Reconstruction and analysis Of 3D models for autonomous vehicles and manufacturing industry

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: ING-INF/05–INFORMATION PROCESSING SYSTEMS

Final Dissertation

Reconstruction And Analysis Of 3D Models For Autonomous Vehicles And Manufacturing Industry

---

by
Pernisco Gaetano:

Supervisors:

Prof. Tommaso Di Noia

Dott. Ettore Stella

Coordinator of Ph.D. Program:
Prof. Mario Carpentieri

Course n°35, 01/11/2019-31/10/2022

Politecnico di Bari

LIBERATORIA PER L'ARCHIVIAZIONE DELLA TESI DI DOTTORATO

Al Magnifico Rettore
del Politecnico di Bari

Il/la sottoscritto GAETANO PERNISCO nato a CASTELLANETA il 09/11/1991

residente a PALAGIANO in via GALVANI 16 e-mail GAETANO.PERNISCO@POLIBA.IT

iscritto al 3° anno di Corso di Dottorato di Ricerca in INGEGNERIA ELETTRICA E DELL'INFORMAZIONE ciclo 35

ed essendo stato ammesso a sostenere l'esame finale con la prevista discussione della tesi dal titolo:

Reconstruction And Analysis Of 3D Models For Autonomous Vehicles And Manufacturing Industry

DICHIARA

1) di essere consapevole che, ai sensi del D.P.R. n. 445 del 28.12.2000, le dichiarazioni mendaci, la falsità negli atti e l'uso di atti falsi sono puniti ai sensi del codice penale e delle Leggi speciali in materia, e che nel caso ricorressero dette ipotesi, decade fin dall'inizio e senza necessità di nessuna formalità dai benefici conseguenti al provvedimento emanato sulla base di tali dichiarazioni;
2) di essere iscritto al Corso di Dottorato di ricerca INGEGNERIA ELETTRICA E DELL'INFORMAZIONE ciclo 35, corso attivato ai sensi del "*Regolamento dei Corsi di Dottorato di ricerca del Politecnico di Bari"*, emanato con D.R. n.286 del 01.07.2013;
3) di essere pienamente a conoscenza delle disposizioni contenute nel predetto Regolamento in merito alla procedura di deposito, pubblicazione e autoarchiviazione della tesi di dottorato nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica;
4) di essere consapevole che attraverso l'autoarchiviazione delle tesi nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica del Politecnico di Bari (IRIS-POLIBA), l'Ateneo archivierà e renderà consultabile in rete (nel rispetto della Policy di Ateneo di cui al D.R. 642 del 13.11.2015) il testo completo della tesi di dottorato, fatta salva la possibilità di sottoscrizione di apposite licenze per le relative condizioni di utilizzo (di cui al sito http://www.creativecommons.it/Licenze), e fatte salve, altresì, le eventuali esigenze di "embargo", legate a strette considerazioni sulla tutelabilità e sfruttamento industriale/commerciale dei contenuti della tesi, da rappresentarsi mediante compilazione e sottoscrizione del modulo in calce (Richiesta di embargo);
5) che la tesi da depositare in IRIS-POLIBA, in formato digitale (PDF/A) sarà del tutto identica a quelle **consegnate**/inviate/da inviarsi ai componenti della commissione per l'esame finale e a qualsiasi altra copia depositata presso gli Uffici del Politecnico di Bari in forma cartacea o digitale, ovvero a quella da discutere in sede di esame finale, a quella da depositare, a cura dell'Ateneo, presso le Biblioteche Nazionali Centrali di Roma e Firenze e presso tutti gli Uffici competenti per legge al momento del deposito stesso, e che di conseguenza va esclusa qualsiasi responsabilità del Politecnico di Bari per quanto riguarda eventuali errori, imprecisioni o omissioni nei contenuti della tesi;
6) che il contenuto e l'organizzazione della tesi è opera originale realizzata dal sottoscritto e non compromette in alcun modo i diritti di terzi, ivi compresi quelli relativi alla sicurezza dei dati personali; che pertanto il Politecnico di Bari ed i suoi funzionari sono in ogni caso esenti da responsabilità di qualsivoglia natura: civile, amministrativa e penale e saranno dal sottoscritto tenuti indenni da qualsiasi richiesta o rivendicazione da parte di terzi;
7) che il contenuto della tesi non infrange in alcun modo il diritto d'Autore né gli obblighi connessi alla salvaguardia di diritti morali od economici di altri autori o di altri aventi diritto, sia per testi, immagini, foto, tabelle, o altre parti di cui la tesi è composta.

