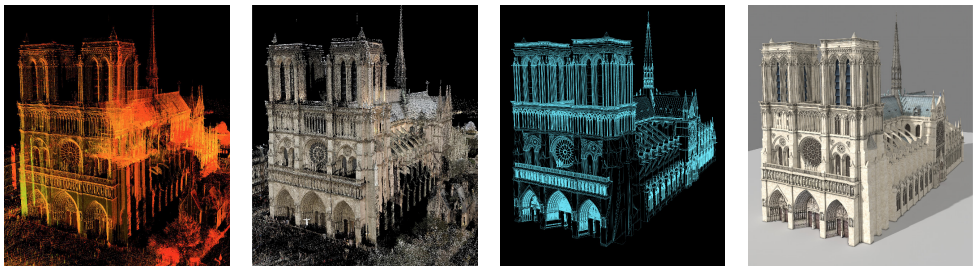
(c) Control of the robotic grasping with the Leap Motion controller. Opening or closing the hand ignite release or grasp operaton [7].



(d) Visualisation of Leap Motion interface. The device records the hands of the user and generates a 3D model. On the virtual interface, the user can move, expand, reduce or create new blocks [90].

(e) Motion capture of a fencer. The reconstructed skeleton overlays one of the videos used for the recording. Markers are represented by the white dots; the markers on the fencer are not clearly visible. Source: University of Delaware

Figure 1.4: Usage of 3D in entertainment.



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

Figure 1.5: 3D Point Cloud of Notre Dame de Paris and mesh generation of the virtual model [50]. (a): Raw 3D point cloud of the cathedral with more than a billion points. (b): Wrapping of photographies to the 3D point cloud, the points are coloured according to the images. (c): Computation of the mesh (planes and lines) model from the point cloud. (d): Virtual generated model of the cathedral.

performances capture of real actors.

The acquisition methods evolved, but the principle is to record the moving person with 2D images from multiple points of view and from them calculate the 3D position and position. The person has to wear reflective markers to triangulate the position, but markerless techniques are emerging.

Further usage of 3D data

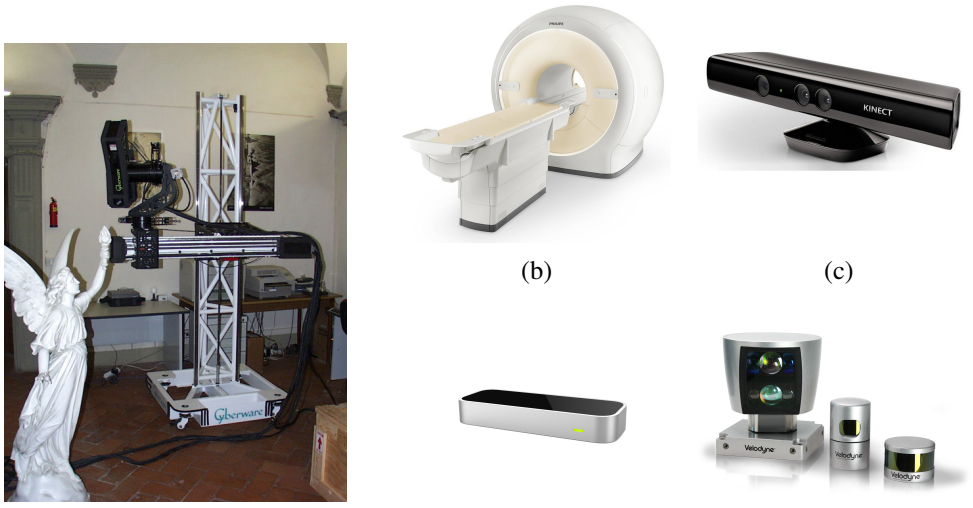
3D data is not limited to the usage previously given in the example. Lidar has multiples fields of application like topography of lands, measure of vehicles speed or even volume of mines. Lidar is even used in astronomy to measure the distance to the moon or the composition of other planets atmosphere. Finally, another exciting usage of Lidar is for archaeology and architecture: they can create a quick and accurate 3D model of archaeological sites. More recently this use of came to light with Notre-Dame de Paris fire. Indeed, in 2001, Andrew Tallon, an architectural historian scanned the cathedral (Hartigan Shea, Rachel [50]). The result is one billion data points with an accuracy of about 5 millimetres that will be undoubtedly helpful for the reconstruction.

Another usage for the 3D is for security application. A 3D scan of the fingerprint holds more information than a simple 2D image. Therefore, it is more difficult to hack the system. 3D face scanners are also used, the best example, even though little known, is the 3D face recognition of the iPhone X. The smartphone uses a sensor, TrueDepth, to get a scan of the user. The technology is similar to the Kinect but for a single usage purpose and a closer observation range: hardware and parameters are optimized. An infrared emitter projects a map of 30000 points on the user's face with a known pattern which is recorded by the dedicated infrared camera. The result is a grayscale depth map of the camera, which is analyzed, using deep learning algorithms to perform the face recognition. Apple claims an accuracy of 1 in a million compared to 1 in 50000 for their older system relying on 2D fingerprints.

1.1.2 3D Data Acquisition

Some methods of 3D data acquisition were presented previously. Multiple images of the same target for different depth can be used to reconstruct a 3D visualisation. A third dimension is added to each image, pixels becoming voxels (for volumetric pixel) and stacked. The result is a regular grid of voxel of the target; each voxel represents the colour value. This method of acquisition is for instance use in medical imaging [55, 22, 6].

Lasers can also be used to get a 3D model of the target, using the time-of-flight, a laser pulse is emitted and the time to receive the reflected pulse is measured, knowing the speed of light, it is easy to get the distance to the point. The process then repeated for many points on the target. A more straightforward technique also using a laser can be used instead: the triangulation. This time, the laser ray emitted, the contact point and the camera form a triangle. By knowing the distance between the camera and the laser emitter and the angle of reflexion, the distance to the object can be deducted. Time-of-flight technique is better suited for long-distance estimation as the accuracy is directly related to the measure of the time-of-flight of the laser. The laser pulse travelling at the light speed, this time-of-flight is very short and hence the



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

Figure 1.6: Example of 3D Sensor Devices. (a): "Handheld" scanner used by Levoy et al. [76] to scan large statue. The scanner is fixed on a solid structure to help the acquisition. (b): MRI scanner, the device size is approximately 3 by 5 by 2 meters and cost between $150000 and $500000. (c): Microsoft Kinect camera. (d): Leap Motion Sensor, the size of the sensor is 8 by 3 by 1 centimetre, the price is $80. (e): Velodyne Lidar, three models are shown, from left to right: HDL-64E, HDL-32E, PUCK. The HDL-64E can be seen on top of the cars on Figure 1.3, the range of usage is up to 120 meters and the system record up to 2.2 million points per second. The PUCK cost $8000 and the HDL-64E $100000 which is a price drop compared to the competitor.

use in autonomous vehicle applications [41, 42] or city mapping [50]. The accuracy is about a few millimetres On the other hand, triangulation is more accurate and can operate at short range only. Triangulation is for instance used to get a 3D model of sculptures thanks to the tens of micrometres accuracy [76]. This type of technique returns a point cloud, which is a list of unordered coordinates of points. Structured light is another technique using laser: a pattern of light is projected on the target and looks deformed because of the shape of the object [43]. This is recorded from different points of view, and the 3D model of the target can be reconstructed. Apple uses this technique for their face recognition cameras.

One last example of methods to record 3D data is the use of RGB-D camera like the Microsoft Kinect. The use of infrared light and infrared camera give a depth map which can be overlain to the RGB image from the classic camera. The Leap Motion also uses a similar system. On the second version of the Kinect, a time-of-flight camera is used instead of the infrared emitter and sensor combo. As a result, the visual field is 60% wider, and the range decreases to 1 meter with an accuracy of about 1 centimetre at 2 meters [63], the Leap Motion has a mean accuracy of 0.7 millimetres [131].

Because the first technique to get 3D data required specific devices, were time-consuming and expensive, the creation of a 3D dataset was not easy. Just like digital cameras and even more smartphones were a turning point for the acquisition of 2D images, the apparition of cheaper sensor like the Microsoft Kinect popularizes and eases the dataset creation process. Hence the late availability of 3D dataset.

1.1.3 Distinctiveness of 3D data

As seen, 3D data has multiple usages in many application fields. The oncoming question would be why the 3D data was so long to be explored when 2D data are used and studied for more than 5 decades.

Since the use of Lena by Roberts [101] in 1962 to illustrate image processing algorithm, much progress has been made. Many algorithms have been proposed to perform image processing and study 2D images, and the aim is to improve their quality or extract information. Many derive from signal processing field, filtering is, for instance, a fundamental operation which relied on the Fourier representation of the image. A low pass filter smooths the image while a highpass highlights the edges in the image, lowpass and highpass Butterworth are examples of such kind of filter, but more straightforward spacial filters exist and operate directly on the original image. The knowledge of the edges is critical to extract information: from the edges, it is possible to segment the image, generating clusters which can be classified. This example is over-simplified: the edges may not be easy to detect, the

segmentation may not cluster proper region of interest and resulting clusters are still to be identified.

Prewitt [94], Sobel [117] and Canny [17] filter are the most common edges detector, Canny filter relies on the gradient of the image but applies a Gaussian filter at first, and the edges are the result of thresholding. Hough transform is a technique to find the lines in the image and can be used as a complement of the edges detection filter.

Image segmentation is another complex problem: the pixels with the same "feature" need to be clustered. The segmentation can rely on the edge (e.g. Canny [17] or Sobel [117]), the region (e.g. Watershed [127]), the colour (e.g. Otsu [93]) or a combination of the three.

Finally, the identification step, more generally, the pattern recognition is unquestionably the hardest to do. Indeed, common patterns in the images have to be found and interpreted, which makes the link with artificial intelligence. In supervised learning, a set of labelled "training" images are used to learn the pattern associated with each label. The generalization of these patterns to unknown data ascertains the global performance of the recognition. For instance, if the images represent letters and the task is to recognize them, the training dataset label would be the content of the images, i.e. the letters. The pattern recognition algorithm then needs to find the typical representation between images representing the letter "a", "b", 'c", etc. and so, despite the variation in the representation of the same letter. Another example is the recognition of fingerprint: in this case, a fingerprint has to be retrieved into a fingerprint database. Features of the query fingerprint need to be found by the algorithm and compared with the database. The first example is a classification problem, while the second involves instance retrieval. In an instance retrieval problem, an unknown pattern has to be found in a known dataset which holds the query pattern. The query pattern is usually not strictly the same than the one in the database because of geometrical transformation, noise or occlusion. Each one of the elements of the database is also considered as unique and different from the others. The classification problem is slightly different but a lot more complex to resolve: this time, the query pattern has

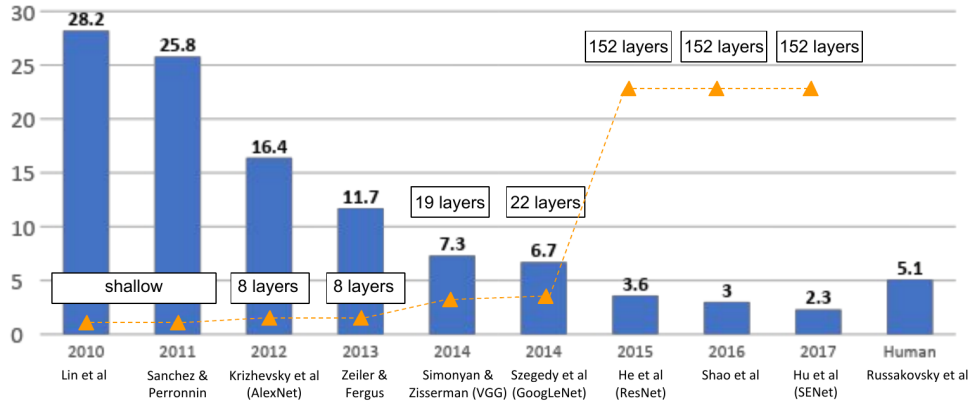


Figure 1.7: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners. The histogram represents the error rate. Versions before 2012 do not use deep learning algorithm. In 2012, start of the "deep learning revolution" with a gain of nearly 10 points compared to the previous year. Since then, the error rate keeps decreasing by using deeper networks, up to 152 layers in 2017 compared to the simple 8 layers architecture of Krizhevsky et al. [68].

to be classified into a category, the exact pattern is not present in the training dataset. In this case, the training dataset holds multiple elements that describe the same category. The objective is to find common features to all elements of the category and then to look for these feature in the unknown pattern to classify it. Algorithms for pattern recognition are multiple: linear or quadratic classifier, decision tree, K-nearest-neighbor, naive Bayes classifier, support vector machines or neural networks.

In fact, the use of deep learning methods like neural networks and its variants like convolutional neural networks, long short-term memory or others is a groundbreaking event in image processing (Figure 1.7). Indeed, in 2012, Krizhevsky et al. implemented convolutional neural networks on GPU and won large-scale ImageNet competition by a significant margin over all other shallow algorithm.

The use of the deep learning follow an exponential curve in image processing with good performance in classification [68], segmentation [80], detection and localization [109] or scene understanding [38]. This development is not only limited to computer vision application, hence the qualification of a new

era for AI since 2012.

Deep learning emerges later for 3D data processing. The reasons are multiples, with the hardware limitation but also of the distinctiveness of 3D data. While creating a 2D picture became really easy with the steep rise of image sensors, the creation of 3D data required a more complex process.

As mentioned and unlike 2D data, the representation of 3D data is not limited to one. When a 2D image is represented as 2D pixel arrays, the 3D object can be represented by a mesh, voxels, a depth map or a point cloud.

A mesh is a collection of polygon describing the object. The polygons are described by the vertices, the edges and the faces, which are usually triangle. The voxel representation is based on a quantification of the space into a regular 3D cube grid. This representation is the closest to the 2D pixel representation with a regular and ordered data structure. A point cloud representation is quite the opposite: the data consist in an unordered list of points.

This diversity leads to different classes of analysis algorithms and approaches, each one processing on only one kind of representation. This segmentation of the research does not help the development of 3D data analysis as much as the lack of data. But most of all, the limiting factor was the hardware capability. Indeed, when a resolution of 512×512 is regularly used for image processing, the resolution of the 3D voxel representation with the same quantity of information would be $8 \times 8 \times 8$ only. To study a grid of $128 \times 128 \times 128$, the 2D image "equivalent" resolution would be 2048×1024, if the grid is $512 \times 512 \times 512$ then the corresponding image size would be 16384×8192. The memory and computing capability required is enormous. On the other hand, the point cloud representation is not regular: the points are not set according to a regular grid. Yet, image processing relies on the regular pixel representation, and it makes sense that 3D data processing is inspired by that. Hence the very recent use of 3D point cloud.

1.1.4 The Point Cloud

The point cloud representation of 3D data is particularly interesting for several reasons. First of all, the point cloud is the closest representation to raw sensor data as well as straightforward. Indeed, a point cloud is just an unstructured list of points coordinate, the immediate output of the sensor.

Another reason is the non-mandatory of pre-processing the data. This step is time-consuming and required much computational power, hence the will to bypass it. The pre-processing can also be a cause of loss of accuracy; for instance, the point cloud is converted to a voxel grid: the data are quantified.

The simplicity of the representation is also the reason why there are few works on point cloud until recently. The analysis is complicated because of the non-regular localisation of the points: because of this, a direct extension of 2D deep learning algorithms is not straightforward. Moreover, the set of points being unordered, the algorithms have to be robust to permutation in the data. Ultimately, the variable number of point in a point cloud is another obstacle. This number depends on the size of the object and the wanted accuracy. A point cloud can hold between a few thousand to several hundreds of thousands of points, yet a deep learning network has a set input vector length. These reasons explain the late emerging by Qi et al. [96] of 3D point cloud deep learning paradigm.

To illustrate this complexity, the scan of a simple plane can be taken as an example. In a mesh representation, only 4 points and 1 face is enough to describe the plane. In a volumetric representation, the plane is represented by an alignment of voxels in two directions. In a point cloud representation, the plane is represented by many points belonging to the plane, but without any relationship between them.

1.2 Research Goal

Considering the increasing usage of 3D sensors and three-dimensional data in many applications, the thesis explores new methods to analyze them. More specifically, the research aims to create algorithms to find features of a

3D object. Because the 3D scan of an object depicts only the shape and not the colour, these features only rely on the geometrical shape of the object.

Moreover, the research focuses on the 3D point cloud representation as an expanding and straightforward representation which coincides with the beginning of this PhD. The 3D point cloud is also a way to bypass time-consuming pre-processing steps required with mesh or voxel representation.

Ultimately, the emerging of deep learning algorithm opens the research on 3D object analysis to task like classification. Before that, handcrafts algorithms were not accurate enough for such tasks and could to do little more than retrieval. Hence the two directions of study during the thesis: one that focuses more on the retrieval of objects and the second on the classification of three-dimensional objects.

1.3 Thesis Contribution

This thesis present the following main contributions:

Semantic Point Cloud Descriptor (SPC) SPC is a geometric descriptor that describes a 3D object. It relies only on a point cloud, making some complex preprocessing like mesh generation or voxelization useless. The algorithm is inspired by the human perception of an object. Indeed, by touching an object and without seeing it, a human being can already have a good idea of what it is. The touch only gives the shape and curvature of the object, hence the idea to use the curvature to describe an object. The descriptor takes the shape of a histogram of the principal curvature. SPC descriptor can perform instance retrieval, being robust to noise and occlusion.

Multi-scale Semantic Point Cloud Descriptor (MPC2) MPC2 is an enhancement of SPC that use several observation scales to describe the objects. The use of multiple scales bypass some problems encountered with the mono-scale version (SPC). MPC2 is also able to perform instance retrieval with overall better performances than SPC.

Robustness Study Here we evaluate the performances of new Deep Learning algorithms for imperfect scanned point clouds. We compare the results to algorithms that performed well for these cases.

In-deep study of the 3D objects dataset ModelNet, the standard and most extensive dataset for 3D classification is studied. The results of the tests show that ModelNet does not provide a realistic representation of reals objects. Indeed, the CAD 3D objects constituting the dataset are composed of a combination of plane, simplifying the representation but removing the wealthiness of curve of the objects.

Multi-scale PointNet Multi-scale PointNet is a proposal of a new convolutional network architecture for 3D point cloud classification. It relies on existing algorithms but adds a notion of scale in the study. For a given scale, the point cloud is grouped by region, and a model is applied to get the region feature property. For each scale, only one region is selected, describing the most singular area for this scale. In the end, the feature of each scale is processed together to get a global feature for an object.

1.4 Thesis Outline

The thesis is divided into two parts that follow the evolution of the field during the last three years.

Part I focuses on the study of 3D descriptor prior to the advent of deep learning networks using raw point cloud, hence this part is more related to the problem of instance retrieval.

The **Part II** explores the different solutions offered with deep learning networks on 3D point cloud and deal with the problem of classification of 3D object dataset.

The chapters are organized as follow: the **Chapter 2** is a study of the different 3D descriptors and of the geometrical properties of 3D object.

In **Chapter 3**, algorithms for the instance retrieval problem are tested. We describe a new algorithm named Principal Curvature Point Cloud Descriptor. We present the different multi-scale variations of the latter and perform benchmarking.

Chapter 4 presents the classification problem and the different algorithms using deep learning with 3D data and more specifically with 3D point cloud.

In **Chapter 5**, algorithms are tested for the classification of 3D object. In a first time, we try to use algorithms that rely on previously computed descriptors. Then present and test a new architecture of neural network working on raw 3D point clouds.

Finally, in the **Chapter 6**, we summarize the thesis ang provide some directions for future research.

Part I

Instance Retrieval of 3D Point Cloud

Introduction

This part describes the research on 3D data based on geometrical features and before the emerging of deep learning for 3D data. The algorithms presented here are designed for handcraft descriptors and rely mostly on the geometrical properties of the 3D object. As explained previously, the research also faced a lack of 3D data, hence most of the research focuses on finding features of objects. These features are then used to do instance retrieval, for instance.

Given a known dataset, referred to the *reference dataset* and an unknown object of the dataset, the instance retrieval problem involves finding which object of the reference dataset it is. The unknown object is usually an altered version of the objects of the dataset: with lower quality, with noise or only partially visible.

A study of the existing algorithms is presented. The mathematical background on which our algorithm relies is also explained. Then the three algorithms we proposed are described and tested on two different sets of data. The first experiment tests the robustness of these algorithms to common recording errors. The second checks the performance with deformable 3D objects.

Chapter 2

3D Descriptor and Geometrical Curvature

In this chapter, the researches related to the instance retrieval of 3D models are presented along with the mathematical background required to understand our algorithm.

2.1 3D Descriptors

The favoured approach focuses on local properties of the model by computing *local descriptors*. They can achieve good robustness against occlusion although noisy data and wrong choice of scale can affect the performance [46].

The local descriptor of a point p depends on its local neighbourhood properties: position with respect to the normal, angle between the vectors, etc. There are two kinds of descriptors: *signature*, which is specific to each point and *histogram* which captures summary statistics. A signature descriptor is highly discriminative, but small deformation or perturbation on the studied surface might output a different signature for the same vertice. On the other hand, a histogram descriptor gathers and partitions the different characteristics of the vertice with respect to its neighbour, and is not specific to a vertice. Two different points can have the same histogram descriptor but not the same signature.

Once the type of descriptor selected, the choice of the metric to compare the descriptor is another essential parameters. Hetzel et al. [56] studied the usual comparing distance for descriptor: Euclidean distance, Chi-Square distance, Kullback-Leibler divergence, etc.

$$d_{\text{Manhattan}}(\mathbf{X}, \mathbf{Y}) = \sum_i |x_i - y_i| \tag{2.1a}$$

$$d_{\text{Euclidean}}(\mathbf{X}, \mathbf{Y}) = \left(\sum_i (x_i - y_i)^2 \right)^{1/2} \tag{2.1b}$$

$$d_{\chi_2}(\mathbf{X}, \mathbf{Y}) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i} \tag{2.1c}$$

$$d_{KL}(\mathbf{X}, \mathbf{Y}) = \sum_i (x_i - y_i) \log \left(\frac{x_i}{y_i} \right) \tag{2.1d}$$

In the equations in 2.1, $\mathbf{X}$ and $\mathbf{Y}$ are the descriptor of the two objects that are compared. The values x_i and y_i are the value of the descriptors when represented as a vector. For instance, if the descriptor is a histogram, they are the value of the i-th bin, if it is a signature, then they are the i-th component. The Manhattan and Euclidean distance are more adapted for a signature descriptor. Indeed, each component is supposed to be independent to the other in this representation. On the other hand, χ_2 distance and the Kullback-Leibler divergence are chosen for histogram descriptor. For histogram distributions, the value in one bin is correlated to the one of the nearby.

d_{χ_2} and d_{KL} are not distances in the formal mathematical definition. For instance, d_{KL} is not symmetric ie. $d_{KL}(\mathbf{X}, \mathbf{Y}) \neq d_{KL}(\mathbf{Y}, \mathbf{X})$ and there is some definition issue when there is one of the bin is empty ie. x_i and/or y_i is equal to 0. Section 3.1.1 shows how to bypass this problem.