Luogo e data Bari, 08/01/2023 Firma _____

Il sottoscritto, con l'autoarchiviazione della propria tesi di dottorato nell'Archivio Istituzionale ad accesso aperto del Politecnico di Bari (POLIBA-IRIS), pur mantenendo su di essa tutti i diritti d'autore, morali ed economici, ai sensi della normativa vigente (Legge 633/1941 e ss.mm.ii.),

CONCEDE

- al Politecnico di Bari il permesso di trasferire l'opera su qualsiasi supporto e di convertirla in qualsiasi formato al fine di una corretta conservazione nel tempo. Il Politecnico di Bari garantisce che non verrà effettuata alcuna modifica al contenuto e alla struttura dell'opera.
- al Politecnico di Bari la possibilità di riprodurre l'opera in più di una copia per fini di sicurezza, back-up e conservazione.

Luogo e data Bari, 08/01/2023 Firma _____

(*) - Department of Electrical and Information Engineering
(**) - Electrical and Information Engineering Ph.D. Program
(***) – SSD ING-INF/05-INFORMATION PROCESSING SYSTEMS

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: ING-INF/05–INFORMATION PROCESSING SYSTEMS

Final Dissertation

Reconstruction And Analysis Of 3D Models For Autonomous Vehicles And Manufacturing Industry

---

by
Gaetano Pernisco:

Referees:

Prof. Vito Renò

Prof. Giovanni Dimauro

Supervisors:

Prof. Tommaso Di Noia

Dott. Ettore Stella

Coordinator of Ph.D Program:

Prof. Mario Carpentieri

Course n°35, 01/11/2019-31/10/2022

To Alessia

Abstract

This thesis falls in the general category of computer vision. In particular, it regards the study of the reconstruction and analysis of 3D models problems with a focus on two different domains: autonomous vehicles and the manufacturing industry.

Computer vision is a research topic deeply studied in the last decades with great interest from both researchers and industries. Contrary to image processing, computer vision aims to extract 3D structures and semantic means from images for a rich and complete understanding. The development of Convolutional Neural Networks (CNNs), allowed computer vision to face up new complex problems reaching impressive results.

But computer vision does not regard only bi-dimensional images but also multi-dimensional data. Indeed, the diffusion of new sensors such as Lidars and 3D scanners requested the design of new algorithms to deal with their data structures. In fact, point clouds - the classic data produced by 3D sensors - have a really different nature compared with images.

Autonomous driving represents a domain in which computer vision finds a wide range of applications. Furthermore, to safely move in the urban environment, driverless cars need an accurate and rich perception of the environment. For this reason, data from different sensors are fused to create a 360° 3D representation of the scene.

On the other hand, the manufacturing industry can leverage computer vision approaches, in synergy with other technologies, to automate and facilitate several processes to make lean the production chain. In particular, quality control and warehouse management processes can leverage robotics, 3D scanning, and mixed

reality to facilitate human work.

In this thesis, several contributions mainly based on computer vision are proposed in both domains. It investigates the power of computer vision leveraging both bi-dimensional and tri-dimensional data peculiarity in different identified tasks typical of both domains. Experimental results shown, analyzed, and discussed in this thesis, support the effectiveness of each proposed method.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Computer vision is a research topic deeply studied in the last decades with great interest from both researchers and industries. With computer vision, we mean the set of processes that aim to create an approximation of the 3D real world from images. Its goal is to emulate the human visual system by understanding the semantic means of digital images and automating tasks.