The distance between two models is computed by averaging the distances between each local descriptor of the first with the closest amongst the descriptors of the second. Hence one main drawback of local descriptor methods is the time complexity of descriptor computation and distance matching. Indeed, as a descriptor can be computed for each vertex of the 3D model, subsampling

has been proposed to reduce this number. However, instead of uniform downsampling, only points with specific features, called *keypoints*, are extracted from the point cloud. Keypoints are points which are selected because of their local properties like the curvature or variation of the surface –i.e. derivative–. The algorithm must be repeatable with respect to the point of view variation, resampling, occlusion or noise. A neighbourhood radius should be chosen for the keypoint detection although there are some adaptative-scale detectors. ISS (Zhong [145]), KSR (Shah et al. [111]), Harris 3D (Sipiran and Bustos [116]) or 3D SURF (Knopp et al. [65]) are exemples of algorithm.The last two are for instance inspired by their 2D version: Harris (Harris and Stephens [49]) and SURF (Bay et al. [8]).

2.1.1 Spin Image

Spin Image by Johnson and Hebert [60] is one of the first 3D local descriptor algorithms. For each point P, its normal vector $\mathbf{n}$ is considered as the local reference axis for the description of points $\{Q_i\}_i$ of its neighbourhood. Two parameters are computed for each Q_i:

(i) the radial distance α_i,

$$\alpha_i = \left(\|\mathbf{PQ_i}\|^2 - (\mathbf{n} \cdot \mathbf{PQ_i})^2 \right)^{\frac{1}{2}}$$

(ii) the signed distance β_i,

$$\beta_i = (\mathbf{n} \cdot \mathbf{PQ_i})^2.$$

The pairs $\{(\alpha_i, \beta_i)\}_i$ are then projected in the 2 dimensions plane oriented by (α, β). Finally, this plane is discretized and the points $\{(\alpha_i, \beta_i)\}_i$ are accumulated in a 2D histogram: the spin image. The algorithm is resistant to occlusion and invariant to rigid transformation but sensitive to the sampling; the data must be uniformly sampled.

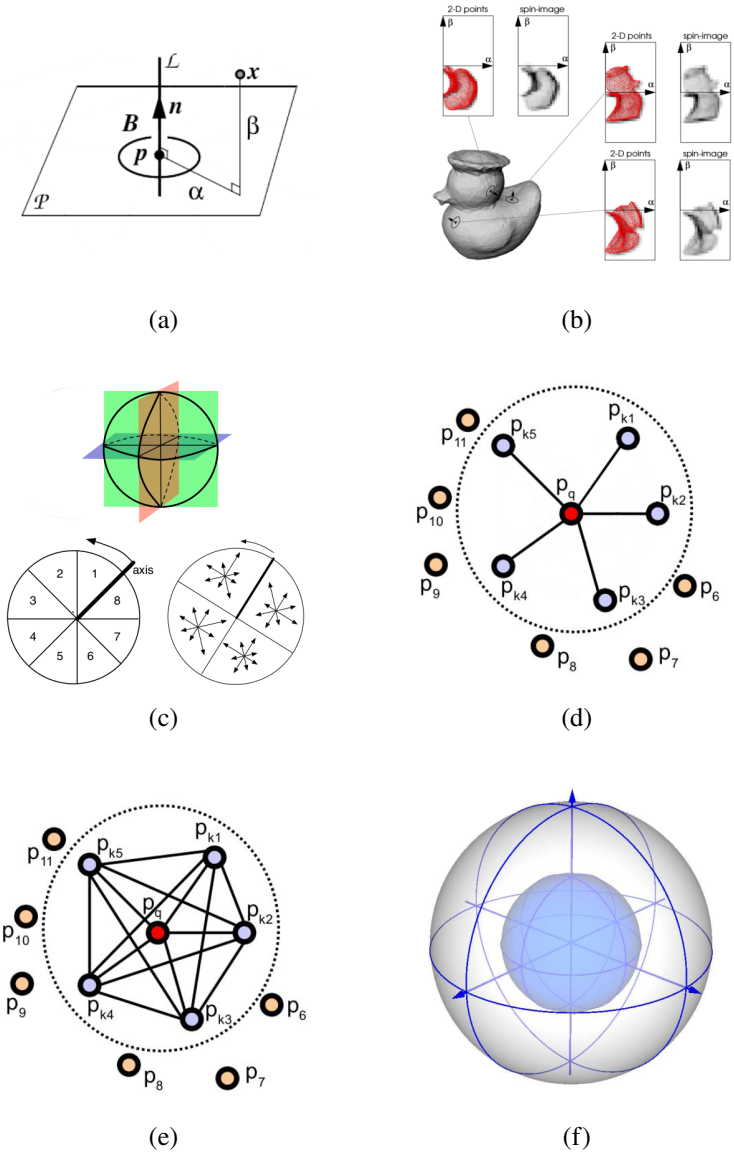


Figure 2.1: Illustration of 3D descriptors algorithms. (a): Spin Image parameters. (b): Spin Image descriptor with (α, β) plane. (c): MeshHOG, top: LRF planes, bottom-left: example of polar coordinate system with 8 slices, bottom-right: visualisation of the histogram regions with 4 polar slices and 8 orientation slices in each. Illustration of FPFH, PFH and SHOT methods. (d): FPFH, visualisation of all pair used by the algorithm. (e): PFH, visualisation of all pair used by the algorithm. (f): SHOT structure of the signature histogram. Note: Illustrations from [60, 143, 105, 104, 106], edited for clarity.

2.1.2 Mesh Histograms of Oriented Gradients

Mesh Histograms of Oriented Gradients or *MeshHOG* by Zaharescu et al. [143] relies on the normal of the point P to construct a local reference frame (LRF). A scalar function f on the mesh is also defined.

$$f : \mathbb{R}^3 \rightarrow \mathbb{R}$$

The descriptor consisted in a histogram computed thanks to ∇f, the gradient of f on the neighbourhood $\{Q_i\}_i$ of P.

For each point Q_i, $\nabla f\,(Q_i)$ is projected on the three orthonormal plane of the LRF. Each plane has been divided into 4 polar regions and 8 sub-regions are generated according to the orientation of $\nabla f\,(P)$ for each of them. Finally, the three histograms of each plane are concatenated to get the MeshHOG descriptor. The algorithm is effective for rigid and non-rigid deformations; however, it relies on mesh structure.

2.1.3 Fast Point Feature Histograms

FPFH for *Fast Point Feature Histograms* provided by Rusu et al. [105] is a descriptor relying on pair relationship between P and it local neighbourhood $\{Q_i\}_i$.

For each Q_i, the vector $\mathbf{r_i} = \mathbf{PQ_i}$ is computed, then measurements are calculated between $\mathbf{n}$ and $\mathbf{r_i}$. These measurements, using angles between the two vectors and their distance are accumulated into a histogram: FPFH descriptor.

FPFH is an enhancement version of PFH [104] motivated by the computational efficiency of the latter. In PFH, the number of bins in the histogram is higher and more measurements are computed. However, the main alteration is in the way the pairs of points are generated. Indeed, this time the pair of points on which the measures are computed is not only between P and Q_i but between all pair in the neighbour, i.e. Q_i and Q_j (with Q_i possibly being P). The number of pairs being significantly smaller in FPFH, the processing time is also even if the algorithm is still slow.

Table 2.1: 3D descriptor for 3D shape analysis

Algorithm		Data type	Properties.
Deep Shape	[37]	Mesh	Robust to deformation.
PointNet	[96]	Point Cloud	Deep learning architecture on point cloud.
Spin Image	[60]	Point Cloud	Amongs first 3D recognition algorithm. Uniform sampling is compulsory.
MeshHOG	[143]	Mesh	Rely on mesh structure.
FPFH	[105]	Point Cloud	Fast version of PFH. Still slow.
SHOT	[106]	Point Cloud	Very fast features computation. Slow to match.

2.1.4 Signature of Histogram of Orientation

SHOT for *Signature of Histograms of Orientation* by Salti et al. [106] is an algorithm inspired by FPFH.

For each point P, an LRF is computed with the help of the normal vector $\mathbf{n}$ and the spherical space is divided into 32 volumes. A local histogram of 11 bins is computed for each of them. A point Q_i is accumulated into the bins according to the angle between its normal $\mathbf{n}_{Q_i}$ and $\mathbf{n}$. Finally, all locals histogram are concatenated into a 32×11 bins' histogram: SHOT descriptor. Being highly descriptive thanks to the local histograms, the authors classified the algorithm as a signature of histograms, hence the name. The algorithm is robust to noise and fast; however, it is sensitive to the density of the point cloud.

2.2 Geometrical Curvature

Intuitively, a curve is opposed to the flatness [1, 18]. And the curvature of an object measures how much it is bent. For instances in a Euclidean plane (two dimensions), a line is a one-dimensional object with a null curvature, and a circle is an object with a constant and positive curvature. In the case of a Euclidean space (3 dimensions), a plane is a two-dimensional object with a null curvature, and a sphere is another two-dimensional object with a constant and positive curvature.

2.2.1 Curvature of 2D plane curves

In the Euclidean 2D space, a curvature measures how much a curve deviates from a straight line. It is an evaluation of the variation of the direction of the tangent of the curve: the bigger the variation, the higher the curvature. To make it simpler, the curvature gives an idea of how much the handwheel must be turned to take a turn: moderate if the curve is small, sharply if it is large.

Formally, if Γ is a curve, twice continuously differentiable ie. Γ' is continuous and differentiable with Γ'' continuous, then Γ can be locally approximated by a parametric pair of functions γ (cf. Deschamps et al. [32]).

$$\forall t \in \mathbb{R} \quad \gamma(t) = (x(t), y(t)) \,.$$

x and y being both twice continuously differentiables. The parametrization is not unique and an origin can be chosen such that with s the *arc length*:

$$\forall s \in \mathbb{R} \quad \|\gamma'(s)\|^2 = x'(s)^2 + y'(s)^2 = 1.$$

Thereby, if M is the point of the curve such that $\gamma(s) = M$, then the curvature for the point M is κ, $\kappa(s) = \|\gamma''(s)\|$.

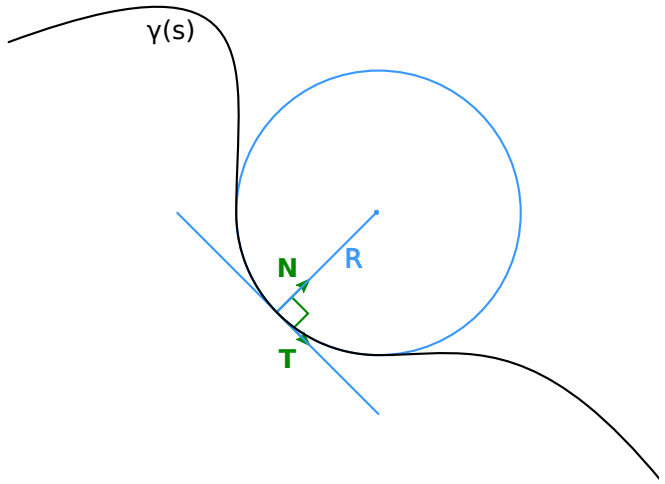


Figure 2.2: Visualisation of the curve γ (black), the Osculating circle (blue) and Frenet-Serret Frame (green). The osculating is the closest circle to the curve at this point. R is the radius of curvature, i.e. the radius of the osculating circle. For this point, $k \geq 0$ (Equation 2.2): the vector $\mathbf{T}$ will turn counterclockwise.

The following vectors and values can be defined:

$$\mathbf{T}(s) = \gamma'(s), \tag{2.2a}$$

$$\mathbf{T}'(s) = \gamma''(s) = k(s)\mathbf{N}(s), \tag{2.2b}$$

$$\kappa(s) = \|\mathbf{T}'(s)\| = \|\gamma''(s)\| = |k(s)|, \tag{2.2c}$$

$$R(s) = \frac{1}{\kappa(s)}. \tag{2.2d}$$

$(\mathbf{T}, \mathbf{N})$ is the Frenet-Serret frame, with $\mathbf{T}$ the unit vector tangent to the curve, pointing in the direction of the motion and $\mathbf{N}$ the normal unit vector. $\mathbf{T}$ and $\mathbf{N}$ can be seen as the direction of the velocity and the acceleration vector. κ is the curvature, k the oriented curvature and R is the radius of curvature.

R can be interpreted geometrically as the value of the radius of the osculating circle. For each point M of the curve, there is a unique circle which is the closest approximation of the curve in a neighbourhood of M, and this circle is the osculating circle. A large value of R means that the circle is big:

in a close region around M, the curve is nearly flat and the curvature κ is small.

Ultimately, k indicates the direction the vector $\mathbf{T}$ rotates along the curve γ. If $k > 0$ the rotation is counterclockwise, the curve turns in the direction of $\mathbf{N}$. Else, if $k < 0$ then the rotation is clockwise, the curve turns in the opposite direction.

Here another way to understand the curvature. If γ is the vector position of an object, $\mathbf{T}$ is the vector velocity and $\mathbf{T}'$ the vector acceleration. With $\kappa(s) = \|\mathbf{T}'\|$, κ is the value of the acceleration. κ indicate the speed, the vector velocity will vary, i.e. how fast the vector will rotate.

2.2.2 Curvature of 3D space curves

In the 3 dimensional space, the situation is not very different than the previous in a 2D plane. A parametrization function γ can be find, and $\mathbf{T}$ can be define in the same way:

$$\mathbf{T}(s) = \gamma'(s).$$

In the same way, $\mathbf{N}$ is still define such as:

$$\mathbf{T}'(s) = \gamma''(s) = k(s)\mathbf{N}(s).$$

The curvature κ is once again the magnitude of the acceleration vector γ''.

$$\kappa(s) = \|\gamma''(s)\|.$$

The Frenet-Serret frame can be completed with $\mathbf{B}$, $\mathbf{B} = \mathbf{T} \wedge \mathbf{N}$ the cross product of $\mathbf{T}$ and $\mathbf{N}$. The plane defined by $\mathbf{T}(s)$ and $\mathbf{N}(s)$ is the osculating plane to the curve at $\gamma(s)$. In this plane, the curve γ can be approximated at s by a circle of radius $R(s)$: the osculating circle to the curve with:

$$R(s) = \frac{1}{\kappa(s)}.$$

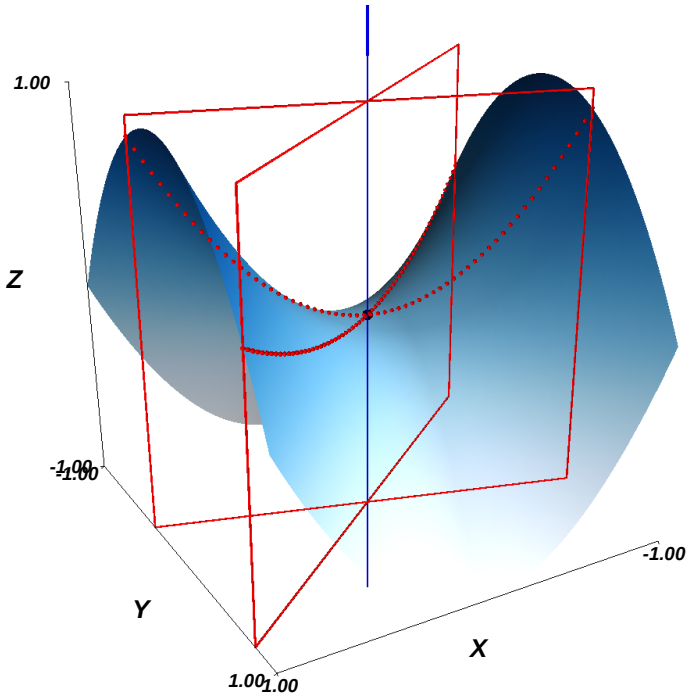


Figure 2.3: Example of 3D surface and curvature of curves on a 3D surface. In blue the surface of the hyperbolic paraboloid: there is no direction prioritised, a point on the surface can go in every direction. The normal line to the surface at $M(0,0,0)$ is drawn in blue, the line is directed by the normal vector $\mathbf{n}$ (oriented upward). Two normal planes are plotted in red, their normal section with the hyperbolic paraboloid is represented by the red dotted line. The normal curvatures are the value of the curvature of the red dotted lines.

2.2.3 Cuvature of curves on surfaces

Now, we focus on the curvature of a curve on a surface. The situation is different from the last two previous sections. Indeed, for a point M, there is "no direction" for the motion, its movement is not limited in a single direction (Figure 2.3).

For a point M on the surface, the unit vector normal to the surface is named $\mathbf{n}$. Then, a normal plane at M is the plane that includes the normal vector $\mathbf{n}$. For one point M, there is an infinity of normal planes which are all the rotation of the plane along the vector $\mathbf{n}$ (Figure 2.3). The intersection of

the surface with each normal plane defines a curve: the normal section. Also, the curvature of the normal section is called the normal curvature. Here, the curvature is defined as previously, i.e. as the inverse of the radius of the osculating circle. If the curvature is positive, then the curve turns in the direction of $\mathbf{n}$, else the curve turns in the opposite direction. For a point M, depending on the normal plane, the normal curvature will be different.

The *principal curvatures* at a point M are the maximum k_1 and minimum k_2 values of this curvature (Figure 2.4). If $k_1 \neq k_2$, then the direction of the two normal planes defining the principal curvature are the principal directions. Euler proved in 1760 already [36] that the principal directions are always perpendicular. If $k_1 = k_2$, all curvature in all direction at M are equal: the surface is a sphere in the neighbourhood of M.

In fact, the principal curvatures and the principal directions are the eigenvalues and eigenvector of the Weingarten endomorphism, the differential of the Gauss map. Because Weingarten endomorphism is symmetric, the spectral theorem applies.

The product of the principal curvatures, K is named the Gaussian curvature, and the average H is the mean curvature.

$$K = k_1 \cdot k_2, \tag{2.3a}$$

$$H = (k_1 + k_2)\,/2. \tag{2.3b}$$

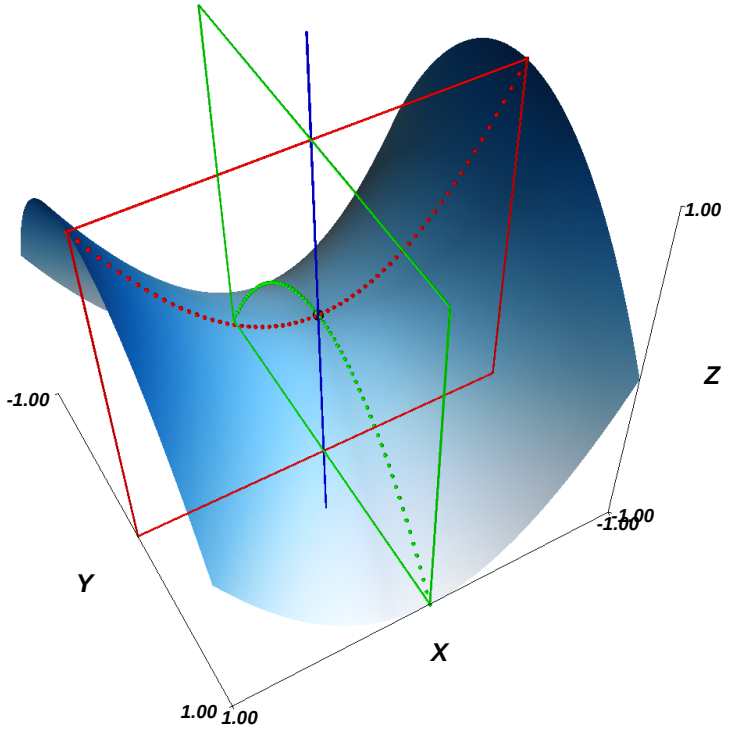


Figure 2.4: Visualisation of the Principal Curvature at a point $M(0,0,0)$ The principal directions are the two planes in red and green; these two planes are perpendicular. The red dotted line is the normal section with the most significant curvature value k_1, and the green dotted line is the one with the smallest curvature value k_2. In this example, the principal curvature are of opposite signe. $k_1 \geq 0$, the red curve "is going" in the same direction than the normal. $k_2 \leq 0$, the green curve "is going" in the oppotite direction.

Chapter 3

Curvature Based Descriptor

3.1 Curvature Based Descriptor

3.1.1 Principal Curvature Point Cloud Descriptor

The selected approach is based on a local measure of the curvature. In this study, these local pieces of information are aggregated to obtain a global idea of the object shape.

In this context, the local information is the curves and idea of the overall shape can be described as a histogram. By comparing a histogram with different recorded histograms, it is possible to recognise a given shape.

For a point cloud V with N points, $V = \{v_i\}_{i\in[1;N]}$, we proposed to apply the following transformation. We call S the surface underlying by V.

Preprocessing

To get more robust results, our first step consists of a normalisation of the point cloud V:

$$\forall i \in [1; N], \quad \begin{cases} \tilde{v_i} = \alpha \cdot \dfrac{v_i - \mu_V}{\sqrt{\sum_i \|v_i\|_2{}^2}} \\ \mu_V = \dfrac{1}{N} \cdot \displaystyle\sum_{i=1}^{N} v_i \end{cases} \tag{3.1}$$

Then, the barycenter of V is set at the origin and $\|V\|_\infty = 1$.

From now onward, V and $\{v_i\}_i$ refers to the centred and normalised data.

Neighbourhood selection

To compute the curvatures of each points v_i of V, we define for every one of them a neighbourhood B_i of size r_λ such as:

$$B_{i,\lambda} = B(v_i, r_\lambda) = \{p \in \mathbb{R}^3 \mid \|v_i - p\|_2 \leqslant r_\lambda\} \tag{3.2a}$$

$$\text{with :} \quad r_\lambda = \lambda \cdot \underbrace{\max_{i,j}(\|v_i - v_j\|_2)}_{M}, \quad \lambda \in [0; 1]. \tag{3.2b}$$

Thus, our neighbourhood radius is proportional to the maximum length M of the object.

For a given λ value, we gather in $V_{i,\lambda}$ all points belonging to both V and $B_{i,\lambda}$: $V_{i,\lambda} = V \cap B_{i,\lambda}$. If the number of points in $V_{i,\lambda}$ is under a threshold τ ($\tau = 6$), then the point is skipped. Indeed, the computation requires a minimum number of point which can affect the accuracy of the computation.

Local and adapted basis

In order to determine an adapted local basis of origin $v_{i,\lambda}$, another step is the PCA processing [19, 33] on the neighbourhood $V_{i,\lambda}$ which materialize

the surface $S_{i,\lambda}$. After a change of basis to this fitted one, called $\mathcal{B}_{\text{fit}}$, we can locally describe the surface $S_{i,\lambda}$ with:

$$z(x, y) = J_{B,n}(x, y) + O(\|(x, y)^{n+1}\|_2), \tag{3.3a}$$

$$\text{with} \quad J_{B,n}(x, y) = \sum_{p=1}^{n} \left(\sum_{q=0}^{p} B_{p-q,q} x^{p-q} y^{q} \right). \tag{3.3b}$$

In $\mathcal{B}_{\text{fit}}$ we select the polynomial function $J_{A,n}$ of degree n ($n = 2$ in our case) with a surface $Q_{i,\lambda}$ closest (in the least mean square sense) to $S_{i,\lambda}$:

$$\varepsilon = \sum_{p \in S_i} \left(J_{A,n}(p) - z(p) \right)^2 \tag{3.4}$$

Finally, a Taylor development of this approximation can provide the normal and the origin of this new Monge frame, $\mathcal{B}_{\text{Monge}}$, that is obtained thanks to the shape operator also called Weinggarten map. Then, it is possible to have access to the first and the second principal directions ($\overrightarrow{d_{i,\lambda,1}}$ and $\overrightarrow{d_{i,\lambda,2}}$). The first and the second principal curvature $k_{i,\lambda,1}$ and $k_{i,\lambda,2}$ ($k_{i,\lambda,1} \geqslant k_{i,\lambda,2}$) are also available for the point v_i.

Principal curvature histogram

The key novelty of our approach is to preserve the data related to $k_{i,\lambda,1}$ and $k_{i,\lambda,2}$. For this reason we did not combine them like in *Gaussian curvature* with

$$k_{gaussian,i,\lambda} = k_{i,\lambda,1} \cdot k_{i,\lambda,2} \,. \tag{3.5}$$

Each curvature is processed *independently*. In our case, the first and the second curvature are associated to a specific distributions $K_{\lambda,1} = \{k_{i,\lambda,1}\}_i$ and $K_{\lambda,2} = \{k_{i,\lambda,2}\}_i$, which provide the histograms $H_{\lambda,1}$ and $H_{\lambda,2}$ in an interval $I = [x_{min}; x_{max}]$. For points which curvature are beyond the extrema of I, their values are reduced to the closest border: x_{min} or x_{max}.. Also, the histograms are also normalised by the number of points used to compute the curvature.

Next step is the concatenation of $H_{\lambda,1}$ and $H_{\lambda,2}$: the resulting histogram

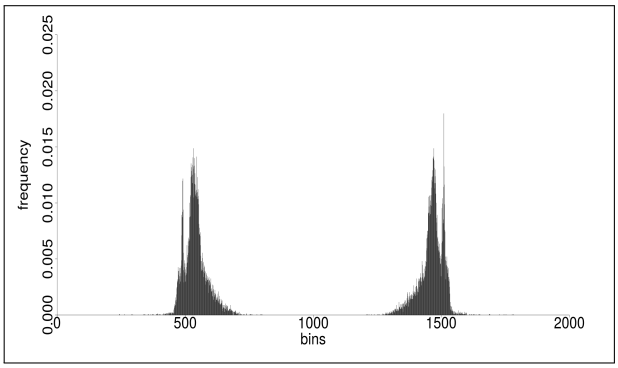


Figure 3.1: Principal curvature histogram of an object with a scale $\lambda = 6\%$

H_λ is constituted by *all the information related to the curvature* of the object, as its signature.

Retrieval test

We will take advantage of these histograms to compare objects by pairs. The estimation of the histograms similarity of two distinct objects A and B is based on Chi-2 distance:

$$d_{Chi2}(A, B) = \sum_j \frac{(H_A(j) - H_B(j))^2}{H_A(j) + H_B(j)} \tag{3.6}$$

To recognise an object, we compute the Chi-2 distance with each element in the reference dataset. We use a *Winner Take All* strategy so that the unknown object is assimilated to the one with the smallest distance.

However, before using the computation of the Chi-2 distance d_{Chi2}, we need to ensure the following:

1. $\sum_j H_A(j) = 1$ and $\sum_j H_B(j) = 1$, these conditions are obtained by normalisation.

2. Supposing that B is our reference object and A the query object (Eq. 3.6), it requires that:

$$\exists \eta \in \mathbb{R}^{+*} \quad / \quad \forall j \in [1; b] \quad \eta \leqslant H_B(j)$$

This implies each bin to hold a sufficient number of element to consider H_B as a valid distribution. To overcome this problem, we aggregate the bins j with it neighbour bin $j+1$ when $H_B(j) < \eta$. To guarantee the comparability (same number of bins), the same aggregation has to be done to the histogram H_A. After such transformation, one might note $d_{Chi2}(A,B) \neq d_{Chi2}(B,A)$: the final bins depend on the aggregate bins which modify with the chosen reference.

3.1.2 Multi-Scale Principal Curvature Point Cloud Descriptor

The premise is the same: use histogram curvature to characterize objects. However, we incorporate information from several scales.

1. As previously, we compute the H_λ, but we do not pick only one value for λ, we compute the curvature histogram for several of them. In our case $\lambda \in \{2, 4, ..., 10\}$%, so we get $H_{\lambda=2\%}$, $H_{\lambda=4\%}$, etc.

2. Once more, we concatenate the histograms: H is defined as the concatenation of all $\{H_{\lambda_i}\}_i$ sorted by increasing value of λ_i.

The use of a multi-scale approach provides several advantages. First of all, we avoid to select a value for λ: among the computed values, some inevitably fit for the object. Lowest scale gives pieces of information on –very– local curvatures like rough spot or noise while greater scales inform about the global flatness of the shape. With the multi-scale approach, the curvature of the shape at different scales can be seen, and each provides information for a different range.

Moreover, according to Wahl et al. [128], Kullback-Leibler (KL) divergence is a good candidate to measure how one histogram $\mathtt{A}$ diverges from a second one $\mathtt{R}$ and it may worth to test the difference with d_{χ^2}.

$$d_{KL}(\mathtt{A},\mathtt{R}) = \sum_j (H_{\mathtt{A}}(j) - H_{\mathtt{R}}(j)) \log \frac{H_{\mathtt{A}}(j)}{H_{\mathtt{R}}(j)} \tag{3.7}$$

The higher the d_{KL}, the more divergent are the histograms. Because of the division and the log (Eq. 3.7), bins of both $H_{\mathtt{A}}$ and $H_{\mathtt{R}}$ can not be empty: we need to preprocess histograms to compute d_{KL}. If R is the reference object and A the one to recognize, we reuse the same method than [74]. First, we remove/merge the empty bins in the reference histogram R and remove/merge the same bins in $H_{\mathtt{A}}$. Once performed, we are guaranteed there are no void bins in $H_{\mathtt{R}}$, but we know nothing about $H_{\mathtt{A}}$. We then apply the same algorithm after swapping the part of A and R. Formally, because of this processing, KL divergence is not a distance metric (as d_{χ^2}). It is not a symmetric relation: $d_{KL}(\mathtt{A},\mathtt{R}) \neq d_{KL}(\mathtt{R},\mathtt{A})$: A and R have different role.

By enhancing SPC to MPC2, we expect an improvement of the recognition rate, particularly in case of noise.

3.1.3 Global and Local Point Cloud Descriptor

The second approach aims to preserve the best of SPC and balances its weakness with the help of local descriptors algorithm hence the name: *Global and Local Point Cloud* descriptor. Preliminary tests show that SPC with bigger scale ($\lambda \geq 8\%$) performs very well with resampling and noise but poorly with occlusion. To adopt a global/local approach, we need to find a method which can reflect the local properties. For instance, the latter must provide excellent results in the case of occlusion. With $\lambda = 10\%$ for instance, we study the global properties of the object, its global shape and flatness. As a result, the recognition rate is low with the presence of occlusions, which can be detected using local properties. Finally, a scheme to combine the two must be chosen: which one should be selected in case of divergence of outcome between the local and the global features.

To select whether the local or the global output is selected, a score characterizing the reliability of the output of each method is used. This score is based on the distance between the object to find and the different objects of

the reference dataset.

Global and Local Distances

Both global and local descriptors offer a metric to determine which of the element in the `Reference dataset` is the closest. We designed $d_{\mathrm{G}}(\mathtt{A}, \mathtt{B})$, the distance between `A` and `B` with the global metric. In the same way $d_{\mathrm{L}}(\mathtt{A}, \mathtt{B})$ is the distance for the local metric. d_{G} is in fact the metric of SPC. As for d_{L}, we use the metric of the selected local feature.

Neighbours Similarities and Score

To determine an index of the reliability of the global or local feature, we interpret the closeness between 1-NN *First Nearest Neighbour* and 2-NN *Second Nearest Neighbour*. For this purpose, we compute a normalized score in $[0, 1]$ to highlight the difference between 1-NN and the other neighbours. Being close to 0 when `A` –the object to find– is close to 1-NN, this score is computed without any a priori on the validity of the 1-NN found previously. The distance between the studied object `A` and the 1-NN found is denoted as $d_{\min}(\mathtt{A})$ and $d_{\min}(\mathtt{A}) = d(\mathtt{A}, \mathrm{1NN})$. Then the score $s_{\mathtt{A},}(\mathtt{R})$ of a reference `R` is computed as follow:

$$\forall\, \mathtt{R} \in \{\mathtt{training\ dataset}\},\ s_{\mathtt{A}}(\mathtt{R}) = \frac{d(\mathtt{A}, \mathtt{R}) - d_{\min}(\mathtt{A})}{d(\mathtt{A}, \mathtt{R})} \tag{3.8}$$

This process is then applied to the algorithm with the global and the local feature. 1-GNN for *First Global Nearest Neighbour* and 1-LNN for *First Local Nearest Neighbour* are respectively used instead of 1-NN for the global and the local feature. In the same way, $s_{\mathtt{A},\ \mathrm{Global}}(\mathtt{R}) = s_{\mathtt{A},\mathrm{G}}(\mathtt{R})$ the global and $s_{\mathtt{A},\ \mathrm{Local}}(\mathtt{R}) = s_{\mathtt{A},\mathrm{L}}(\mathtt{R})$ the local score are defined.

$$\forall\, \mathtt{R} \in \{\mathtt{training\ dataset}\},\ \begin{cases} s_{\mathtt{A},\mathrm{G}}(\mathtt{R}) = \dfrac{d_{\mathrm{G}}(\mathtt{A}, \mathtt{R}) - d_{\mathrm{G,min}}(\mathtt{A})}{d_G(\mathtt{A}, \mathtt{R})} \\ s_{\mathtt{A},\mathrm{L}}(\mathtt{R}) = \dfrac{d_{\mathrm{L}}(\mathtt{A}, \mathtt{R}) - d_{\mathrm{L,min}}(\mathtt{A})}{d_L(\mathtt{A}, \mathtt{R})} \end{cases} \tag{3.9}$$

Note: $s_{\mathrm{A,G}}$ and $s_{\mathrm{A,L}}$ are computed with their own feature attribute ie. their proper $d_{\min}$ and metrics.

Decision Criterion

Once all these scores computed, a decision has to be taken: which of the global or local feature is the best? To find the answer, Fig. 3.2, the reliability of each feature is checked using $s_{\mathtt{A},\mathtt{G}}(\mathtt{R})$ and $s_{\mathtt{A},\mathtt{L}}(\mathtt{R})$. Because the chosen approach is to recognize an object with its global shape and use the very local details only to overcome global weakness, the first feature to check is the global. If $s_{\mathtt{A},\mathtt{G}}(R = 2\mathrm{GNN})$ is above the threshold τ_1 (more details below), the reliability is considered as good enough, the object `A` is matched with 1-GNN. Otherwise, the object as a whole is too different from the references: the global feature is not fit enough to determine the type of the object. In this case, the algorithm focus on local features and 1-LNN with $s_{\mathtt{A},\mathtt{L}}(R = 2\mathrm{LNN})$ are used. The reliability is once again checked: if $s_{\mathtt{A},\mathtt{L}}(R = 2\mathrm{LNN})$ is above the threshold τ_2 then the local feature is considered trustworthy enough, `A` is matched with 1-LNN. Else, neither of the global or the local feature outcome is reliable enough: the global feature is favoured and `A` is matched with 1-GNN.

Implementation

The local feature used is SHOT [106]. It has been selected for its fast computation time and robustness. Consequently, Euclidean distance is adopted for d_{L}.

The threshold τ_1 and τ_2 are chosen with empirical study after preliminary observations. Figure 3.3 plot the distribution of the scores of the two features; each histogram represents the density of objects with proper or wrong matching according to their score. τ_1 is determined based on figure 3.3a: as shown in, the score of mismatched objects are below a score of 0.225 whereas a score higher than 0.375 implies a proper matching. Hence the choice of 0.20 for τ_1. The second histogram, figure 3.3b, is related to the local feature

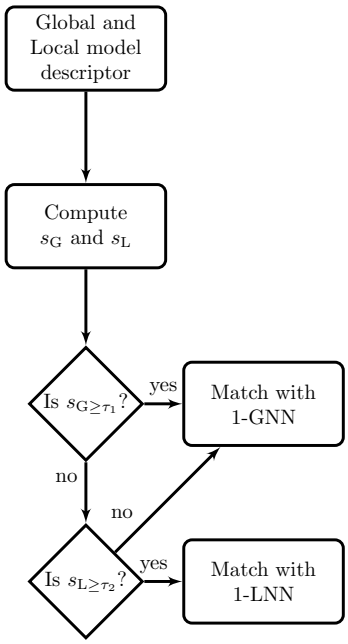
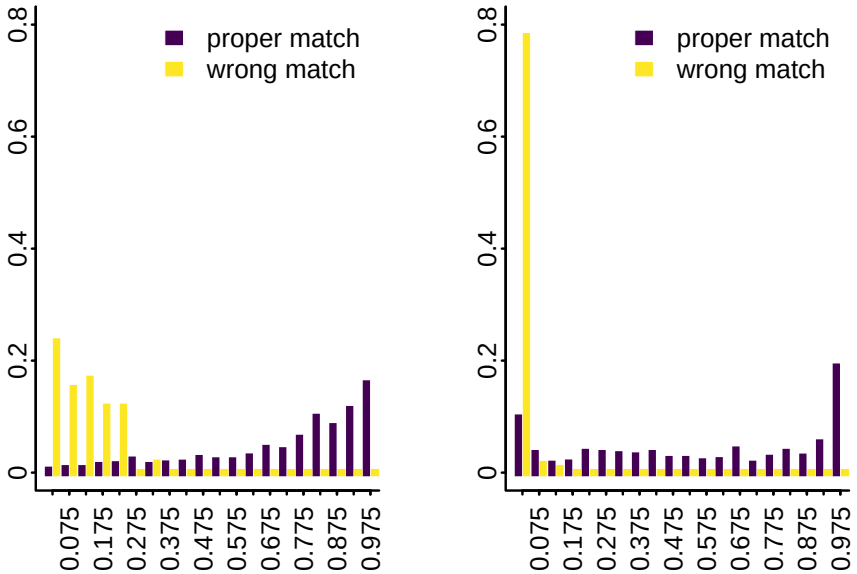


Figure 3.2: Decision algorithm of GLPC.



(a) Global (b) Local

Figure 3.3: Score distribution.

thus the value of τ_2. Its shape is very different from figure 3.3a: a yellow peak in the first bin is present, meaning 80% of the mismatched objects get a score below 0.05. In fact, there is no error in the matching if the score is above 0.125 hence the value of 0.05 for τ_2. On the other hand, one may note around 10% of the objects gets a meagre score but a proper matching with the local when the ratio is below 5% with the global feature. More generally, local feature scores for proper matching is more uniform than the global one, the ratio of proper matching objects with a low score is also higher. These thresholds can be tuned depending on the type of objects, but so far, these values fitted all studied dataset.

3.2 Robustness to Common Recording Errors

3.2.1 Dataset description

This dataset aims to test the robustness of the algorithm with respect to basic geometric transformation or data corruption. It is made up with standard 3D models dataset like: *The Stanford 3D Scanning Repository* [125, 28, 67], *A. Mian dataset* [86] and *McGill 3D Shape Benchmark* [114]. They can be seen in Figure 3.4 and represent animal toys, sculptures, decorative objects, etc. The meshes were removed, and the models centred and normalized. The dataset is, in the end, a high-resolution dataset. The objects are holding an average of 150000 with no perceptible noise. However, there is a significant gap between the smallest object, made by only 13000 points (`hand`) while the biggest hold more than 230000 (`angel`).

There are 16 references –`reference/training dataset`– and 624 generated objects –`test dataset`–. The latter is composed of alteration and degradation of the `training dataset`. For instance, a point of view variation is performed through translations and rotations while degradations are results of downsamplings, occlusions or noise corruptions. All the parameters for these transformations are randomly chosen. The downsampling follows a uniform distribution when partial objects are the result of the section of the

model name — number of points			
(a) angel	237020	(i) happy	250081
(b) armadillo	172976	(j) horse	48487
(c) bunny	35949	(k) lucy	250002
(d) cheff	176922	(l) parasaurolophus	184935
(e) chicken	135144	(m) rhino	79936
(f) chinese-dragon	249459	(n) statuette	250002
(g) dragon	180042	(o) teddy	15641
(h) hand	13091	(p) t-rex	176510

Figure 3.4: Reference objects (`training dataset`) of dataset 1.

original object by a plan, the resulting objects holding $\{10, 20, \ldots, 90\} \pm 5\%$ of the original number of points. Finally, the noise follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 0$ and $\sigma \in \{M \cdot \{0.1, 0.2, \ldots, 0.9, 1.0\}\%\}$ (Eq. 3.2).

3.2.2 Results

Table 3.1 and Figure 3.5 present the retrieval rate for the tested algorithms. The best overall retrieval rate is up to 98.3% with GLPC. SHOT performs slightly better than FPFH, but both results remain at similar values, approximately equal to 80%. SPC with $\lambda = 8\%$ gives very good results, up to 98.6% for downsampling and 100% for noisy data. With this parameter, the overall score is even better than MPC2 with 93.3% against 91.9%.

Multi-scale Methods

Figure 3.6 and Table 3.2 shows the different results for the multi-scales methods. We note good results with more than 90% of retrieval rate for all tested parameters and they are even up to 98.3% for GLPC. GLPC performs better than MPC2, which has an excellent rate for downsampling and slightly below but still good output ratio for occlusion and noisy data. Two different parameters are tested and presented for GLPC method, $\lambda = 6\%$ and $\lambda = 8\%$. $\lambda = 8\%$ presents better results than $\lambda = 6\%$, mostly because the global feature with a bigger parameter value offers better tolerance for noise. The drawback of big value is the loss in case of occlusion: 72.2% vs 78.4%. However, for both values, the matching rate is close to the best of the local and the global feature. For instance, for downsampled objects and $\lambda = 8\%$, the local feature offers 41.6% against 98.6% for global feature and 97.2% for GLPC, which is just slightly below the global. On the other hand, still for the same value of λ and for partial objects, the behaviour is inverted: local feature hits 95.1% compared to only 72.2% for the global and GLPC offers the same result as the local features. These results show GLPC algorithm can select the best between the global and the local features.

Table 3.1: Summary of results for dataset 1. Retrieval rate in % of the different tested algorithms. GLPC offers the best retrieval rate with 98.3%. SPC performs better than MPC2 but is less steady while SHOT and FPFH offer lower results.

	total	raw	translation	rotation	downsampling	occlusion	noise
# objects	624	16	80	80	144	144	160
GLPC (λ = 8%)	**98.3**	100	100	100	97.9	**95.1**	**100**
MPC2	91.9	100	100	100	93.7	86.1	86.8
SPC (λ = 8%)	93.3	100	100	100	**98.6**	72.2	**100**
SHOT (λ = 10%)	81.4	100	100	100	95.1	85.4	45.0
FPFH (λ = 5%)	78.6	100	91.2	100	85.1	88.1	45.0

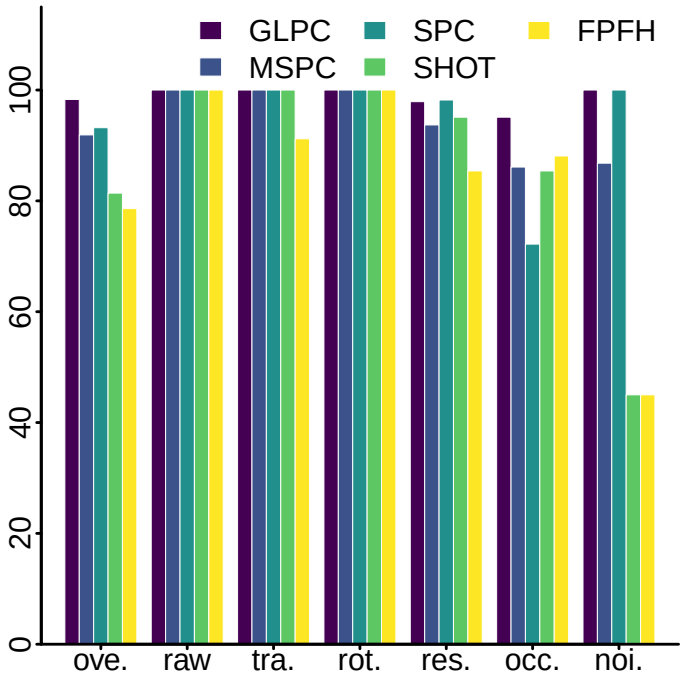
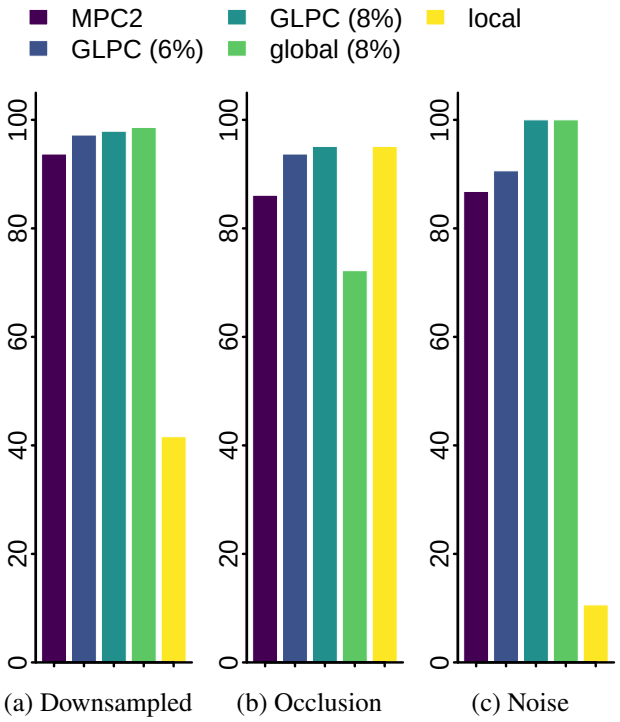


Figure 3.5: Summary of results for dataset 1. Retrieval rate histograms. SHOT and FPFH underperform in case of noisy data.