Computer vision research began in the late 1960s as a building block for giving robots intelligent behavior. Different from the image processing fields, in that case, the aim was the extraction of 3D structures from images to a complete scene understanding. Since then, computer vision has always been a research field very active finding applications in a wide range of fields. Up to the present day, when the rapid deep learning development allowed us to face up really complex problems reaching impressive results. In particular, the use of Convolutional Neural Networks allowed us to automatically learn from images addressing many tasks quite unattainable before, such as classification, semantic segmentation, object detection, object recognition, object tracking, etc.

Computer vision image data can take many forms, such as multiple camera views, multi-dimensional data, video sequences, etc. The diffusion of new sensor types, such as time of flight sensors, presented researchers with new challenges.

Indeed, to exploit the accuracy of the 3D data that these sensors are able to provide, new approaches are necessary. As a matter of fact, point cloud data, are really different from images given their unorganized structure. Their nature, for instance, does not permit to use of classical discrete convolutional operations - typically used in CNNs - directly on raw point cloud forcing to a pre-processing step (with a loss of information) or to develop new deep learning architectures. Deep learning on 3D data is a very innovative and largely studied research topic in the last few years.

This work falls into the general computer vision field with a focus on the reconstruction and analysis of 3D models. In particular, new efficient and reliable approaches are proposed to face up several tasks into two main fields with the common aim to exploit bi-dimensional and tri-dimensional data to model and understand the acquired scene. In particular, two main research lines are investigated.

First, computer vision in autonomous vehicles is analyzed. Autonomous driving is one of the fields that more take advantage of computer vision since driverless cars need to accurately perceive and reconstruct the environment to perform their tasks and safely navigate the environment. Usually, they leverage both cameras and time of flight sensors fusing their data to create a rich and complete representation of the scene. Here, two main challenging objectives have been set. First, an effective deep learning method for car tracking exploiting 3D point clouds will be proposed and tested. Then, a new deep learning efficient methodology to infer depth on single-view images will be presented.

Secondly, the manufacturing industry field is explored and new computer vision and AR/VR approaches to facilitate and speed up their processes. With the Industry 4.0 revolution, emerging technologies are adopted in industries to improve their processes. The synergy of computer vision, robotics, and AR/VR can deeply automate some processes in the production chain. In this work, three different manufacturing processes are analyzed proposing as many contributions to automate and facilitate them. Two of these contributions aim to improve the control quality process in the aerospace industry, while the third exploits computer vision

to automate the movement of goods leveraging mobile robots.

In summary, the aim of this thesis is the design and development of new methods to reconstruct, analyze, and understand scenes exploiting both 2D and 3D data. This drive the research to propose new efficient and reliable algorithms to face up to the identified tasks as well as the improvement of those already present in the literature. Effective algorithms are developed and tested to tackle very challenging problems in the two analyzed fields.

1.2 Overview

This thesis is organized as follows:

- **Chapter 1** presents motivation and publications related to this thesis.
- **Chapter 2** introduces the visual feature concept analyzing standard methods for local features detection and description in 2D domain.
- **Chapter 3** introduces Neural Networks with a focus on Convolutional Neural Networks (CNNs).
- **Chapter 4** introduces Point Clouds analyzing their acquisition methods, standard methods for local features detection and description in 3D domain, and deep learning on these data structures.
- **Chapter 5** describes how autonomous vehicles can take advantage of 2D and 3D computer vision and deep learning with a focus on 3D car tracking and monocular depth estimation methods.
- **Chapter 6** describes different methods based on computer vision and AR/VR to improve processes in the manufacturing industries.
- **Chapter 7** provides overall conclusions of the work presented in this thesis and future work.