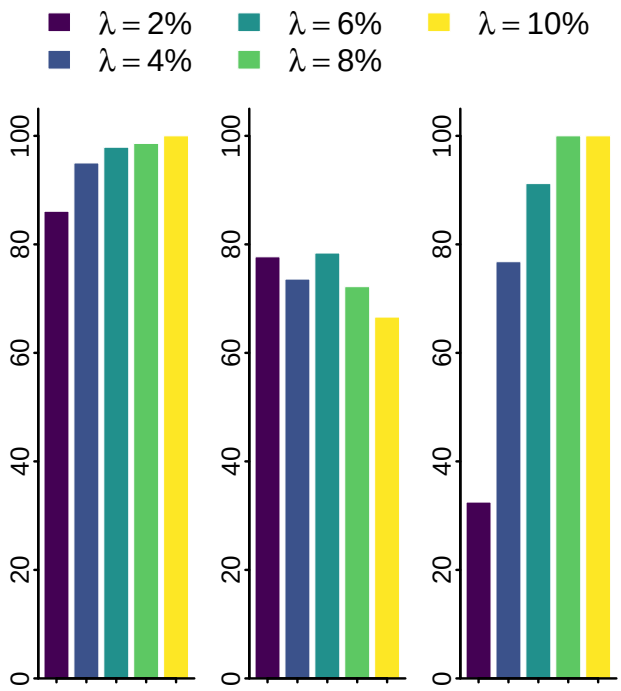


(a) Downsampled (b) Occlusion (c) Noise

Figure 3.6: Retrieval rate histogram for multi-scale methods (dataset 1).

Table 3.2: Retrieval rate for multi-scale methods (dataset 1). GLPC (8%) performs better than MPC2 and GLPC (6%). GLPC is always closed to the best between both local and global feature.

	total	*raw*	*translation*	*rotation*	*downsampling*	*occlusion*	*noise*
MPC2	91.9	100	100	100	93.7	86.1	86.8
GLPC 6%	95.5	100	100	100	97.2	93.7	90.6
GLPC 8%	**98.3**	100	100	100	**97.9**	**95.1**	**100**
glo 6%	92.3	100	100	100	97.9	78.4	91.2
glo 8%	93.3	100	100	100	98.6	72.2	100
loc	60.5	100	97.5	87.5	41.6	95.1	10.6



(a) Downsampled (b) Occlusion (c) Noise

Figure 3.7: Retrieval rate histogram for SPC (dataset 1).

SPC Method

In this part, we study results with SPC algorithm. A look at Figure 3.7 and Table 3.3 reveals an improvement of performance when the scale get bigger. For instance with $\lambda = 2\%$ (resp. $\lambda = 8\%$), SPC has an overall retrieval rate of 73.3% (resp. 93.3%), 86.1% (resp 97.9%) for downsampled object, 77.7% (resp. 72.2%) for partial object and 32.5% (resp. 100%) for noisy data. The bigger the scale, the better the result for downsampling and noise. Indeed, such perturbations affect more the local curvature and less the global. On the other hand, for partial objects, the behaviour is the opposite, small values offer better results. This is not surprising since an occlusion is only a local degradation of the object and SPC as a global descriptor cannot manage such deformations very well.

These results confirm the motivation about the use of SPC as part of a

Table 3.3: Retrieval rate for SPC (dataset 1). The performances improve when the scale is bigger for downsampled and noisy objects. For occlusion, the retrieval rate decrease after $\lambda = 6\%$.

	total	raw	translation	rotation	downsampling	occlusion	noise
$\lambda = 2\%$	73.8	100	96.2	100	86.1	77.7	32.5
$\lambda = 4\%$	86.8	100	100	100	95.1	73.6	76.8
$\lambda = 6\%$	92.3	100	100	100	97.9	**78.4**	91.2
$\lambda = 8\%$	**93.3**	100	100	100	98.6	72.2	**100**
$\lambda = 10\%$	92.3	100	100	100	**100**	66.6	**100**

multi-scale method. Mixing the smaller and bigger scale with a local and global feature can offer excellent results for all deformations cases.

SHOT Results

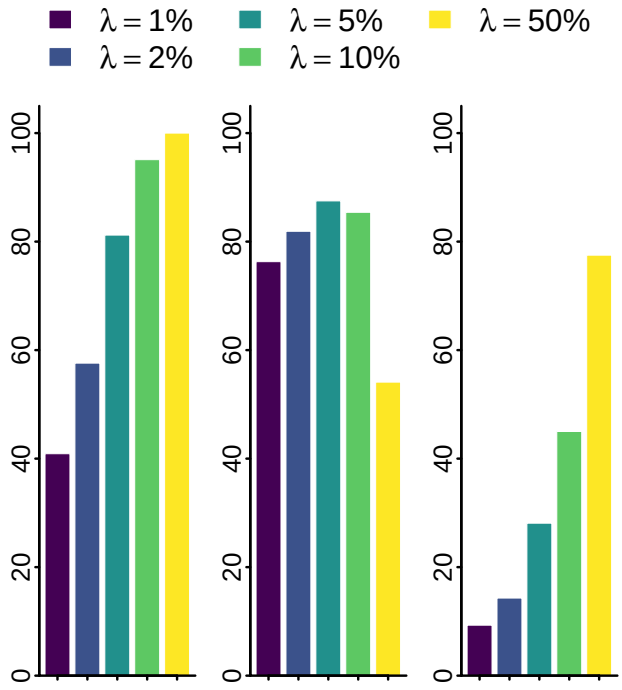
Figure 3.8 and Table 3.4 presents SHOT results for several radius size. In the same way than SPC, we notice when there is downsampling (Figure 3.8a) or noise (Figure 3.8c), the performance increases with the size of the radius. In cases of occlusion (Figure 3.8b), the retrieval rate increase before being less efficient when the studied size is too large. The best average value with good results for all transformation seems to be for 10%. However, the best result for occlusion is got with a radius size of 5%

3.3 Validation with Deformable Objects

3.3.1 Deformable Object

Dataset 2 Description

The aim of the dataset 2 is to study the robustness of the algorithm for deformable objects. For this purpose, we created a dataset made by object

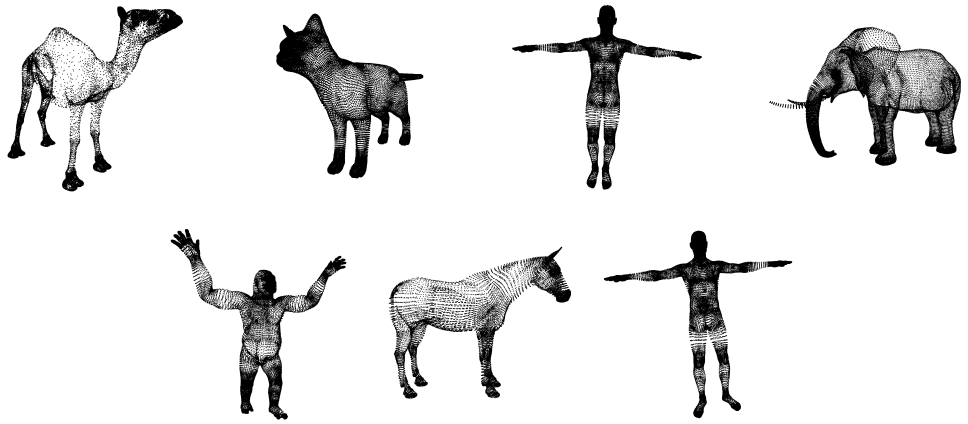


(a) Downsampled (b) Occlusion (c) Noise

Figure 3.8: Retrieval rate histogram for SHOT depending on radius size (dataset 1).

Table 3.4: Retrieval rate for SHOT depending on the radius size (dataset 1). The retrieval rates follow the same pattern than SPC algorithm: increases with the radius size for downsampling and noise and decrease after a specific value for occlusion.

	total	*raw*	*translation*	*rotation*	*downsampling*	*occlusion*	*noise*
λ = 1%	55.4	100	95.0	87.5	40.9	76.3	9.30
λ = 2%	62.9	100	91.2	100	57.6	81.9	14.3
λ = 5%	73.7	100	95.0	100	81.2	**87.5**	28.1
λ = 10%	81.4	100	100	100	95.1	85.4	45.0
λ = 50%	**81.7**	100	85.0	100	**100**	54.1	**77.5**



model name — number of points			
(a) camel	21889	(e) gorilla	41397
(b) cat	27896	(f) horse	19250
(c) david	52567	(g) michael	52567
(d) elephant	42323		

Figure 3.9: Dataset 2, reference objects (`training dataset`).

of: *R. Sumner dataset* [122] and *TOSCA high resolution dataset* [14]. We get 7 references and 169 objects holding between 20000 and 52000 vertices (Figure 3.9). Each of the 169 elements of the dataset is the result of a deformation of one of the reference object. One more time, they are isolated objects in high resolution, without significant noise corruption and no meshes, only vertices.

The 7 classes in this dataset are a `camel`, a `cat`, a human model named `david`, an `elephant`, a `gorilla`, a `horse` and another human model named `michael`. The deformed object from classes `camel`, `elephant` and `horse` are scanned of their reference during different moment of a simple motion (Figure 3.10). The rest — `cat`, `david`, `gorilla` and `michael` — are the references objects in different position: sit or stand up for instance. One last point to raise is the similarities between the two classes `david` and `michael`. As we can see from Figure 3.11 it is impossible even for the reader to distinguish between the classes `david` and `michael`.

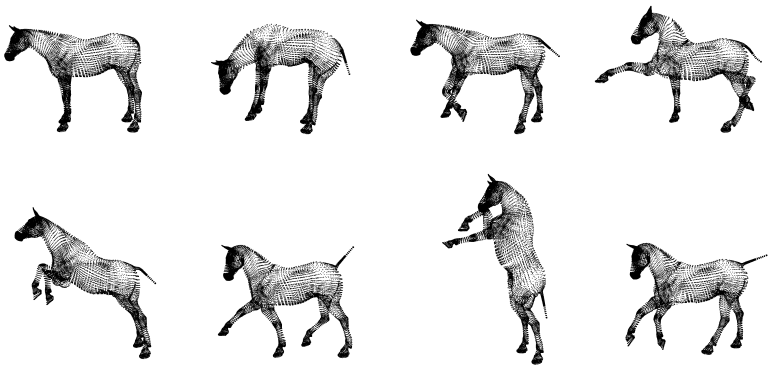


Figure 3.10: Example of object deformation with the class `horse` of the dataset.

(a) 4 objects of class `david`.

(b) 4 objects of class `michael`.

Figure 3.11: Illustration of the similarities between the two classes `david` and `michael`.

Table 3.5: Recognition rate (in %) for the dataset 2 with multi-scales methods (dataset 2). In contrast to dataset 1, GLPC (6%) is slightly more accurate than GLPC (8%), cf. Table 3.2.

	total	camel	cat	david	elephant	gorilla	horse	michael
# obj.	169	59	11	7	60	4	8	20
MPC2	92.8	100	100	100	100	50.0	100	50.0
GLPC (6%)	**98.2**	100	100	100	100	100	100	85
GLPC (8%)	97.6	100	100	100	100	100	100	80
global (6%)	95.2	100	100	85.7	100	100	100	65.0
global (8%)	91.1	100	100	71.4	100	50.0	100	45.0
local	96.4	96.6	63.6	100	100	100	100	100

3.3.2 Results

Table 3.5 and 3.6 display the recognition rate for the different techniques. GLPC still offers the best results with 98.2% while the other algorithms return miscellaneous but still good performances ratio. Indeed, with $\lambda = 6\%$ and $\lambda = 8\%$, GLPC recognition rate hit 98.2% and 97.6% respectively. With $\lambda = 6\%$, only a few instances of `michael` are not recognized: one more time, the global/local approach get the best of the two features. Unlike to dataset 1, with $\lambda = 6\%$, the results are slightly more accurate than with $\lambda = 8\%$. On the other hand, MPC2 is less effective, the recognition rate is only 92.8% with errors for classes `gorilla` and `michael`.

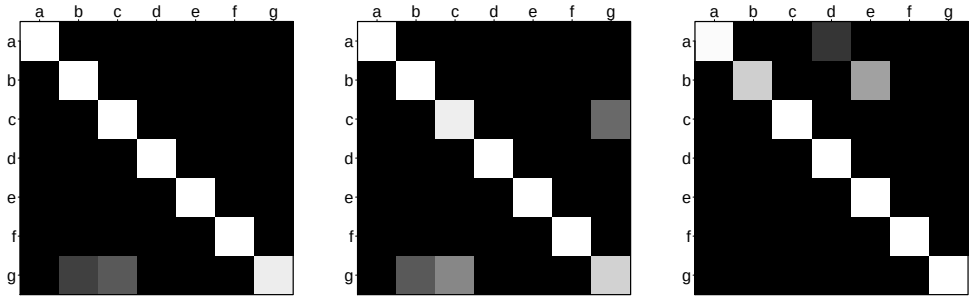
For the analysis of SPC results, Table 3.6 shows that a smaller studied radius gives better results than a big one. For instance with $\lambda = 2\%$ the overall rate is 94.0% vs. 86.9% for $\lambda = 10\%$. Results with SHOT and FPFH are comparable to SPC ($\lambda = 6\%$) with a recognition rate approximately equal to 95%. Such results are not surprising because dataset 2 focus on deformable objects and in such case, only a small part of the object is modified. Consequently, studying the object on a small scale is more ingenious, and it

Table 3.6: Recognition rate (in %) for the dataset 2 with monoscales methods (dataset 2).

	scale (in %)	total	camel	cat	david	elephant	gorilla	horse	michael
SPC	$\lambda = 2$	94.0	100	100	100	100	50.0	100	60.0
	$\lambda = 4$	86.9	100	36.3	85.7	95.0	50.0	100	55.0
	$\lambda = 6$	95.2	100	100	85.7	100	100	100	65.0
	$\lambda = 8$	91.1	100	100	71.4	100	50.0	100	45.0
	$\lambda = 10$	86.9	100	100	71.4	100	25.0	100	45.0
SHOT	$\lambda = 5$	95.2	96.6	63.6	100	100	100	100	90.0
FPFH	$\lambda = 5$	**97.6**	98.3	100	100	100	100	100	85.0

reflects the deformation better.

The confusion matrix of GLPC with $\lambda = 6\%$ is plotted in Figure 3.12. Objects are renamed according to the alphabetical order and the brighter the colour, the more object had been pooled into the class. At first glance at Figure 3.12, we notice that the diagonal is bright, underlying the high matching rate. GLPC only mismatches objects of class `michael` –85% of accuracy rate– into the classes `cat` or `david` and sort the objects of the others classes perfectly. We notice it can counterbalance the weakness of the global or the local feature with the other.



(a) GLPC (b) global feature (c) local feature

a: `camel` **b**: `cat` **c**: `david` **d**: `elephant` **e**: `gorilla` **f**: `horse` **g**: `michael`

Figure 3.12: Confusion matrix for algorithm GLPC with $\lambda = 6\%$.

Summary of Part I

This part focused on the instance retrieval of 3D point clouds object. Here the algorithms are tested against common errors encountered during the recording like noise, occlusion or subsampling. The tests also check their capability to retrieve deformable objects. The test dataset is composed of real scans of 3D objects in good quality and high resolution.

New descriptors are presented, relying on the curvature of the 3D objects; they offer the best performances for instance retrieval.

SPC descriptor describes the distribution of the curvatures in the object. The descriptor reflects the global shape of the object. The curvature being computed on a local neighbourhood of each point, a scale parameter has to be chosen. Compared to the state-of-the-art, SPC is more robust to noise and subsampling. However, as a global descriptor, the performances are slightly lower for occlusion. For deformable object, SPC offers similar results to the state-of-the-art.

MPC2 is a multiscale version of SPC, using different scales to compute the curvatures bypassing the choice of the study scale with SPC. With the usage of multiple scales, the global accuracy of MPC2 is slightly lower than SPC. The algorithm is less robust to noise and subsampling than SPC but better than the state-of-the-art and offers better accuracy than SPC in the case of occlusion.

Ultimately GLPC is another multiscale descriptor which uses two different feature descriptor for the global or local features of the objects. SPC is selected as the global descriptor and SHOT as the local. The two algorithms are complementary: SPC performs well against noise and subsampling and

SHOT against occlusion. The performances of GLPC outperforms the other algorithms, outputting the best of the local and the global for each recording error. For deformable object, GLPC also outperforms the state-of-the-art.

These descriptors are efficient for the instance retrieval problem, in the next part, they will be tested for a more complex problem: the classification of data. An attempt to reuse these computed features in more recent and powerful machine learning algorithms will be made to classify 3D objects.

Part II

Classification of 3D Point Cloud

Introduction

This part focuses on the problem of classification of 3D point clouds models. Given a known data set which holds different instances of objects in some categories and an unknown object, the classification problem involves retrieving the category of the latter. As presented in the previous sections, this problem is extremely popular with two-dimensional images as can be seen with the ILSVRC contest (Figure 1.7). The found solutions imply heavy use of deep learning algorithm.

However, for three-dimensional data and more particular 3D point clouds, the classification problem is relatively new and quite challenging. One more time, just like image processing and analysis, enhancement of computer capabilities and development of large dataset redefine the approach to 3D data analysis. Moreover, the emerging of the new paradigm of deep learning on 3D data is another reason for such growth, making it possible to solve old unresolved problems.

To start with, Chapiter 4, gives the principle of supervised machine learning and presents the main algorithms. Then, the dataset of 3D objects are described, and a study of the existing algorithms for classification is done.

A first step toward classification is tested in the Chapter 5 by using 3D descriptor from the previous Part I on the classification dataset. Then, the results lead us to study the deep learning algorithms performance with more realistic 3D object models and common 3D data recording errors. A comparison with curvatures-based algorithms is also performed. Ultimately, a new architecture based on 3D point cloud to do classification is proposed.

Chapter 4

Deep Learning with 3D Point Cloud

4.1 Supervised Machine Learning

4.1.1 Definition

Supervised machine learning algorithms are the collections of methods to build a prediction model from a labelled dataset. The training step is required to optimize the model. Formally, the set of data used for the training, named *training* or *learning* dataset is noted X. With $\mathbf{X} = \{\mathbf{X}_i\}_{i\in[\![0,M-1]\!]}$, M the number of element in the dataset. Each $\mathbf{X}_i = \left\{x_{i,j}\right\}_{j\in[\![0,d-1]\!]}$ is a feature vector of dimension d. Hence, X can be seen as a matrix of size $M \times d$. The labels of the data are noted $\mathbf{Y}$, $\mathbf{Y} = \{y_i\}_{i\in[\![0,M-1]\!]}$. Typically, $y_i \in [\![0, C-1]\!]$ with C the number of categories.

Finally, the model to approximate is the function f. After the training steps, f can predict an output y given an input vector $\mathbf{Z}$.

$$f : \begin{cases} \mathbb{R}^d \to [\![0, C-1]\!] \\ \mathbf{Z} \mapsto y \end{cases}$$

The objective of the supervised learning training is to find the *best* model,

i.e. function f. *Best*, means the model can predict the proper label for the data $\mathbf{X_i}$ of the training set but also for unknown $\mathbf{Z}$ data.

4.1.2 Loss Function

The algorithm uses the labelled dataset $\{\mathbf{X}, \mathbf{Y}\}$ and functions f_θ to estimate f. f_θ is a function depending on the hyper-parameters θ, hyper-parameters to optimize.

The loss function $L_M(\theta)$, also called cost function measures the prediction error made by the model f_θ on the training set $\{\mathbf{X}, \mathbf{Y}\}$. This loss function is in the simplest case the $0-1$ function (add 1 for each misclassified data), the mean square error (Equation 4.1) or the cross entropy loss (Equation 4.2), other functions can also be considered. The selection of the loss function depends on the nature of the wanted prediction.

$$L_M(\theta) = \frac{1}{M}\sum_{i=1}^{M}(y_i - f_\theta(\mathbf{X_i}))^2 \tag{4.1}$$

$$L_{M,i}(\theta) = -\log\left(\frac{e^{f_\theta(\mathbf{X_i})}}{\sum_{k=1}^{M} e^{f_\theta(\mathbf{X_k})}}\right) \tag{4.2a}$$

$$L_M(\theta) = \sum_{k=1}^{M} L_{M,i}(\theta) \tag{4.2b}$$

During the training stage, different parameters θ are tested, and the function f_θ with the lowest loss is selected.

$$f = f_{\theta_{\min}}, \quad \text{with } \theta_{\min} = \arg\min_{\theta} L_M(\theta) \tag{4.3}$$

The training step can be seen as an optimization problem. However, it is unsolvable with an analytical solution. This is why iterative algorithms are used to approximate the best parameters for the models.

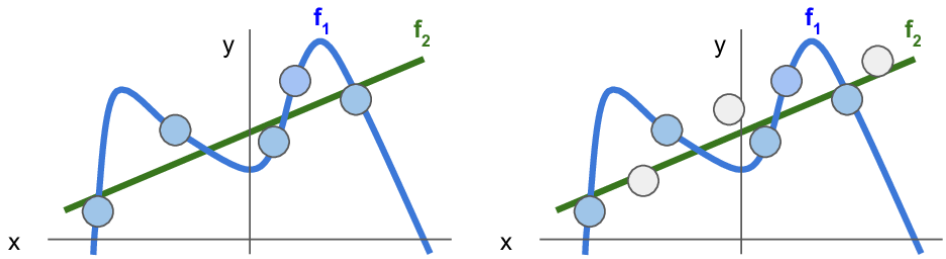


Figure 4.1: Illustration of overfitting. (a): example of two separation boundary. In blue, a line that entirely separates the data, in green a line that accepts error but with a simpler mathematical expression. (b): when new sample data are added, the blue line is no more a correct boundary contrary to the green line, which keeps the same behaviour than previously. The regularization pushes against fitting the data too well, keeping the boundary simpler, giving better generalizations performances.

4.1.3 Overfitting

The *overfitting* occurs when a model prediction is too close to a particular set of data. Basically, the overfitted model extracted some of the residual variation, the noise for instance, as a piece of representative information. When the model overfits, the performances for generalization fall.

Usually, the overfitted model has too many parameters to optimize. The model is uselessly complex when a simpler one, with fewer parameters, would be enough.

Cross-validation is usually used to avoid overfitting. The training data set is divided into two: a train set and a validation set. The train set is used to train the model and then the model is used to predict the output of the validation set. A comparison (Figure 4.2) of the predicted output and the proper value can be done and give an idea of the generalization capabilities of the trained model.

Another solution is to use regularization, i.e. add parameters in the

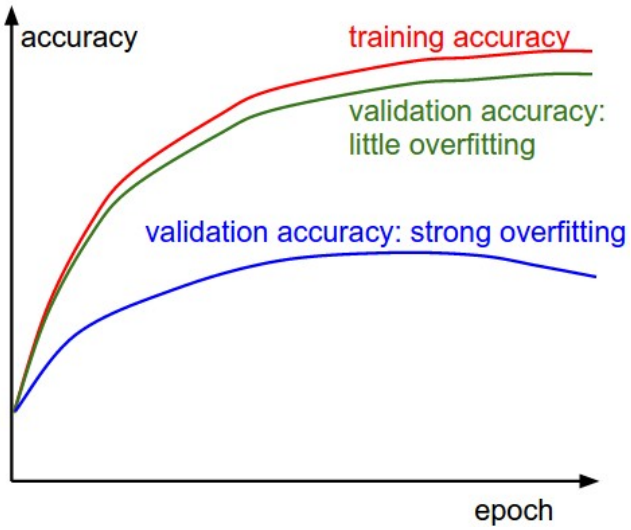


Figure 4.2: Train/Validation accuracy graph. The gap between the training and the validation accuracy indicates the overfitting tendency of the model. If the two lines are close (green line), the model can generalize the learnt features to the validation dataset: the model is "good". On the other hand, if the lines move apart (blue line), the model has strong overfitting. The features learnt rely too much on the training set.

training stage, a penalty factor to limit the complexity of the model. Penalize the extreme values, which often leads to overfitting, is a usual regularization technique. For this, the norm of the data is computed and added to the function to minimize. A condition on the parameters θ can also be set to keep it "small".

$$L_M(\theta) = \sum_{k=1}^{M} L_{M,i}(\theta) + R(\theta) \tag{4.4}$$

For instance, with $L2$ regularisation, $R(\theta) = \frac{1}{2}\lambda\|\theta\|_2$, $L1$ regularization is simply with $R(\theta) = \lambda\|\theta\|_1$. A combination of the two can also be used.

4.1.4 Classification Algorithms

Lots of supervised learning algorithms with different approach has been studied. Here, a few of them are presented.

Naive Bayes classifier [82] is maybe one of the oldest classifier. It relies on Baye's theorem with the hypothesis of a strong correlation between the feature. In this model, one feature of a class is considered independent of its other features. For instance: a horse is an animal with 4 legs and a tail, if another animal presents the same feature, it will be considered as a horse. A dog for instance, would be considered as a horse if the model does not include more features in the analysis. This simple example does not reflect the efficiency of Bayes classifiers. Indeed, this algorithm is efficient in several applications like medicals diagnosis or text categorization (categorize a text as a sport, science, politics or spam, etc.). For text categorization, the frequencies of the words are used as a feature, and in medical diagnosis, it is the symptoms. One advantage of Bayes classifiers is that the amount of data required for the training can be small.

k-Nearest-Neighbours (k-NN) [27] is another commonly used algorithm. To ascertain the prediction of an unknown data with this classifier, a search of the k-th nearest neighbours is done. k is a hyper-parameters set by the user. The predicted class is the most common class in this neighbourhood. When $k = 1$, the predicted class is the one of the closest neighbours. The metric distance is a crucial choice. Depending on the type of data: Manhattan or Euclidean distance can be considerate, but Hamming distance is more adapted to text classification, χ_2 distance for histograms, etc. Sometimes weights are applied to give more importance to the closest neighbour and less to the ones which are further. This method is then straightforward and does not require a training step. However, for every prediction, the k closest neighbours have to be found which can be costly in time, even more, if the data set is large.

Decision tree learning [98] uses decision tree as a predictive model. A decision tree is built during the training step by using the training data set. In a decision tree, each node performs a test on the incoming data, depending on the result, one of the branches is selected. In the end, the reached leaf node (the last level of nodes) represents the decision taken. The training is required to find the best test for each node: a good test is a test that splits

the data into equivalent subsets of data. The training should build the most unadorned tree: a tree with a large number of nodes and leaves means that many tests are performed on the training set, which can lead to an overfitted model. For the same performance on the learning set, the smallest the tree, the more generalizable it is. Decision trees are easy to understand and implement, moreover, no processing is required on the data. On other hands the training is a complex, the problem is NP-complete [71], and the algorithm is also prone to overfitting. Moreover, the decision trees are not very robust: a small noise or change on the data can considerably modify the tree, thus the prediction.

Random forest classifier [12] is an enhancement of the decision tree learning. The main idea is to use train multiple decision trees and different subsets of data. The main contribution of Breiman [12] is the "bagging" which catch up decision tree tendency to overfit. The bagging, from *b*ootstrap *agg*regat*ing* consists in choosing B subsets of training data, each one holding a certain number n of sample. Then for each of these samples, a classification tree is trained. In the end, an averaging of the prediction of all the classification tree is done to get the final prediction. Because the decision trees are usually sensitive to small variations of the learning dataset, the different decision trees are not correlated. Hence the better prediction when using the prediction of all the forest. In fact, in a random forest training, the training of the decision trees is modified. Instead of using all input features, only a random subset is used. In this way, the correlation between the trees is decreased, reinforcing the accuracy of the final prediction.

Genetic programming [66] is a technique inspired by the Natural selection mechanism described by Darwin in 1859. The idea is to select the most important features, and a random modification is done: the mutation. A study of the model is done, and the mutations are validated or not. This step is repeated a given number of time. This algorithm is costly in term of computation as a significant number of mutation can be tested. However, the enhancement of CPU alleviates this problem, leading to more intensive use. Genetic programming is used when the form of the solution is barely known

in advance. For instance, to fit an unknown curve with a function, the use of a genetic programming algorithm should definitely be considered.

Two more majors algorithm for classification are proposed in the literature: the Support Vector Machine by Vapnik [126] and the Multi-Layer Perceptron by Rosenblatt [102]. Both of them are detailed in the following sections.

4.2 Support Vector Machine Algorithm

Support Vector Machine (SVM) are supervised learning algorithms which perform classification and regression analysis. The first version proposed by Vapnik [126] was published in the 1963s. In the version, the SVM was designed for binary classification of linearly separable data. It is only in 1992 with [10], the *kernel trick* (cf. below Section 4.2.2) were introduced, allowing a non linear classifiers. In 1995 [25], the introduction of *soft margin* allowed another improvement. This version, [25], set up the SVM as we know and trigger the use of the SVM.

Initially, the SVM was designed for binary classification, i.e. problems with two classes only. The main idea is to find a hyperplane in the space of the data which separate them into their classes. For multi-class classification, different strategies were developed, cf. Section 4.2.4.

4.2.1 Linear SVM

In the first version proposed by Vapnik [126], the SVM was limited to binary and linearly separable data. With the previously introduce notation (cf. Section 4.1), the separator hyperplan h can be written:

$$\{\mathbf{x} / \quad h(x) = \mathbf{w} \cdot \mathbf{x} - w_0 = 0\} . \tag{4.5}$$

With $\mathbf{w}$ a normal vector to the hyperplan and $\frac{b}{||\mathbf{w}||}$ the offset of $\mathbf{w}$ at the origin.

In the case of linearly separable data, the choice of the separator hyperplane is not trivial. Indeed, an infinite number of hyperplane would properly separate the training data but with different performances for generalization. In fact, two parallel separator hyperplanes can be found with the distance between them being as large as possible. This distance is called the *margin* and the separator hyperplane is the one that lies halfway between them. *Support vectors* are then vectors which are the closest to the separator hyperplane, they belong to one of the previously parallel hyperplanes. Because it is a linear classification, any data "above" the hyperplan, i.e. $h(\mathbf{x}) \geqslant 0$ belong the class labelled 1, any "below" ($h(\mathbf{x}) \leqslant 0$) belong to the other class, labelled as -1. Hence,

$$\forall k \in [\![0; M]\!], \quad y_i \left(\mathbf{w} \cdot \mathbf{x}_k + w_0\right) \geqslant 0.$$

The SVM algorithm is then equivalent to find $\mathbf{w}$ and $\mathbf{w_0}$ such that:

$$\arg \max_{\mathbf{w},\mathbf{w_0}} \min_k \{\|\mathbf{x} - \mathbf{x_k}\| \ / \ \mathbf{x} \in \mathbb{R}^d, \mathbf{w} \cdot \mathbf{x} + w_0 = 0\}$$

The distance between the hyperplan and a vector $\mathbf{x}$ is equal to:

$$\frac{|\mathbf{w} \cdot \mathbf{x} + w_0|}{\|\mathbf{w}\|}$$

For a support vector $\mathbf{x}_{\text{support vector}}$, the distance is $\frac{1}{\|\mathbf{w}\|}$. The separator hyperplan witht the maximum margin is then given by:

$$\arg \max_{\mathbf{w},\mathbf{w_0}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_k \left\{ y_k \left(\mathbf{w} \cdot \mathbf{x_k} + w_0\right) \right\} \right\}$$

To simplify the problem, $\mathbf{w}$ and w_0 can be choosen such that for the support vectors $\mathbf{x_k}$:

$$\mathbf{w} \cdot \mathbf{x_k} + w_0 = \pm 1.$$

Then

$$\forall k \in [\![0; M-1]\!], \ y_i \left(\mathbf{w} \cdot \mathbf{x}_k + w_0\right) \geqslant 1, \tag{4.6}$$

with equality if $\mathbf{x}_k$ is a support vector. The optimization problem becomes:

$$\boxed{\begin{aligned}&\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2\\&\text{subject to } y_k\left(\mathbf{w}\cdot\mathbf{x}_k + w_0\right) \geqslant 1,\ k \in [\![0; M-1]\!].\end{aligned}} \tag{4.7}$$

Resolution of the optimization problem

The resolution of this problem can be made with Lagrange multipliers with:

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{k=1}^{M} \alpha_k \left(y_k\left(\mathbf{w}\cdot\mathbf{x_k} + w_0\right) - 1\right). \tag{4.8}$$

L needs to be minimize over $\mathbf{w}$ and w_0, and maximaze over α.

By setting the partial derivative of L to zero:

$$\mathbf{w} = \sum_{k=1}^{M} \alpha_k y_k \mathbf{x}_k \tag{4.9a}$$

$$0 = \sum_{k=1}^{M} \alpha_k y_k \tag{4.9b}$$

By combining Equations 4.8 and 4.9 the problem because:

$$\boxed{\begin{aligned}&\text{Maximize } L(\alpha) = \sum_{k=1}^{M} \alpha_k - \frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{M} \alpha_j \alpha_k y_j y_k \mathbf{x}_j \cdot \mathbf{x}_k\\&\text{subject to } \begin{cases}\alpha_k \geqslant 0\\ \displaystyle\sum_{k=1}^{M} \alpha_k y_k = 0.\end{cases}\end{aligned}} \tag{4.10}$$

This is the dual form of the problem. Since L is a quadratic function of $\{\alpha_k\}_k$, the problem can be solved with commons method like active set [91] or augmented Lagrangian [31]. Once the optimal $\{\alpha_k^*\}_k$ found, they are

reintroduced into the Equation 4.9 and in the hyperplan Equation 4.5.

$$\mathbf{w} = \sum_{k=1}^{M} \alpha_k^* y_k \mathbf{x}_k, \tag{4.11a}$$

$$h(\mathbf{x}) = \sum_{k=1}^{M} \alpha_k^* y_k \mathbf{x} \cdot \mathbf{x}_k + w_0. \tag{4.11b}$$

One may point out some consequences of this optimal solution. Under Karush-Kuhn-Tucker condition [69]:

$$\forall k \in [\![0; M-1]\!],\ \alpha_k \left(y_k h(\mathbf{x}_k) - 1\right) = 0$$

Yet Equation 4.6 shows $y_k h(\mathbf{x}_k) = 1$ if $\mathbf{x}_k$ is a support vector.

Hence,

$$\alpha_k = 0 \text{ if } \mathbf{x}_k \text{ is not a support vector.}$$

With Equation 4.11a, this shows that $\mathbf{w}$, the normal vector to the separator hyperplan a linear combination of the support vectors only. The rest of the vectors do not impact the separator plan: they become "useless" for the model.

4.2.2 Non Linear SVM

The problem of non-linear SVM is studied by Boser et al. [10]. Because no separator hyperplane can be found in the space of the data, the idea is to use a transformation to increase the dimensionality. In this higher dimension space, such hyperplane can be found. This operation is called the *kernel trick*. The transformation function is φ, the input is the data, and the output is vectors in a higher dimension and possibly with infinite dimension. This latter space is called the feature or kernel space.

In the feature space, the separator hyper plan is given by:

$$\{\mathbf{x} \,/\, h(x) = \mathbf{w} \cdot \varphi(\mathbf{x}) - w_0 = 0\}\,.$$

The only difference with Equation 4.5 is the use of the function φ. The same

approach than linear SVM outputs the following problem.

$$\begin{aligned} &\text{Maximize } L(\alpha) = \sum_{k=1}^{M} \alpha_k - \frac{1}{2} \sum_{j=1}^{M} \sum_{k=1}^{M} \alpha_j \alpha_k y_j y_k \varphi(\mathbf{x}_j) \cdot \varphi(\mathbf{x}_k) \\ &\text{subject to } \begin{cases} \alpha_k \geqslant 0 \\ \sum_{k=1}^{M} \alpha_k y_k = 0. \end{cases} \end{aligned} \tag{4.12}$$

With this formulation, the dot product of $\varphi(\mathbf{x_j})$ and $\varphi(\mathbf{x_k})$ need to be computed. However, in high dimensional space, this operation can be computationally expensive. To bypass this problem, a kernel function K is used, such as:

$$K\left(\mathbf{x}_j, \mathbf{x}_k\right) = \varphi(\mathbf{x}_j) \cdot \varphi(\mathbf{x}_j).$$

And finally, the separator hyperplan is given by:

$$h(\mathbf{x}) = \sum_{k=1}^{M} \alpha_k^* y_k K(\mathbf{x}, \mathbf{x}_k) + w_0. \tag{4.13}$$

So, the computation is done in the data space, and the function φ does not need to be known. The most commons kernel functions are the polynomial and the Gaussian kernel.

$$K(\mathbf{x}_j, \mathbf{x}_k) = \left(\mathbf{x}_j \cdot \mathbf{x}_k\right)^q \tag{4.14a}$$

$$K(\mathbf{x}_j, \mathbf{x}_k) = \exp\left(-\gamma \cdot \|\mathbf{x}_j - \mathbf{x}_k\|^2\right), \quad \gamma \geqslant 0 \tag{4.14b}$$

4.2.3 Soft Margin

Until now, the data were implicitly supposed linearly separable (in the feature space). However, in most case, some data do not fit for a perfect separation (Figure 4.3), hence the impossibility to find a proper separator hyperplane.

Cortes and Vapnik [25] solve this problem by using a *soft margin*. Cost parameters ξ_k, $\xi_k \geqslant 0$ are introduced in order to tolerate some training data

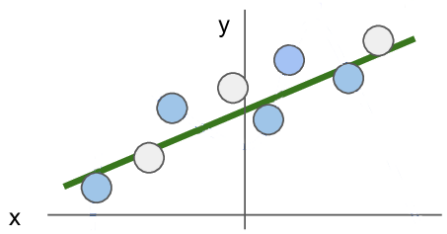


Figure 4.3: Illustration of soft a margin. Without a soft margin, i.e. the acceptance of error in the classification by the hyperplane, there is no way to find a proper and simple separator hyperplane.

to be misclassified, Equation 4.6 becomes:

$$\forall k \in [\![0; M-1]\!],\, y_i\,(\mathbf{w}\cdot\mathbf{x}_k + w_0) \geqslant 1-\xi_k \quad \xi_k \geqslant 0.$$

The optimization problems (Equation 4.7) becomes:

$$\boxed{\begin{array}{l}\text{Minimize } \dfrac{1}{2}\|\mathbf{w}\|^2 + C\displaystyle\sum_{k=1}^{M}\xi_k \\ \text{subject to } \begin{cases} y_k\,(\mathbf{w}\cdot\mathbf{x}_k + w_0) \geqslant 1-\xi_k \\ \xi_k \geqslant 0,\ k \in [\![0; M-1]\!]. \end{cases}\end{array}} \tag{4.15}$$

The parameters C, set by the user, control the amount by which the training data can be on the wrong side of the separator hyperplane. This optimization problem can then be solved with the same method than previously.

Boser et al. [10] and Cortes and Vapnik [25] successively improved the first version of the SVM proposed by Vapnik [126]. Getting better generalization capacities. However, the hyper-parameters can notably influence the results, and this choice is crucial for the performances as wrong values can considerably decrease the efficiency.

4.2.4 Multiclass SVM

For non-binary classification problems, two main solutions exist. With P is the number of categories.

One versus all: P binary classifiers are built, for each of them the labelled class is associated to 1 and the rest to -1. The predicted class is the one with the highest margin.

One versus one: $P(P-1)/2$ binary classifiers are built, each class is confronted with each other. During the test, the data is tested with each classifier: the winning classes are reported for all of them. In the end, the class with the most win is the predicted class.

These 2 methods have some defaults. Firstly, because the classifiers are independents between them, there is no reason the margins could be compared (no normalization). The other drawback is the unbalanced distribution: only $1/P$ ratio of data represent the proper class and $(P-1)/P$ the other one.

4.3 Multi-Layer Perceptron

A *multi-layer perceptron* or MLP is a class of artificial neural network. At least three layers form an MLP, the information going only in one direction through the layers, hence the use of the term "feed-forward" to qualify them. The structure of an MLP is made of layers. Each one holding a variable number of neurons and each of them uses a nonlinear activation function.

MLP can classify well non linearly separable data and generalize to unknown data. However, the training process is complex and costly in time and resources. Moreover, with neural networks, it is difficult to determine the architecture suitable for the problem. Several trials could be required before getting positive progress. Finally, MLP is often be seen as a black box: the mechanism behind the layers and the nodes is difficult to understand and interpret.

4.3.1 The Perceptron

A neuron or perceptron is a mathematical function that mimics the behaviour of a biological neuron and being able to do a binary classification. The perceptron was invented in 1957 by Rosenblatt [102], but although promising, it was not an active research topic for a while.

If f is the function associated to the perceptron, $\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^m$ the input data and a the activation function, then:

$$f(\mathbf{x}) = a(\mathbf{w} \cdot \mathbf{x} + b)$$

$\mathbf{w}$, $\mathbf{w} \in \mathbb{R}^m$ is the weight vector and b, $b \in \mathbb{R}$ is the bias. The bias shifts the boundary decision by a constant factor which is independent of the data. The activation functions a maps the weighted input sum to the output and makes sure that the data remains into a desirable range.

Historically, the activation function used is either the Heaviside step function (Eq. 4.16a), the hyperbolic tangent (Eq. 4.16b) or the sigmoids (Eq. 4.16c).

The simplicity of the perceptron limits the capabilities of the model hence the little use.

$$a_1(x) = \begin{cases} 1, & \text{if } x \geqslant 0 \\ 0, & \text{else} \end{cases} \tag{4.16a}$$

$$a_2(x) = \tanh(x) \tag{4.16b}$$

$$a_3(x) = \frac{1}{1 + e^{-x}} \tag{4.16c}$$

4.3.2 Multi-Layer Perceptron

In opposition to the perceptron or single-layer perceptron, *multi-layer perceptron* was developed. The term "multi-layer" is misleading. Indeed, an

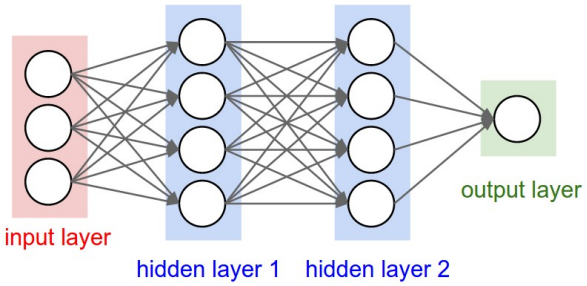


Figure 4.4: Schematic view of a multi-layer perceptron. MLP with two hidden layers (blue), the input layer (red) and the output layer (green). The layers have respectively 3, 4 and 1 nodes, and the nodes of a layer are connected to all the node of the previous layer.

MLP is not a perceptron with multiple layers but multiple layers with multiple perceptron unit in each. An MLP structure is the following: an input layer, a certain number of hidden layer and finally output layer. The input layer receives the input features from the dataset, and the output layer is the last one, the layer that predicts the outcome of the model. Hidden layers are "inside" the model: the user does not see them, hence the name. The number of these hidden layers depends on the architecture wanted.

The layers are connected in sequential order, and the information is following only one way. They are numbered from 0 for the input to $M-1$ for the output layer, M being the number of layers, L_i being then the i-th layer.

Then, the layer i of the MLP is a function $f_i \colon \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$, with d_i and d_{i+1} the number of perceptrons in L_i and L_{i+1}. d_0 is the size of the data entry vector and d_M is the size of the ouput. A layer L_i has a similar functioning that the Perceptron presented in 4.3.1 except that the weight is now a matrix instead of a vector.

$$\begin{aligned} f_i\left(\mathbf{x_i}\right) &= a\left(\mathbf{W}_i \cdot \mathbf{x}_i + \mathbf{b}_i\right) \\ f_i\left(\mathbf{x_i}\right) &= \mathbf{x}_{i+1} \end{aligned} \tag{4.17}$$

$\mathbf{W}_i$ is the weight matrix of size $d_{i+1} \times d_i$, $\mathbf{b}_i$ is the bias vector of size d_{i+1} and a is the activation function. The previously presented activation functions

(Eq. 4.16a and Eq. 4.16b) can be used in their vectorized version. However, an alternative function: the rectifier linear unit or ReLU [48, 47] (Eq. 4.18) has been proposed to face the *vanishing gradient problem* during the training. The training step is also speeded up by using ReLU [73].

$$a_4(x) = \max(0, x) \tag{4.18}$$

MLP Training

The weights are the crucial element behind the training of an MLP. During the training step, the weights are adjusted to decrease the value returned by the loss function and so improve the accuracy.

There are two crucial stages in the training, the feed-forward propagation and the back-propagation.

During the **feed-forward**, the data feed the network, each intermediate output $\mathbf{x}_{i+1}$ is computed thanks to f_i (eq. 4.17). For each layer, the output is multiplied by the weight matrix, bias is added, and nodes are activated or not. The resulting vector is then the input of the following layer (Eq. 4.17). The last layer gives the prediction vector. The proper output vector is also computed thanks to the known label. The two vectors are compared with the loss function L which is a function of the training set, $\mathbf{W_i}$ and $\mathbf{b_i}$ (cf. Section 4.1.2).

The **back propagation** step updates the weights and the bias to reduce the error, i.e. the value of the loss function. The gradient descent process [72] reduces the loss: the gradient of L is computed and the weight and bias are updated according to the following formula:

$$\mathbf{W}_i \leftarrow \mathbf{W}_i - \eta \cdot \nabla_{\mathbf{W}_i} L(\mathbf{W}_i) \tag{4.19a}$$

$$\mathbf{b}_i \leftarrow \mathbf{b}_i - \eta \cdot \nabla_{\mathbf{b}_i} L(\mathbf{b}_i) \tag{4.19b}$$

η is the *learning rate*, a positive hyperparameter, $\mathbf{W}_i$ and $\mathbf{b}_i$ the weights and bias of the i-th layer. The idea by using the gradient of the loss function is

to update the parameters toward values such that the loss function gets smaller. The learning rate controls the step of the overriding value. If η is too high, the step might be too big, and the algorithm might jump a minimum. On the contrary, if η is small, the algorithm is longer to converge, but an unwanted local minimum instead of the global on can be reached.

Once the back-propagation over, another forward pass is performed and so one. The number of forward-backward passes is the number of epochs. The training is over when a certain amount of epochs is reached or when the loss function does not decrease after a few updates.