1.3 List of Publications

This thesis is based on the following publications produced during the Ph.D. program:

- Towards Improving Car Point-Cloud Tracking Via Detection Updates
  Deldjoo, Y., Di Noia, T., Di Sciascio, E., Pernisco, G., Renó, V., Stella, E.
  3rd International Conference on Image Processing and Machine Vision
  2021, May

- Towards real-time monocular depth estimation for mobile systems
  Deldjoo, Y., Di Noia, T., Di Sciascio, E., Pernisco, G., Renò, V., Stella, E.
  Multimodal Sensing and Artificial Intelligence: Technologies and Applications II
  2021, June

- Virtual and Augmented Reality for Quality Control of Aircraft Interiors
  Mosca, N., Pernisco, G., Summa, M. D., Renò, V., Nitti, M., Stella, E.
  International Conference on Image Analysis and Processing
  2022

- An accurate hardware calibration and 3D point cloud stitching towards automatic quality control in aerospace manufacturing
  Dibari, P., Nitti, M., Patruno, C., Pernisco, G., di Summa, M., Mosca, N., Renò, V.
  IEEE 9th International Workshop on Metrology for AeroSpace
  2022, June

- Extended reality and artificial intelligence: Synergic approaches in real world applications
  Di Summa, M., Renò, V., Dibari, P., Pernisco, G., Sacco, M., Stella, E.
  Roadmapping Extended Reality: Fundamentals and Applications
  2022

- Optical encoder neural network: a CNN-based optical encoder for robot localization
  Patruno, C., Renò, V., Nitti, M., Pernisco, G., Mosca, N.
  Optical Engineering
  2022

Chapter 2

Features in Computer Vision

Computer vision applications are really varied and obviously strictly dependent on the specific goal. Generally, computer vision tasks include: object detection, object classification, object segmentation etc. Notwithstanding these great variety, it is possible to recognize a typical sequence of steps to analyze images, referred to as computer vision pipeline [1].

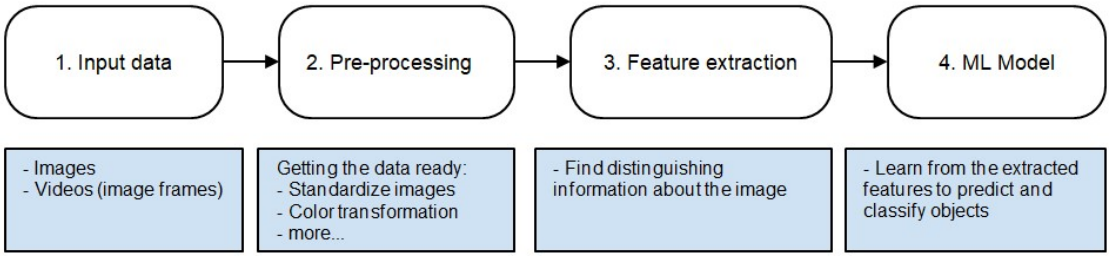


Figure 2.1: The typical computer vision pipeline. It takes input images, pre-process it, extracts features and finally send the processed information to the Machine Learning Model.

As represented in Figure 2.1 the pipeline takes in input one or more images, pre-processes them, extracts relevant features and finally uses the latter to train a machine learning model. This chapter will focus on the features mean and on methods to automatically extract them from images.

In Computer Vision, a feature is a piece of more high level information extracted from image raw data regarding his content which is important to solve a task related to a specific application. It can be different structures such as points,

objects, lines or curves. Furthermore, features can be also the result of neighborhood operations. A good feature permits to distinguish object one another. For this reason, it is desirable that a feature satisfies at least several of the following characteristics:

- **Informativeness:** feature significantly differ from his spatial neighborhood
- **Invariance:** to certain transformations as scale and rotation
- **Robustness:** it has to be identifiable even in presence of noise, blur, discretization and compression
- **Efficiency:** for some application the features detection has to be executed in real-time
- **Distinctiveness:** individual features can be matched in a large database of objects
- **Relevance:** with respect to the specific application. For different object classes, features should provide small variance on the same class objects and large variance when compared with object from different classes