Parameters Initialization and Regularization

Several strategies exist to initialize the weights and bias parameters. The best solution is to use random weights and limits its variance. If the i-th layer input is a vector of size d_i, then the weights of $\mathbf{W}_i$ follow a normal distribution of mean 0 and variance $\sqrt{\frac{1}{d_i}}$. The same is applied for the bias $\mathbf{b}_i$. Glorot and Bengio [45] advise a value of $\sqrt{\frac{2}{d_i+d_{i+1}}}$ while He et al. [51] suggests to a value of $\sqrt{\frac{2}{d_i}}$. Initialize the all parameters to zero or a constant often lead to poor results. Finally, the *batch normalization* imagined by Ioffe and Szegedy [57] is a technique which decreases the effect of the weights initializations on the training. The batch normalization transforms the output of a layer into a unit Gaussian distribution.

The overfitting with MLP networks can also be prevented using the methods presented in Section 4.1.3. However, for neural network, *dropout* [119] is an extremly effectif and simple technique to avoid overfitting. During the training, a percentage p of the neurons are "deactivated": their weights are set to zero. After each epoch, the deactivated neurons are randomly updated.

Implementation Notes

Here, a few notes about MLP implementation are given. As presented previously, the gradient descent process computes the gradient and updates the parameters. This step should be performed on the whole training data set. For each sample of the set, the MLP is applied and the lost function is computed, the global loss function is the sum of all loss. Only then, the gradient is computed and the parameters updated. The process then is repeated and so on. However, for large a large dataset, this algorithm can be costly since the parameters update is only done once the MLP feed-forward the whole set. This is the reason why *stochastic gradient descent* (SGD) and *mini-batch stochastic gradient descent* (MBSGD) are used instead [11]. The idea is to use less data before the update of the weights. In SGD, the weights are updated after each training example. In MBSGD, the update is done on a batch of training samples; usually, the batch size is between 16 and 128. The three algorithms offer similar performances, MBSGD converging a bit faster than the "vanilla" gradient descent.

Some extension and variant are also proposed: Momentum [103], AdaGrad [34], RMSProp [124] or Adam [64]. All of them try to bypass the need to set a learning rate.

4.4 3D Models Datasets

4.4.1 KITTI Dataset

KITTI dataset [42] is a dataset combining all the data captured by a car during a 6 hours experiment. A car with multiple sensors recorded the data while driving in Karlsruhe in Germany. The car includes 2 grayscale cameras, 2 colour cameras, 1 Velodyne 3D laser scanner and some inertial and GPS sensors. All these sensors generate several datasets: a stereo/flow benchmark, an odometry benchmark and objects detection. The object detection offers 2D and 3D representation. The 3D object dataset is constituted of objects that are among the following categories: cars, vans, trucks, pedestrians, cyclists

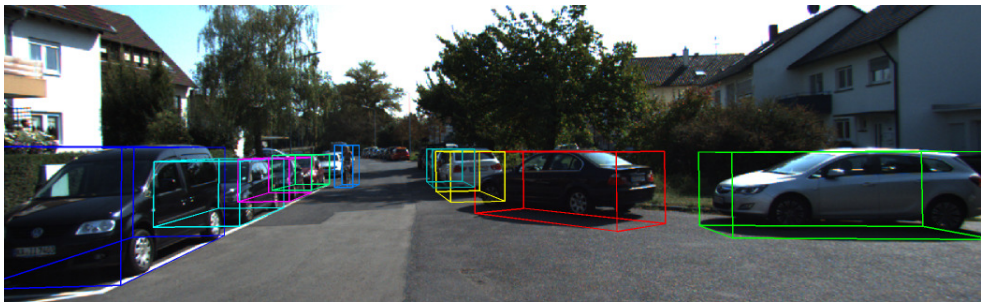


Figure 4.5: Illustration of KITTY dataset. Visualisation of a 2D image with 3D bounding boxes. 9 bounding boxes are visible: 8 cars and 1 cyclist. Illustration from [41].

and trams. The dataset is not a dataset of isolated objects: the whole scene is given, and an accurate bounding box to interesting objects is added to the data. The label was done manually from the raw data of the Velodyne sensor.

The 2D and 3D object detection contest evaluate the accuracy of the detection thanks to the computed bounding boxes. For the last contest, in 2017, the dataset consisted of 7481 training and 7518 test samples. As a sample is not limited by holding only one object, there is a total of 80256 labelled object. The contest is limited to three categories object detection: car, pedestrian and cyclist. The best algorithm so far is Wang and Jia [130] with 76.5%, 45.6% and 64.7% for respectively car, pedestrian and cyclist. Liang et al. [78], Shi et al. [113], Lang et al. [70] and many more also use KITTI 3D objects dataset to evaluate their performances for detection.

4.4.2 ShapeNet Dataset

ShapeNet [20] is an extensive annotated repository of 3D CAD models of objects. The annotations use WordNet taxonomy [87], they described the size, the alignment, the parts, the symmetries plan, etc. In total, ShapeNet has more than 3 millions of models with 220000 of them which are classified into more than 3000 categories. The objects are meshed models from popular public repositories: Trimble 3D Warehouse and Yobi3D. ShapeNet is constituted of

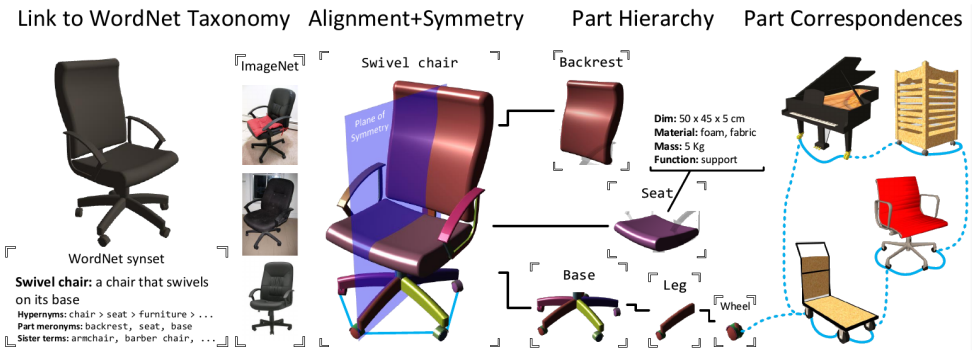


Figure 4.6: Illustration of ShapeNet annotation. Here a meshed model of a chair is used as an example. Left: Use of WordNet taxonomy to describe the object and link to the corresponding images in ImageNet. Middle-left: Alignment and symmetries. Middle-right: Decomposition of the object into parts. Each part is annotated with a name, the size, the materials, the symmetries, etc. Right: Creation of the correspondences network: all the parts of the objects are connected between them. Illustration from [20].

2 subsets:

- ShapeNetCore which is a subset of the full dataset. The data are clean and manually verified and aligned. This subset represents 55 categories of objects with 51300 3D objects in total.
- ShapeNetSem [107] is smaller with only 12000 3D objects shared out in 270 categories. The objects are also manually verified, aligned and finely annotated with information about their size, weight, volume, composition, etc. This dataset aims to provide richly-annotated 3D objects for research on 3D semantic and reasoning.

ShapeNet is widely used, for shape retrieval [120, 5] or as base for 3D dataset [134, 29, 89].

4.4.3 PartNet Dataset

Very recently, PartNet dataset was published by Mo et al. [89]. While existing datasets do not provide part annotation for a many objects instance [21] or only rough part annotations, PartNet provides fine-grained and hierarchical

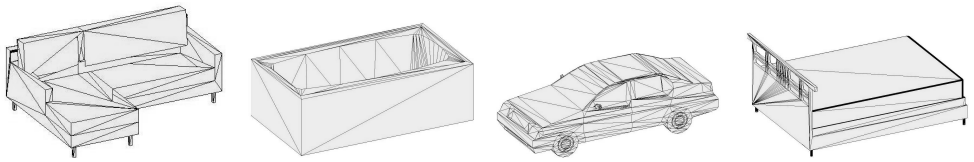


Figure 4.7: Examples of 3D model of ModelNet [132].

part information from ShapeNet dataset. PartNet is composed of 26671 3D objects in 24 categories: the total number of part is up to 573585. As more and more researches use point cloud representation, the author chooses this representation for the object of the dataset.

This dataset can be used for 3D segmentation, shape or semantic analysis. As a very new dataset, only a few studies can be found, for instance, Yi et al. [140] focus on instance segmentation in a point cloud.

4.4.4 ModelNet Dataset

The standard benchmarking dataset for 3D classification is ModelNet by Wu et al. [132]. ModelNet is a large object dataset of 3D objects in voxel grid representation. The grid size is $30 \times 30 \times 30$. Two versions are published:

- ModelNet40 with 12311 models in 40 categories and
- ModelNet10 which reduce the number of categories to 10 and therefore the number of models with 4899 of them.

Hand-made objects from different categories are selected using statistics computed from SUN database [135]. The categories are many and various like airplane, bathtub, car, desk, laptop, radio, sofa, table, wardrobe, etc. The dataset is then manually labelled, for ModelNet10, the object are also aligned. One last thing to note is that ModelNet is not composed of real scan objects, all objects in the dataset are CAD models. Most classification algorithms are tested with ModelNet, c.f. Table 4.1.

4.5 3D Classification Algorithms

4.5.1 Deep Learning on Descriptor

Deep learning on descriptor was explored when the deep learning emerged. As described in Part I, descriptors are handcrafted algorithms and hence suffer from significant limitation as they only compute one kind of feature. However, deep learning algorithms are very efficient to find similarities between the training data and hence generalize to unknown data. Usually, the descriptors are computed in a first step and feed a deep learning network.

Liu et al. [79] use a low-level 3D shape descriptor based on a visual bag-of-words. These low-level features feed a network: deep belief networks to extract high-level semantic features of the input. These high-level features are then used for shape classification and retrieval. The experiments show that the results with high-level features are better than the standard bag-of-words low-level feature.

Bu et al. [16] has a similar approach, except that there are three instead of two levels of features. The first level is the descriptor computed with the scale-invariant heat kernel signature [15]. The bag-of-feature, the middle-level features is composed by the bag-of-word and the previously computed descriptor. Finally, the deep belief networks are used to get the high-level feature.

Fang et al. [37] used the heat kernel signature of the object as a descriptor. By using the heat kernel, they simulate the heat diffusion of the 3D object, getting a signature of this one. The descriptor then feeds a deep learning model to learn the features for 3D retrieval.

These algorithms are outperformed by more recent approaches that do not rely on handcrafted shape descriptor. Because the use of descriptor limits the learning to one type of feature in the data, supervised machine learning methods cannot be used at full potential. Hence, deep learning algorithms on descriptor are better to classify the pattern from the descriptors which

express, for instance, geometric features of objects.

4.5.2 Deep Learning on 3D Data Projection

Here, the 3D data are projected on 2D planes, and this projection is then used by deep learning models. Shi et al. [112] use a cylindrical projection around the principal axis of the object. The projection then feeds a traditional 2D convolutional neural network. Sinha et al. [115] project the 3D objects on a 2D grid so that classical 2D convolutional neural networks can be used. Sfikas et al. [110] represent a 3D object as panoramic views extracted from a normalized posed of the object.

Projection of 3D data on a 2D space is a simple and effective way to use deep learning on 3D data. However, these techniques rely mostly on the 2D deep learning models that need to be fine-tuned to get accurate results.

4.5.3 Deep Learning on RGB-D Data

Deep learning on RGB-D is relatively popular since this type of dataset is more straightforward to create than other representation. Moreover, RGB-D data structure is close to the standard RGB 2D image. The idea is to use typical deep learning architecture for the RGB data and create another network to process the depth information. The two networks are then combined to get the final result.

Socher et al. [118] proposed the first approach: two convolutional neural networks are used, one for the colour image and the other one for the depth map then the output of each feed a recurrent neural network. Finally, the two output are combined and used as input of a softmax classification.

Couprie et al. [26] use a combination of multiple techniques. The RGB-D data, the depth being seen as the fourth colour channel, feed to multiple convolutional neural networks, each one processing a different scale and the output of all these networks is combined. A segmentation of the RGB image is used as superpixels along with the output of the convolutional neural networks

as the input of another network which label the data. The algorithm is 6% more accurate than previous models.

Eitel et al. [35] use two neural networks, one for the RGB image and the other one for the depth map image. There are five convolutional layers and two fully connected layers, the output of these two networks is concatenated and feeds a fully connected neural network. The method outperforms existing recognitions methods.

Alexandre [2] goes even further, four CNN's are used, one for each channel. The CNN's are processed independently before a transfer learning between them.

As seen, deep learning with RGB-D data is accessible and offers good performances. However, these methods can be seen more like an extension of 2D images rather than real 3D data.

4.5.4 Deep Learning on Volumetric Data

Just like 2D pixel image representation, a 3D Volumetric data representation is set according to a regular grid. Hence the close paradigm between the deep learning model on image and Volumetric data.

Wu et al. [132] are the first to propose a deep learning model relying on 3D voxels: ShapeNet. The input is a binary grid of dimension $30 \times 30 \times 30$, the value of the voxel indicate if the voxel belongs to the object or not. The network consisted of five layers: the input, three convolutional layers and the output layer. ShapeNet was tested for 3D objects classification, view-based recognition and next base-view recognition. ModelNet dataset was created by Wu et al. [132] to test ShapeNet for 3D objects classification (cf. Table 4.1). The network has many constrain, because the CNNs are in three dimensions rather than two, the computationally of the model is more complex. This also limits the size of the input grid and hence the difficulty to process high-resolution data.

Maturana and Scherer [83] also use three-dimensional convolutional layers with VoxNet: the network is composed by the input layer, two convolutional

layers, a pooling layer and two fully connected layers. The input grid size is a bit bigger than ShapeNet, with a dimension of $32 \times 32 \times 32$. The experiments show that VoxNet outperforms ShapeNet on ModelNet dataset. [108] proposed a modified version of VoxNet, ORION that includes the orientation of the 3D object in the learning step.

Brock et al. [13] proposed Voxception-ResNet (VRN), a convolutional neural networks inspired by VoxNet but also Inception [123] and ResNet [52] architecture. VRN is composed of 45 layers, most of the layers being 3D convolutional layers like VoxNet. However, VRN offers better performance than the *shallow* VoxNet by improving by more than 50% the classification accuracy on ModelNet (Table 4.1). VRN is until now the state-of-the-art performance on the dataset. However, the depth of VRN makes it prone to overfitting, hence the extensive use of data augmentation during the training step. Hence, the complexity of VRN is a limiting factor for using it.

As seen, deep learning algorithms on volumetric representation are efficient, but the models required an enormous computational power because of the three-dimensional convolution and hence, the significant number of parameters.

4.5.5 Deep Learning on Multi-View Data

By using multi-view data, the aim is to have a model that is less complex than volumetric models. It is also possible to use 2D deep learning paradigms that are well developed.

Xie et al. [136] offer a model that uses 20 multi-view images that are uniformly captured by a sphere at the centre of the object. The convolutional layers shared by all view, i.e. get the same weights.

Su et al. [120] propose Multi-View CNN (MVCNN) that process the different views of the object without any specific order thanks to a pooling layer. MVCNN use 80 views to perform the classification, in a first step, each view is processed independently by a CNN and one view is selected with a

max-pooling layer. This view then feeds to a second CNN with a softmax output layer. Only a few views are contributing to the final representation of the 3D object. MVCNN is pre-trained with ImageNet dataset and fine-tuned with ModelNet dataset. MVCNN outperforms ShapeNet by more than 25%.

Johns et al. [59] use multi-view representation by representing the object under unconstrained camera trajectories with a pair of 2D images. The model classifies each pair independently and gives a contribution for each to get the final result. The model is composed of five convolutional layers and three fully connected layers.

Qi et al. [95] use a multiresolution approach with massive data augmentation. Kanezaki et al. [61] offer state-of-the-art performance with RotationNet. The model takes multi-views images and estimates the pose and the object category, computing its likelihood.

As can be seen in Table 4.1, deep learning algorithm on multi-view representation perform slightly better than volumetric representation model and with lower complexity. The main challenge here is to define the number of views to use and how they are acquired. Moreover, this type of algorithm relies more on 2D image deep learning performances than 3D deep learning model.

4.5.6 Deep Learning on Point Cloud

As a set of an unordered list of point, deep learning on point cloud is fundamentally different from the previous examples.

Ravanbakhsh et al. [99] propose a model that is robust to permutation in the input vector. The layer is obtained by parameter-sharing to lean the permutation invariance as well as a rigid transformation on the data.

Qi et al. [96] is a pioneer for deep learning on point cloud with PointNet. PointNet input layer is directly the set of points represented by their (x, y, z) coordinates. A feature transformation can be done as pre-processing to enhance accuracy. The model is composed of fully connected layers, a max-pooling layer and another set of fully connected layers. Each point is

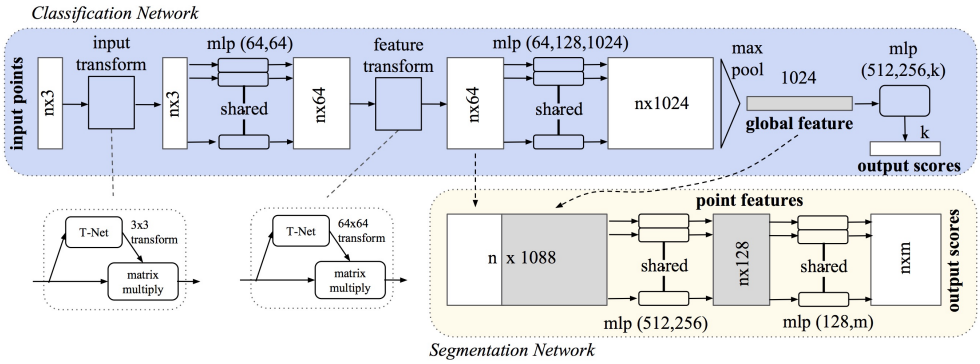


Figure 4.8: PointNet Architecture by Qi et al. [96]. The classification network takes n points as input. A feature transformation is optionally done. The network is composed of five shared multi-layer perceptron. The output is then aggregated by a max-pooling layer giving the *global feature* of the point cloud. The classification is given after another MLP network. A segmentation network is also proposed, combining different layer of the classification network.

processed independently, and the max-pooling layer selects the point with the most activated node. This layer solves the problem of the unordered set and permutation of the input points. After the pooling layer, the network is a simple classification MLP. PointNet is robust again partial data and input perturbation. However, PointNet does not take advantage of the point local structure: each point is processed as if there were no neighbours. PointNet++ by Qi et al. [97] addresses the problem. The points are clustered into local regions, and PointNet is applied in nested partitions of the input points set. More features are computed, but the model is also more complex with more parameters, increasing the computational time.

Li et al. [77] propose SO-Net, a network invariant to permutation which tolerates unordered input point cloud. SO-Net models the spatial distribution of point cloud by building a Self-Organizing Map (SOM). Based on the SOM, SO-Net performs hierarchical feature extraction on individual points and SOM nodes, and ultimately represents the input point cloud by a single feature vector. The model is tested on classification and produce promising results. Yang et al. [138], with FoldingNet, propose to use an autoencoder to model the 3D objects. A folding-based decoder that deforms a 2D canonical

grid into the underlying surface of the 3D point cloud is used. FoldingNet leans how to generate cuts on the 2D grid to create 3D surfaces and generalize to some intra-class variations of the same 3D object category.

Table 4.1: Comparison of 3D Neural Network on ModelNet Dataset.

	Algorithm	Accuracy		Notes
		MN40	MN10	
Voxel	Wu et al. [132]	77	83.5	3D ShapeNets
	Maturana and Scherer [83]	83	92	VoxNet
	Sedaghat et al. [108]	-	93.8	ORION
	Brock et al. [13]	**95.54**	**97.14**	Voxception-ResNet
	Arvind et al. [4]	86.5	-	
	Ren et al. [100]	90.5	-	
C1	Hegde and Zadeh [53]	90.8	93.11	FusionNet
	Su et al. [121]	95.0	-	VoxMVCNN
2D Image	Su et al. [120]	90.1	-	MVCNN
	Johns et al. [59]	90.7	92.8	
	Qi et al. [96]	91.4	-	MVCNN-MultiRes
	Sfikas et al. [110]	90.7	91.1	Panorama-NN
	Wang et al. [129]	93.8	-	
	Arsalan Soltani et al. [3]	82.1	-	
	Kanezaki et al. [61]	**97.37**	**98.46**	RotationNet
	Feng et al. [39]	93.1	-	
	Yu et al. [142]	94.7	95.0	MHBN
C2	You et al. [141]	93.2		PVNet
Point Cloud	Ravanbakhsh et al. [99]	90.0	-	
	Qi et al. [96]	89.2	-	PointNet
	Li et al. [77]	**93.4**	**95.7**	SO-Net
	Qi et al. [97]	91.9	-	PointNet++
	Zaheer et al. [144]	90.3	-	
	Yang et al. [138]	88.4	94.4	FoldingNet
	Yavartanoo et al. [139]	92.63	97.25	SPNet
O	Sinha et al. [115]	83.9	88.4	Geometry Image
	Ben-Shabat et al. [9]	91.6	95.2	3DmFV-Net

MN40: ModelNet40, MN10: ModelNet10
C1: combination of voxel and image representation as input.
C2: combination of image and point cloud representation as input.
O: other type of input.

Chapter 5

Classification of 3D Point Cloud

5.1 Classification with 3D Descriptor

In this section, MPC2 descriptor is tested for classification of ModelNet dataset. It is a challenging task, as MPC2 was not designed for this. Indeed, MPC2, like other descriptors, only describes one type of feature of the object, here the curvatures.

Here, MPC2 descriptor is used into an SVM and a simple MLP to test if it is discriminating enough to be used for classification.

5.1.1 First Approach: SVM with MPC2 Descriptor

A first approach to classify ModelNet dataset is tested by using our previous geometric descriptor as vectors for an SVM algorithm. Geometric descriptor studied in the previous chapter are signatures for each object. Hence, we want to check if there is a *simple* relation or similarity for all instance of the same object.

Implementation

For this study, MPC2 geometric descriptor is tested for instance retrieval. The dimension of the space is then the size of the descriptor, i.e. 10000. The descriptor for an object i is noted $\mathbf{X}_i$, $\mathbf{X}_i$ is a vector of size d, d being the size

of the descriptor. Then dataset can be noted $\mathbf{X}$, $\mathbf{X} = \{\mathbf{X}_i\}$ with $i \in [\![0; M]\!]$, M being the number of object in the dataset. X is then a matrix of size $M \times d$ which represents the dataset. More specifically, the train, validation and test dataset are respectively noted $\mathbf{X}_{\text{train}}$, $\mathbf{X}_{\text{validation}}$ and $\mathbf{X}_{\text{test}}$ the matrix for the train, validation and test dataset. Their sizes are respectively $M_{\{\text{test, train, validation}\}} \times d$ with $M_{\text{train}} + M_{\text{validation}} + M_{\text{test}} = M$.

The optimal parameters of the model for the dataset are computed with a grid search. The Gaussian kernel (Section 4.2.2) is selected, and the parameters γ (Eq. 4.14) is searched in $[\![10^{-8}; 10^{4}]\!]$. The soft margin parameters C (Section 4.2.3) is to be found in the same interval $[\![10^{-8}; 10^{4}]\!]$. Fine-Grid research is then done according to the results returned by the primary research. Ultimately, a one-versus-all strategy is adopted for the problem.