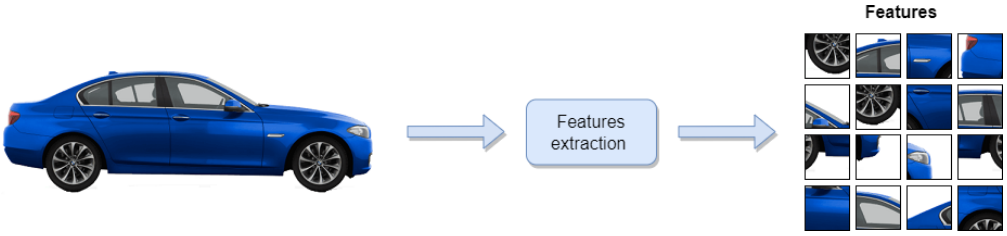


Figure 2.2: Example of features extraction. The algorithm finds patterns within the image to create the feature vector.

Figure 2.2 shows an example of possible features extracted from a car image. It is possible to note that the features detection algorithm chooses elements such as wheels, headlights, turn signal light and antenna as features. Indeed, these elements are more peculiar to distinguish cars.

2.1 Local Features Detection

Several approaches to account local features detection were proposed in literature. The most commonly used ones are presented in the following subsections.

2.1.1 Harris Corner Detector

In 1988, Chris Harris and Mike Stephens [2] presented a corner detection operator widely used in computer vision algorithms. It leverage the intuition behind the Moravec corner detector [3] which considers local image patches and the average change of intensity resulting from its small shifts in both directions. For this measure the Sum of Squared Differences (SSD) between the two considered patches is used. Let be $I(x, y)$ the image pixel values and $w(x, y)$ the window centered in position (x, y), the SSD value $E_{(u,v)}$ resulting from shifting the considered patch by (u, v) is defined as:

$$E_{(u,v)} = \sum_{(x,y)} w(x, y)[I(x + u, y + v) - I(x, y)]^2 \tag{2.1}$$

In this scenario, the goal is to maximize the term $[I(x + u, y + v) - I(x, y)]^2$. Using Taylor series to approximate that term, it is possible to rewrite the Equation 2.1 in the form:

$$E_{(u,v)} \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.2}$$

where M is the so-called auto-correlation matrix:

$$M = \sum_{(x,y)} w(x, y) \begin{bmatrix} I_x^2(x, y) & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y^2(x, y) \end{bmatrix} \tag{2.3}$$

where $I_x(x, y)$ and $I_y(x, y)$ are the partial derivatives of the image respectively on x and y directions. The eigenvalues λ_1 and λ_2 of M are used to discriminate corner regions from non-corner ones. More precisely:

- Both eigenvalues are small, this indicates little changes in $E_{(u,v)}$ shifting the window. Therefore, it denotes an approximately constant intensity in the local neighborhood. This is a flat zone.

- One eigenvalue is high and the other one is small, this indicates little changes in $E_{(u,v)}$ shifting the window only in one direction (i.e. along the edge). This is an edge.
- Both eigenvalues are high, this indicates high changes in $E_{(u,v)}$ shifting the window in both directions. This is a corner.

Given these assumptions, a score R is defined as:

$$R = det(M) - k(trace(M))^2 \tag{2.4}$$

where:

$$det(M) = \lambda_1 \lambda_2$$

$$trace(M) = \lambda_1 + \lambda_2$$

k represent a sensitive parameter. By thresholding the R score it is possible to detect corner windows. This allows to avoid the eigenvalues calculus, improving the algorithm performances.

2.1.2 SIFT Detector

The Scale Invariant Features Transform (SIFT) algorithm was first presented in 1999 [4] and then improved in 2004 [5] by D. Lowe. In contrast to Harris corner detector, SIFT algorithm was meant to be scale invariant. The SIFT detector is based on an approximation of Laplacian function and it is manly composed by two steps: scale-space extrema detection and keypoints localization.