5.1.2 CurvNet

Here, MPC2 descriptor is used as an input of a more sophisticated machine learning algorithm: a multi-layer perceptron. As mention previously, an MLP is a robust classification algorithm, able to handle non-linear separable data. One more time, the test aims to verify if the previously computed geometrical descriptors are discriminating enough to classify ModelNet dataset.

Implementation

This approach uses multi-layers perceptron networks to classify MPC2 descriptor. Different architectures are tested, each one with different numbers of layers and nodes, c.f. Table 5.1. ReLu activation function (Equation 4.18) is used in addition of common regularization techniques like dropout or batch normalization. Because the output layer is a softmax layer, the categorical cross-entropy loss function (Equation 4.2) is used to estimate the accuracy of the training step.

The neural network designed takes the MPC2 geometric descriptor of a point cloud for input. The output is the predicted class. The architecture of the network is simple and consists of a series of MLP layers.

Table 5.1: Configurations tested for CurvNet. On the left, the config number and on the right, the number of nodes for this config. k is the number of categories.

#	# nodes
0	[64, 64, k]
1	[128, 128, k]
2	[64, 64, 64, k]
3	[128, 128, 128, k]
4	[64, 64, 64, 64, 64, k]
5	[128, 128, 128, 128, 128, k]

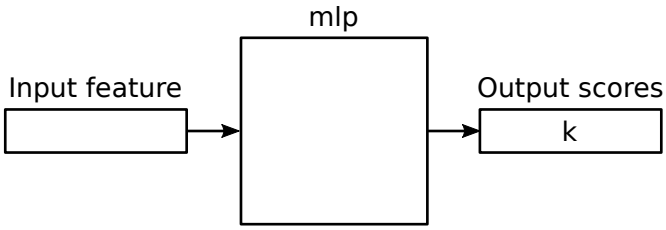
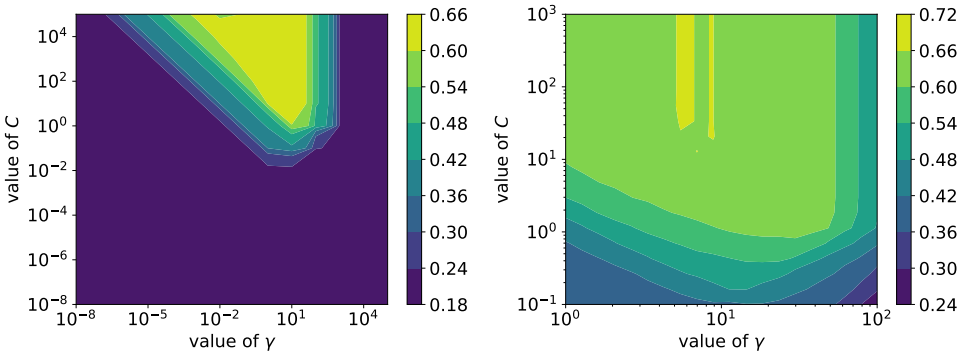


Figure 5.1: CurvNet Architecture. The input layer is the MPC2 features, and the MLP layers are defined with the configuration presented in Tab 5.1. The output layer is the score of each of the k categories.

5.1.3 Experimental Results

The best models are selected after an extensive research of the optimal parameters. For the SVM algorithm, the tests of the parameters can be seen on Figure 5.2 and Figure 5.3 show the performances of different models explored for CurvNet. The best parameters for the SVM are for $C = 33.60$, and $\gamma = 5.46$, with these parameters, the accuracy on the train set is 0.66

On the other hand, many architectures are tested for CurvNet, the best being the model #1: a 2-hidden layers MLP with 128 nodes for each. Indeed, this model is the fastest, 35 epochs only, to reach an accuracy of about 60% for both the train and validation set. The others need at least the double number of epochs without offering better accuracy.

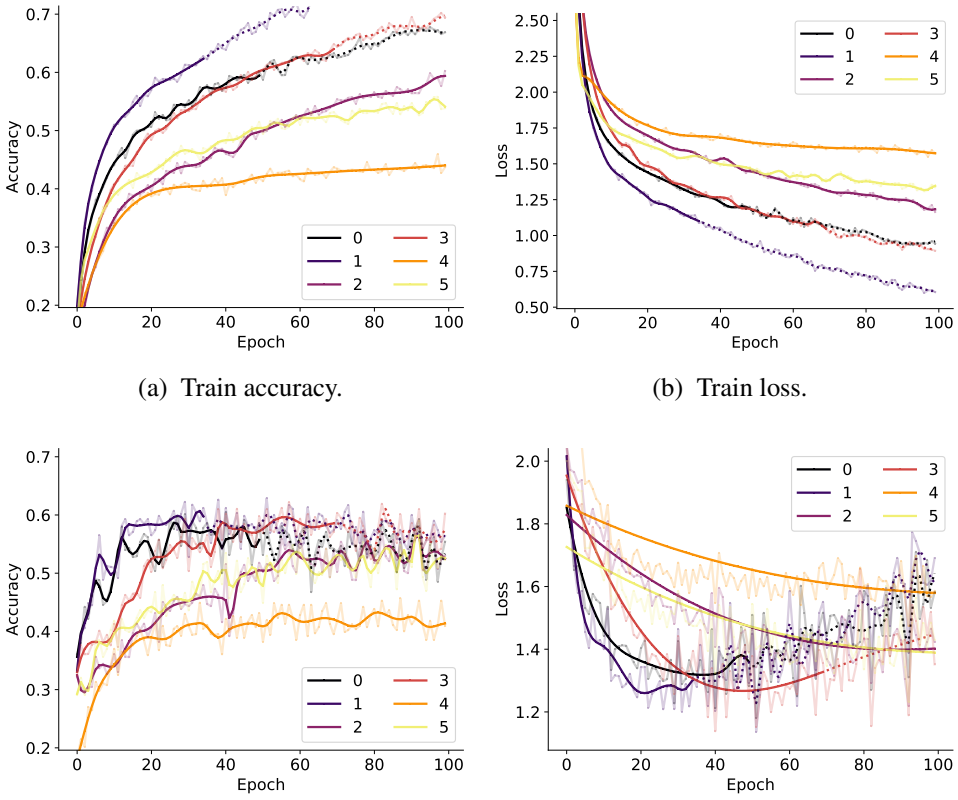


(a) (b)

Figure 5.2: Research of the optimal parameters for the SVM on the train set. Visualisation of the accuracy of the SVM on MPC2 descriptor depending on C and γ parameters. Best train accuracy is 0.66 with $C = 33.60$ and $\gamma = 5.46$. (a) shows the accuracy with coarse variation of the parameters. (b) show a fine search of the best parameters

Table 5.2 and Table 5.3 show the results of the different tested algorithm. The results are quite similar with an accuracy of 0.60 for the SVM, CurvNet performing a bit lower with 0.53. The precision and recall of both algorithms are quite related: a low score for a given label with the SVM also give a low score with CurvNet. As can be seen, the objects labelled 0 are not well classified by both, but CurvNet gets particularly low performance with only 0.02 as recall. On the other hand, the model performs well on objects labelled 2 and 7, returning better values than the SVM.

Other remarks can be made from the two Tables: the average precision is 0.62 and 0.56 for the SVM and the MLP respectively. This value is however push up by the particularly good precision for objects labelled 9. For the other categories, the values are lower. Ultimately, with CurvNet, categories 3, 5 and 6 have high precision and low recall values, meaning that only a few of the objects of the class has been appropriately classified. On the contrary, categories 2 and 7 have the opposite behaviour: many objects were misclassify into these categories. The SVM model has the same behaviour but in a lower proportion.



(a) Train accuracy.

(b) Train loss.

(c) Validation accuracy.

(d) Validation loss.

Figure 5.3: Research of the best architecture and parameters for CurvNet on the train and validation set. The lines are smoothed for better visualisation, and the real data are represented in light colours. In plain colours, the valid range of training, in dotted, the model overfit. The overfitting epoch, is detemined with the graph on (d). As can be seen, model 2, 4 and 5 are very slow to converge. Model 1 and 2 have a similar behaviour, but 2 is faster. At the end of the training, around the 35^{th} epoch for the model 1, the train and validation accuracy are both around 0.6.

Table 5.2: Performance of the SVM with Descriptor-Based Feature. Parameters of the SVM selected with previous study, $C = 33.60$ and $\gamma = 5.46$.

Label	Precision	Recall	Support
0	0.55	0.24	50
1	0.50	0.69	100
2	0.50	0.79	100
3	0.66	0.44	86
4	0.69	0.53	86
5	0.58	0.57	100
6	0.63	0.48	86
7	0.66	0.67	100
8	0.54	0.60	100
9	0.92	0.79	100
avg/total	**0.62**	**0.60**	908

Table 5.3: Performance of CurvNet. The model selected is the model number 2 (c.f. Table 5.1).

Label	Precision	Recall	Support
0	0.50	0.02	50
1	0.48	0.51	100
2	0.39	0.84	100
3	0.49	0.35	86
4	0.67	0.59	86
5	0.61	0.25	100
6	0.64	0.34	86
7	0.46	0.84	100
8	0.54	0.49	100
9	0.84	0.77	100
avg/total	**0.56**	**0.53**	908

5.1.4 Discussion

SVM on Descriptors

The results presented in the previous section are goods. However, the analysis has to be pushed forward and the results, moderated. Two remarks can be made:

(i) The accuracy between the train set and the test set are similar.

(ii) The accuracy is relatively good.

Even if 0.60 is not a particularly high accuracy, it is still too high when the results are examined more closely. Indeed, a look at the number of support vectors in Table 5.4 indicates that the SVM relies too much on the train set. There are 3192 objects inside, and 2719 are used as support vectors, which represents a ratio of 0.85. This is way too much.

The pair of optimal parameters (C, γ) is computed with a grid search and also cross-validation on the train set. The accuracy on the train set is 0.60, the one on the test set is 0.66: the two are quite similar. This indicates the generalisation of the algorithm is right, that elements of the train set are a good representation of the test set. The question is then: why so many support vectors are required by the algorithm. The answer can be found in the characteristic of the descriptors.

As described in Part I, the descriptors relies on the curvatures of the 3D object. Given to the SVM, they are not discriminative enough for a proper partition of the dataset. Hence, this indicates that the study of the curvatures of the object is not enough to correctly classify this dataset, whereas the good performances in the previously tested dataset.

A look at the object in the dataset seems to agree with this. Indeed, one can note that the objects are “flats”, most of them are only an assembly of plane structures. The objects do not hold any curvatures. This lead to an MPC2 descriptor in the shape of a Dirac delta function as the distribution of curvature is limited to a value close to 0 because of the flatness.

Table 5.4: Support Vector (SV) of the SVM algorithm. The ratio support vector on support (SV/S) very high with an average value of 0.85.

	SV	**S**	**SV/S**
0	78	81	0.96
1	367	387	0.94
2	530	736	0.72
3	156	159	0.98
4	143	163	0.88
5	357	371	0.96
6	166	167	0.99
7	433	546	0.79
8	288	318	0.91
9	201	264	0.76
avg/total	2719	3192	0.85

CurvNet

A similar analysis can be done with CurvNet. The performance during the test stage follows the one during the training, but the average performances are still low compared to other state-of-the-art algorithms.

However, although theoretically *better* than the SVM, CurvNet has lower accuracy. This performance finds its answer in the use of regularisation techniques. MLP architectures are more complex and include efficient methods to tackle overfitting. Usually, regularisation restricts the learning of the feature by the network to get better generalisation performances. In this particular case, dataset partition between the train and test set seems good as validated by the results of the SVM and also by the accuracy CurvNet on the train and test set. Here, the regularisation limits the overall learning of the feature since the descriptors of each class are similar. Hence the lower value than the SVM for which such evaluation techniques do not exist.

5.2 Robustness of Deep Learning Algorithm to Noise

As shown in the previous section, the objects in ModelNet are not a good representation of the real objects as very flat. In this section, the state-of-the-art's deep learning methods are tested with more realistic 3D objects model. Because large dataset of real scanned objects does not exist, the algorithms are tested for instance retrieval with the dataset of the first part.

In a first experiment, the performance for occluded models is tested while a second one deals with noise corruption robustness.

5.2.1 Algorithm and implementation

MPC2 MPC2 is used with the configuration previously presented. 5 scales of SPC are concatenated, and the Chi-2 distance is used to compare the descriptor [74]. Each model to retrieve is compared to the references dataset. The closest is considered as the retrieved object. Since SPC descriptor takes the form of a histogram distribution, the Chi-2 distance is particularly adapted to describe the similarities between them.

SPC Two scales of SPC are used for the comparison, the smallest and the biggest used by MPC2. Like MPC2, Chi-2 distance is used.

SHOT SHOT [106] is used in combination with ISS [145] keypoints algorithm. The default parameters are used for both. SHOT descriptor is a signature represented by a vector of size 352 and comparison is done with L1 distance.

PointNet PointNet [96] is designed for classification of 3D model. While it does an excellent job in this task with 89.2% accuracy rate on ModelNet40, there is no report about performances with corrupted data. Because PointNet and MPC2 have to be compared in fair condition, PointNet is pre-trained on ModelNet10 and uses the layer called *Global Feature* (c.f. Figure 4.8) as the

Figure 5.4: Examples of occluded models. The occlusion ratio of these models varies between 10% and 80%.

descriptor for the tested model. The 16 references objects are given to the network to get the 16 reference descriptors. Then, each model to retrieve is fed to PointNet, and the output of the *Global Feature* layer is compared with each of the references. L1 distance is used this time again.

It is important to note that PointNet is not further trained with the 16 reference models of our dataset. Indeed, MPC2 and other methods compared here are not trained on these 16 reference models: the descriptors are only computed and compared. The expectation is that PointNet has learned the 3D object shape representation from ModelNet object categorization. Therefore, the 16 descriptors computed hold the properties of their respective model.

5.2.2 Experimental Results

Evaluation of Robustness to Occlusion

In this part, the performances of the algorithms are evaluated for occlusion. The aim is to verify if the algorithms can retrieve the original object from a partial vision of the latter. The test dataset is composed of non-full objects generated by splitting the reference model by a random plane. The splitting plane is chosen randomly but ensures the occlusion ratio varies from 10% to 90% of the initial model. 20 different instances are generated for each reference and each ratio.

The results are shown in Figure 5.5. All algorithms performed quite well, most of them have more than 75% of retrieval rate when only 30% of the model is visible. Above this value, the retrieval rate of SPC decreased below 50%. On the other hand, if more than half of the object is visible, there is

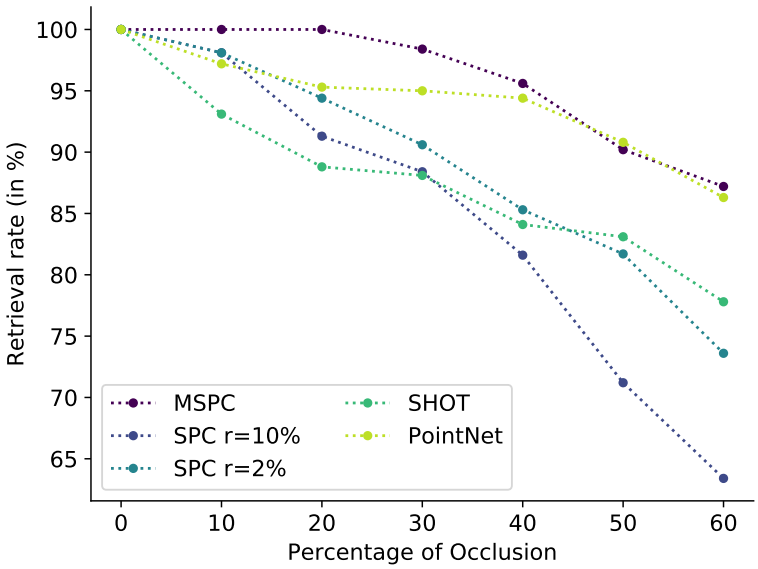


Figure 5.5: Variation of the retrieval rate with the visible ratio.

more than 90% chance to retrieve the object with MPC2, SHOT and PointNet. One can note that MPC2 offers slightly better performance than PointNet. For small occlusion SHOT is overtaken by the latter two but outperforms them when only a small part of the object is visible. SPC with r=10% drops quickly, with r=2% the tendency is the same but less marked.

Evaluation of Robustness to Noise

To evaluate the noise robustness, the dataset is expanded by adding a centred Gaussian noise to the original 16 references models. Since the models come from diverse sources, they are normalized into the unit sphere and then the noise is added. Depending on the technology used, the accuracy of the scan varies between 0.25 mm for 3D scanners [76], 0.3 cm at a distance of 1 m (typically used range) for a Kinect [63] (up to 1 cm at 2.5 m). LiDAR error is more significant, state-of-the-art device: Velodyne HDL-64E claims an error up to 1.3 cm [44]. However, such kind of sensor is mainly used for outdoor mapping and not for object scanning. With the previous observations, the noise is limited to 1 cm, i.e. 1% of the size of the object. In the end, there is

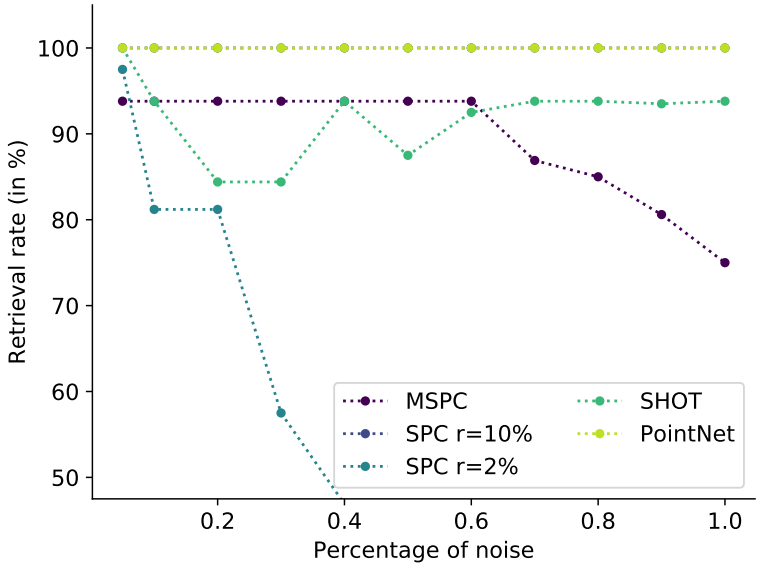


Figure 5.6: Variation of retrieval rate with the noise ratio. PointNet and SPC r=10% lines are overlapping.

1920 noisy models to test.

In this case, Figure 5.6, PointNet and SPC with r=10% are excelent with no mismatch errors. On the opposite, MPC2 sees its performances declining as soon as the noise is higher than 0.70%. SPC with r=2%, contrary to r=10% sees its performances collapse from 0.2% (ie. 2 mm) noise while SHOT remains constant. In the case of PointNet, a noise up to 6% (not shown in the figure) had been tried and the retrieval rate remaining on top with 100%.

5.2.3 Discussion

The first thing to note is the excellent performance of the algorithms. The overall retrieval rate for occlusion is above 80% even with large occlusion and PointNet even seems immune to noise.

The performance of PointNet on occlusion can be explained by the compulsory subsampling. Indeed, the default configuration only uses 1024 points of the 3D model for the network input. These points being selected randomly throughout the training and the testing step beside a large number of points in

the original models: the effect of occlusion is then minimized. For the same reason, PointNet is robust to downsampling. However, the generalization capabilities of PointNet are such that even with a small number of points, i.e. a prominent occlusion, the *Global Feature* layer can retrieve the properties models and return a descriptor closed to its reference.

On the other hand, the performance of PointNet with noisy data is excellent even when compared with MPC2. The noise applied is nevertheless smaller than the one applied during data augmentation –ie. 2‰– in the training of PointNet.

Results of MPC2 are more expected since it is a descriptor relying on geometrical properties. Performances of MPC2 should be analyzed beside SPC as an enhancement of the latter. As can be seen on Figure 5.5 and Figure 5.6, the effect of the radius results in a trade-off on the performance robustness when the deformation is an occlusion or a noise addition. MPC2 manages to keep the performances of the best radius and does not collapse when SPC does.

5.3 Multi-scale PointNet

Because the curvature-based algorithms developed cannot provide good performances for classification the existing dataset, a new algorithm is presented here with Multi-scale PointNet. The idea is to propose a classification algorithm that operates directly on 3D point cloud and relies on different observation scale. Indeed, when a person looks at an object, the observation can be global by looking at the whole object, but it can also be partial when the eyes focus on one part of the object. By combining all observation of the different scales, a person is then able to recognize an object. For instance, if a person looks to a ball from far (i.e. large scale), the only information is that it is a spherical object. However, looking closely (i.e. small scale), one can figure if it is a football or a basketball thanks to its roughness.

Multi-scale PointNet aims to mimic this behaviour. For each scale, a feature is computed to describe the most representative part. The features from the different scale are then gathered to characterize the object.

5.3.1 Algorithm

In Multi-scale PointNet, Figure 5.7, the feature of the scales are computed with a multi-layer perceptron neural network. MLP layers for point clouds proved their efficiency for classification tasks [96]. Multi-scale PointNet relies on multiple multi-layer perceptrons neural network. Each scale gets its own, but they share the same architecture, and only the weights are different, adapted to the scale, Figure 5.7c. Because many regions are considered for one study scale, a feature is computed for each region. In the end, only one of these features is selected as the *scale feature*, Figure 5.7b. The latter represents the regions, which is the most distinctive from the other for this study scale.

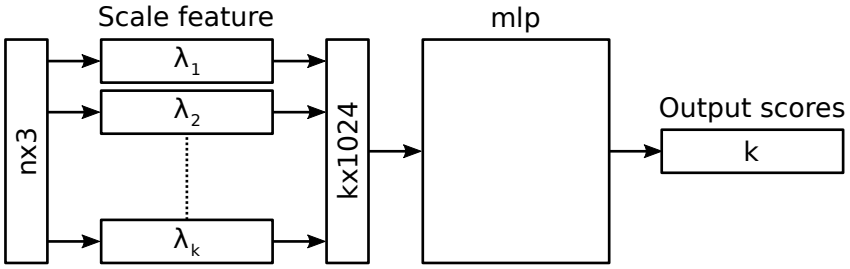
Before this step, the regions have to be selected according to the scale and the number of centroids wanted. A region is a local point cloud, a sub-part of the original point cloud, describing one part of the object. They are composed of the points of the original point cloud which are inside a local neighbourhood. This neighbourhood is defined by the centroids, the centres of the regions and the study scale. Hence, the number of centroids and the scale must evolve oppositely. A large number of centroids with a big scale is useless while a small number with a small scale does not cover the full point cloud.

Finally, once the features describing each scale computed, Figure 5.7a, they are all merge and the resulting *global feature* is the input of another classification network. A simple MLP network is used with a softmax layer as output to return the probabilities for each category.

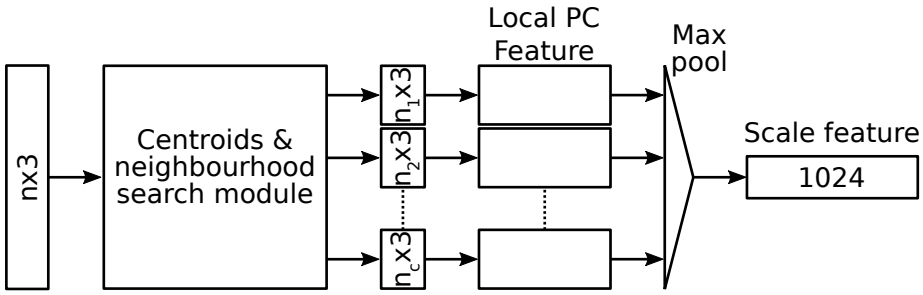
5.3.2 Implementation

Local Point Cloud

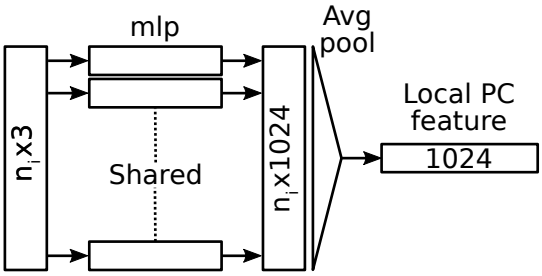
The number of scales is a parameter set by the user. For this study, the scales are limited to 3 values: 0.25, 0.5 and 1. The number of regions for each scale is another parameter set by the user, with 27 for the smallest scale, 9 for the medium and 1 for the largest (i.e. the whole object). The regions are



(a) Multiscale PointNet Architecture. The input is the point cloud, the scale features are computed and concatenated. Then an MLP network is used to output the prediction scores.



(b) Scale feature module. For a given scale, the centroids and their respective neighbourhood are computed. Each of the local point clouds is then the input of the local feature module. A max-pooling of all the local feature is done to output the scale feature.



(c) Local feature module. For each local point cloud, a feature is computed with an MLP network. The problem of the unordered and unsorted structure is solved with an average pooling layer. The output is the feature of the local point cloud The local module is shared by all the local point cloud of a same scale.

Figure 5.7: Multiscale PointNet Architecture. The network is described with the two submodules: the scale feature and the local feature module.

distributed in a regular three-dimensional grid.

Another method is to select the centroids randomly, but this also implies to increase their number. By increasing the number, the probability of overlapping between the region increase compensating the effect of the random choice. However, this method drastically increases the computational need as much more regions has to be processed.

For a given number of region n, a selection of the centroids with the furthest sampling algorithm can also be considered. In this algorithm, the n furthest points from each other are found and used as centroids. This algorithm is used by Qi et al. [97] in PointNet++.

During this step, all resulting regions are likened to independent and centred point clouds.

Scale Features

The computation of the scale feature is very similar to what it is done in the presented algorithm of the previous section. Indeed, in this part, the feature of a region is computed. This region being a point cloud, the algorithm can rely on existing research. There are as many networks than the number of study scale; all of them are MLP layers with the same architecture. However, with different weight as not trained with the same input data.

The MLP is composed of 5 layers, the first three with 64 nodes, the forth hold 128 nodes connected to the 1024 nodes of the fifth layer. Each point of a point cloud is processed independently. Once all the point processed, the average of all fifth layer output is computed as the point cloud feature. And an averaging function is selected to bypass the problem of the unordering structure of the data but also because the research must be representative of the whole region. For each scale, there is then as many features as regions: only one is selected as the *scale feature*. Because the most distinctive is looking for, the solution is to select the most significant feature, the one that activates the more nodes.

Global Feature

The *global feature* is the feature representing the whole object for different scales. Each scale being described by one scale feature, these latter are combined to define the global feature. If there is k scale, the size of the feature is then $k \times 1024$.

The global feature is then the input of another classification MLP network composed of three hidden layers and the output softmax layer.

Alternative Network

An alternative, version of Multi-scale PointNet can also be tested. Indeed, the algorithm presented previously is expensive in computational resources and in time as there is a significant number of parameters to find. Hence, this alternative aims to reduce the training time.

In this version, the difference lays in the computation of the scale feature. The number of scales and regions per scale remain unchanged, only the way the scale features are computed is. Only one network is used, shared by all the scale and layers. This network, like the previous ones, computes the feature of each point independently, these features are then reused to compute the different scale features. Then for a given scale, and a given region, the algorithm is reduced to the simple computation of the mean of all the feature of the point in the region.

To sum up: one network is used to compute a feature for each point of the point cloud. Then the search region bloc gathers the feature of the points whether they are in the region or not. For each region of a given scale, an average of the feature of the points is computed, and the region with the *biggest* feature is selected as the scale feature. The global feature is then defined as previously and the classification network also.

Summary of Part II

This part focused on the classification problem of 3D point clouds models. The research is expending quickly as new deep learning paradigms step in with models running directly on raw point clouds. The most recent algorithms start exploring a multi-scale analysis of the objects, but the main drawback is the heavy computation required.

The standard dataset for classification benchmark is ModelNet [132]. The dataset offered two large sub-datasets for the benchmarking of the algorithms. However, the study with MPC2 descriptor shows the dataset is made of an association of plane that does not reflect the wealthiness of curve of reals objects.

Deep learning algorithms, PointNet particularly, are then tested with real scanned objects for instance retrieval. Indeed, no large dataset of real scanned objects for classification can be found. The performances are excellent as training on ModelNet can be used for the instance retrieval, with good accuracy for common recording errors, bearing out the performances of the model.

Finally, a new deep learning architecture is proposed to classify ModelNet dataset. Relying on existing and efficient DL network, the aim is to use the multi-scale feature to improve the performance of existing algorithms. The preliminary tests are, however, not satisfactory and do not challenge existing algorithms yet. More research on the optimal parameters has to be done. However, by lack of time, this will be left for another research team.

Chapter 6

Conclusion and Perspectives

This thesis focused on the use of 3D point cloud representation for three-dimensional data. 3D data becomes more and more popular with the improvement of sensors and computational capability of computers. This point cloud representation has many advantages; for instance, it is the direct output of Lidar sensors, used in many applications like autonomous vehicles or scenes mapping. The data format is also simple as an unordered list of points coordinates. However, this particular advantage is also a handicap: the non-regular structure makes the data challenging to analyze. Because of all this, the use of 3D point clouds is recent, and it became an active research topic for the past few years only.

In the first part of the thesis, we studied the geometric properties of the point cloud. We proposed SPC, a new and efficient descriptor describing the shape of the object. More specifically, our descriptor summarizes the distribution of the curvature in the object, telling how much the object is plane or not. An enhancement is proposed with MPC2, a multi-scale version of SPC. With MPC2, it is possible to *see* the object from different distances: get the global but also the precise shape of the object. GLPC proposes another approach combining two descriptors: one for the local feature and SPC for the global shape of the object. One more time, the idea is to analyze the object at different scales.

Because they are handcraft algorithms and because they were used in

a context where not much datasets existed, these algorithms are tested for an instance retrieval problem. The performances are goods, the algorithms being able to have more than 90% of overall retrieval rate, being better than state-of-the-art algorithms for common recording error deformation. While SPC and MPC2 are robust to noise and downsampling, GLPC gets the best result with 98.3%: combining two descriptor returns the best of them.

The second test is performed on a dataset of deformable objects. This time, the dataset represents the same object in different postures. GLPC is one more time able to beat all state-of-the-art algorithms with more than 98% of retrieval rate.

The results of our algorithms for instance retrieval are promising, but progress in deep learning opens up new research perspective. This is the purpose of the second part of this thesis. In this one, deep learning techniques are used with 3D point cloud data to address the classification problem.

First, we tried to reuse the geometrics descriptors with neural networks to test the performances in classification. These tests did not return conclusive accuracy results but highlight a crucial problem with current 3D objects classification dataset like ModelNet. Indeed, they show that SPC descriptor and its variants are not adapted to ModelNet. The main reason is because the dataset does not reflect the curvature wealth of real object. As a CAD dataset, it is logical that the objects are simplified. However, ModelNet completely removes the curve of the objects making the dataset unrealistic as a combination of planes.

In a second time, the performances of deep learning models are tested with a real 3D objects dataset. The training on ModelNet is still efficient for this dataset, the models being also robust toward noise corruption or occlusion. The models proved their efficiency on a small but unseen dataset.

Ultimately, we propose a new neural network architecture for 3D objects classification. This new architecture is a multi-scale approach that wants to collect the feature of the object for different scale of observation. Relying on existing models, the approach is inspired by novelties in 2D deep learning paradigm. Still in development, the model is not fully effective and required

some adjustments to return a good accuracy.

This thesis took place in a tipping point for 3D analysis and mainly 3D point cloud analysis. The research shows that previously designed descriptors are still efficient for tasks like instance retrieval and have the advantage to be able to run with very few data. However, with the advances in deep learning, many new promising networks on point cloud are proposed for more complex tasks like classification or segmentation of 3D objects.

As the usage of 3D data and more particularly of point cloud intends to be more an more popular, it is essential to dispose of a large annotated and authentic dataset of 3D point cloud objects. Indeed, the research and advance in 2D image classification were possible thanks to the ImageNet project, which started in 2006 and gathers more than 14 million of annotated images. The research in 3D deep learning architecture will undoubtedly benefit from the creation of such dataset. Of course, ShapeNet and its variations like ModelNet is a first step, but as shown, the 3D objects in the dataset are CAD and not always a realistic representation of the reality. Obviously, the creation, gathering and annotation of 3D objects is still more complicated than for 2D data. A simple smartphone is enough to create and upload an image, while the use of CAPTCHA [137] for annotation tools is common. CAPTCHA was first used for text annotation and then for image annotation. Once a 3D dataset of objects available, it will not be surprising to adapt the CAPTCHA system again to annotate 3D objects. Even if 3D sensors are more common and easier to install, the creation of a real 3D scan dataset is still the significant challenge [23]. So project and research to facilitate the creation of point cloud dataset should also be promoted.

Another research perspective would be to adapt existing and efficient 2D deep learning advancement to 3D point cloud deep learning models. Many progress has been made since the beginning of the use of neural networks on images classification while proposed models for 3D point cloud can be qualified as shallow in comparison. VGG, ResNet or InceptionNet are architectures that proved their efficiency for images classification. They rely on convolution neural networks that are not in use in the models proposed to

classify 3D point cloud. The idea, explored by Multi-scale PointNet architecture, would be to use MLP layers as the convolution layers. The size of the kernel is then simulated by feeding the MLP a point cloud restricted to the local neighbourhood of the points. The behaviour of efficient working models for images can then be replicated. However, this approach implies heavy computations as an MLP layer has much more parameters than a simple convolution layer.

This manuscript sum up the last three years spent in IPAL lab on the study of 3D point cloud. We proposed new geometric descriptors and archived good results, for instance, retrieval of point cloud. Then we led a study on new deep learning network on point cloud and proposed new architecture. The thesis was challenging as more and more research are lead on the field and also because many research axis can be taken.

List of Publications

International Conferences

[1] Justin Lev, Joo Hwee Lim, Ouarti Nizar, and Mokhtari Mounir. Towards robust retrieval for imperfect scanned point cloud object. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019

[2] Justin Lev, Joo Hwee Lim, and Nizar Ouarti. Principal curvature of point cloud for 3d shape recognition. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 610–614. IEEE, 2017

Bibliography

[1] Elsa Abbena, Simon Salamon, and Alfred Gray. *Modern differential geometry of curves and surfaces with Mathematica*. Chapman and Hall/CRC, 2017.

[2] Luís A Alexandre. 3d object recognition using convolutional neural networks with transfer learning between input channels. In *Intelligent Autonomous Systems 13*, pages 889–898. Springer, 2016.

[3] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1511–1519, 2017.

[4] Varun Arvind, Anthony Costa, Marcus Badgeley, Samuel Cho, and Eric Oermann. Wide and deep volumetric residual networks for volumetric image classification. *arXiv preprint arXiv:1710.01217*, 2017.

[5] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5023–5032, 2016.

[6] Dale L Bailey, Michael N Maisey, David W Townsend, and Peter E Valk. *Positron emission tomography*. Springer, 2005.

[7] D Bassily, C Georgoulas, J Guettler, Thomas Linner, and T Bock. Intuitive and adaptive robotic arm manipulation using the leap motion

controller. In *ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1–7. VDE, 2014.

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[9] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks. *arXiv preprint arXiv:1711.08241*, 2017.

[10] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[11] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[14] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[15] Michael M Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1704–1711. IEEE, 2010.

[16] Shuhui Bu, Pengcheng Han, Zhenbao Liu, Junwei Han, and Hongwei Lin. Local deep feature learning framework for 3d shape. *Computers & Graphics*, 46:117–129, 2015.

[17] John Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.

[18] James Casey. *Exploring curvature*. Springer Science & Business Media, 2012.

[19] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005.

[20] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[21] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3d mesh segmentation. In *Acm transactions on graphics (tog)*, volume 28, page 73. ACM, 2009.

[22] Daniel Chernoff and Paul Stark. Principles of magnetic resonance imaging. *UpToDate. Waltham MA: UpToDate. Retrieved February*, 2010.

[23] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv preprint arXiv:1602.02481*, 2016.

[24] Nicholas H Cohrs, Anastasios Petrou, Michael Loepfe, Maria Yliruka, Christoph M Schumacher, A Xavier Kohll, Christoph T Starck, Marianne Schmid Daners, Mirko Meboldt, Volkmar Falk, et al. A soft total artificial heart—first concept evaluation on a hybrid mock circulation. *Artificial organs*, 41(10):948–958, 2017.

[25] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[26] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013.

[27] Thomas M Cover, Peter E Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[28] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[29] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[30] Abou de Souffle, Jean Neymar, and Babor Lelefant. À déterminer. In *Le Meilleur de Carambar*, page 42, 1998.

[31] Frédéric Delbos and Jean Charles Gilbert. *Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems*. PhD thesis, INRIA, 2003.

[32] Claude Deschamps, Francois Moulin, Andr'e Warusfel, Cleirec Nathalie, and Yoann Gentric. *Math'ematiques tout-en-un MP-MP**. J'int'egre, Dunod, 2016.

[33] Manfredo Perdigao Do Carmo and Manfredo Perdigao Do Carmo. *Differential geometry of curves and surfaces*, volume 2. Prentice-hall Englewood Cliffs, 1976.

[34] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[35] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687. IEEE, 2015.

[36] Leonhard Euler. Recherches sur la courbure des surfaces. *Memoires de l'academie des sciences de Berlin*, pages 119–143, 1767.

[37] Yi Fang, Jin Xie, Guoxian Dai, Meng Wang, Fan Zhu, Tiantian Xu, and Edward Wong. 3d deep shape descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2319–2328, 2015.

[38] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

[39] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–272, 2018.

[40] Aaron Fenster, Jeff Bax, Hamid Neshat, Derek Cool, Nirmal Kakani, and Cesare Romagnoli. 3d ultrasound imaging in image-guided intervention. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6151–6154. IEEE, 2014.

[41] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[42] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[43] Jason Geng. Structured-light 3d surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, 2011.

[44] Craig Glennie and Derek D Lichti. Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning. *Remote Sensing*, 2(6):1610–1624, 2010.

[45] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[46] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3d object recognition in cluttered scenes with local surface features: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.

[47] Richard HR Hahnloser and H Sebastian Seung. Permitted and forbidden sets in symmetric threshold-linear networks. In *Advances in Neural Information Processing Systems*, pages 217–223, 2001.

[48] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405 (6789):947, 2000.

[49] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[50] Hartigan Shea, Rachel. Historian uses lasers to unlock mysteries of Gothic cathedrals. `https://news.nationalgeographic.com/2015/06/150622-andrew-tallon-notre-dame-cathedral-laser-scan-art-history-medieval-gothic/`, April 2019.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[53] Vishakh Hegde and Reza Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016.

[54] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[55] Gabor T Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.

[56] Günter Hetzel, Bastian Leibe, Paul Levi, and Bernt Schiele. 3d object recognition from range images using local feature histograms. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2001.

[57] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[58] Haiyang Jin, Qing Chen, Zhixian Chen, Ying Hu, and Jianwei Zhang. Multi-leapmotion sensor based demonstration for robotic refine tabletop object manipulation task. *CAAI Transactions on Intelligence Technology*, 1(1):104–113, 2016.

[59] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3813–3822, 2016.

[60] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[61] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018.

[62] Hyun-Wook Kang, Sang Jin Lee, In Kap Ko, Carlos Kengla, James J Yoo, and Anthony Atala. A 3d bioprinting system to produce human-scale tissue constructs with structural integrity. *Nature biotechnology*, 34(3):312, 2016.

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

[64] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3d surf for robust three dimensional classification. *Computer vision–ECCV 2010*, pages 589–602, 2010.

[66] John R Koza and John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[67] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 313–324. ACM, 1996.

[68] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[69] Harold W Kuhn and Albert W Tucker. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer, 2014.

[70] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784*, 2018.

[71] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1): 15–17, 1976.

[72] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[73] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[74] Justin Lev, Joo Hwee Lim, and Nizar Ouarti. Principal curvature of point cloud for 3d shape recognition. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 610–614. IEEE, 2017.

[75] Justin Lev, Joo Hwee Lim, Ouarti Nizar, and Mokhtari Mounir. Towards robust retrieval for imperfect scanned point cloud object. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019.

[76] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, et al. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th annual conference on*

Computer graphics and interactive techniques, pages 131–144. ACM Press/Addison-Wesley Publishing Co., 2000.

[77] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018.

[78] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019.

[79] Zhenbao Liu, Shaoguang Chen, Shuhui Bu, and Ke Li. High-level semantic feature for 3d shape based on deep belief networks. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[80] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[81] Manu S Mannoor, Ziwen Jiang, Teena James, Yong Lin Kong, Karen A Malatesta, Winston O Soboyejo, Naveen Verma, David H Gracias, and Michael C McAlpine. 3d printed bionic ears. *Nano letters*, 13(6): 2634–2639, 2013.

[82] Melvin Earl Maron. Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, 8(3):404–417, 1961.

[83] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[84] Tom McKay. This adorable girl's amazing story shows the real promise of 3-d printing. `https://www.mic.com/articles/100350/this-adorable-girl-s-amazing-story-shows-the-real-promise-of-3-d-printing`, October 2014.

[85] Steven McKenzie. Handy work: The making of hayley fraser's new hand. https://www.bbc.com/news/uk-scotland-highlands-islands-29473906, October 2014.

[86] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 28(10):1584–1601, 2006.

[87] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[88] Vladimir Mironov, Thomas Boland, Thomas Trusk, Gabor Forgacs, and Roger R Markwald. Organ printing: computer-aided jet-based 3d tissue engineering. *TRENDS in Biotechnology*, 21(4):157–161, 2003.

[89] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. *arXiv preprint arXiv:1812.02713*, 2018.

[90] Leap Motion. Blocks, create and interact with shapes, manipulate gravity and more. https://gallery.leapmotion.com/blocks/, February 2016.

[91] Katta G Murty and Feng-Tien Yu. *Linear complementarity, linear and nonlinear programming*, volume 3. Citeseer, 1988.

[92] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011.

[93] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1): 62–66, 1979.

[94] Judith MS Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[95] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[96] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2): 4, 2017.

[97] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[98] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1): 81–106, 1986.

[99] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016.

[100] Mengwei Ren, Liang Niu, and Yi Fang. 3d-a-nets: 3d deep dense descriptor for volumetric shapes with adversarial networks. *arXiv preprint arXiv:1711.10108*, 2017.

[101] Lawrence Roberts. Picture coding using pseudo-random noise. *IRE Transactions on Information Theory*, 8(2):145–154, 1962.

[102] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 (6):386, 1958.

[103] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5 (3):1, 1988.

[104] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent point feature histograms for 3d point clouds. In *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pages 119–128. IOS Press, 2008.

[105] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[106] Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014.

[107] Manolis Savva, Angel X Chang, and Pat Hanrahan. Semantically-enriched 3d models for common-sense knowledge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 24–31, 2015.

[108] Nima Sedaghat, Mohammadreza Zolfaghari, Ehsan Amiri, and Thomas Brox. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351*, 2016.

[109] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[110] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Exploiting the panorama representation for convolutional neural network classification and retrieval. In *3DOR*, 2017.

[111] Syed Afaq Ali Shah, Mohammed Bennamoun, and Farid Boussaid. Keypoints-based surface representation for 3d modeling and 3d object recognition. *Pattern Recognition*, 64:29–38, 2017.

[112] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015.

[113] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *arXiv preprint arXiv:1812.04244*, 2018.

[114] Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven Dickinson. Retrieving articulated 3-d models using medial surfaces. *Machine vision and applications*, 19(4):261–275, 2008.

[115] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016.

[116] Ivan Sipiran and Benjamin Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27(11):963–976, 2011.

[117] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[118] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in neural information processing systems*, pages 656–664, 2012.

[119] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[120] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recogni-

tion. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[121] Jong-Chyi Su, Matheus Gadelha, Rui Wang, and Subhransu Maji. A deeper look at 3d shape classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[122] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)*, 23(3):399–405, 2004.

[123] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[124] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[125] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318. ACM, 1994.

[126] Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

[127] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):583–598, 1991.

[128] Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 474–481. IEEE, 2003.

[129] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *Proceedings of British Machine Vision Conference (BMVC)*, volume 12, 2017.

[130] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *arXiv preprint arXiv:1903.01864*, 2019.

[131] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.

[132] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[133] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. Human detection using depth information by kinect. In *CVPR 2011 workshops*, pages 15–22. IEEE, 2011.

[134] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European conference on computer vision*, pages 160–176. Springer, 2016.

[135] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[136] Zhige Xie, Kai Xu, Wen Shan, Ligang Liu, Yueshan Xiong, and Hui Huang. Projective feature learning for 3d shapes with multi-view depth images. In *Computer Graphics Forum*, volume 34, pages 1–11. Wiley Online Library, 2015.

[137] Yang Yang, Bin B Zhu, Rui Guo, Linjun Yang, Shipeng Li, and Nenghai Yu. A comprehensive human computation framework: with application to image labeling. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 479–488. ACM, 2008.

[138] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[139] Mohsen Yavartanoo, Eu Young Kim, and Kyoung Mu Lee. Spnet: Deep 3d object classification and retrieval using stereographic projection. *arXiv preprint arXiv:1811.01571*, 2018.

[140] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. *arXiv preprint arXiv:1812.03320*, 2018.

[141] Haoxuan You, Yifan Feng, Rongrong Ji, and Yue Gao. Pvnet: A joint convolutional network of point cloud and multi-view for 3d shape recognition. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 1310–1318. ACM, 2018.

[142] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 186–194, 2018.

[143] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 373–380. IEEE, 2009.

[144] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[145] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 689–696. IEEE, 2009.

[146] Aaron Zinman, Doug Fritz, Greg Elliott, and Roy Shilkrot. Depthjs. `https://github.com/doug/depthjs`, 2010.