Scale-space Extrema Detection

It is obvious that it is not possible to use the same searching window to detect features with different scales in an image. For this reason, scale-space filtering is used. The scale-space of an image is a well-known concept in Computer Vision and it is a function $L(x, y, \sigma)$ that is produced using a Gaussian convolutional kernel at different scales of the image. Scale-spaces is divided in octaves, which number depends on the image resolution, each with half the resolution compared to the

previous one. In each octave, the image is progressively convolved with a Gaussian kernel with different σ values in order to create a list of blurred images.

Subtracting neighboring images in the same octave, yields the so-called Difference-of-Gaussians (DoG) pyramid, represented in Figure 2.3. DoG images permit to find interesting points in images. It is obtained as the difference of Gaussian blurring of an image with two different σ, let it be σ and $k\sigma$.

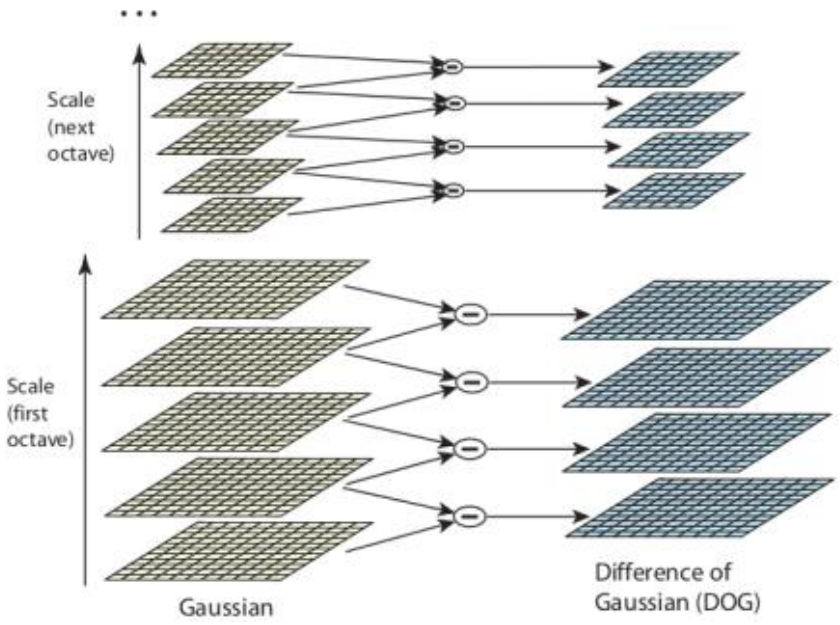


Figure 2.3: The set of scale space images on the left, the Difference-of-Gaussians pyramid on the right.

Once created the DoG pyramid, DoG images are used to found local extrema over space and scale. As shown in Figure 2.4, each pixel in each image is compared with its eight neighbours in the same image and with its nine neighbours in previous and next scale. If the pixel is a local extrema, it is a potential keypoint and it means that this is better represented in the identified scale.

Keypoint localization

Once potential keypoints are found, they have to be filtered and their location improved to subpixel accuracy to get better results. A Taylor series expansion of

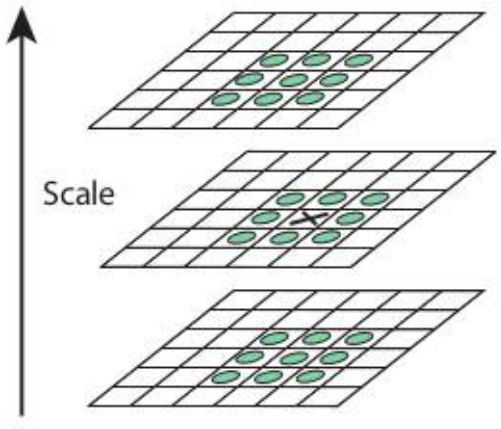


Figure 2.4: Detection of local extrema

scale-space allows to interpolate the local extrema location and all extrema are filtered by their intensity value. Indeed, since DoG has an high response also for edges, an approach similar to the Harris corner detector is used to reject them.

2.1.3 SURF Detector

In 2006, Bay et al. [6] introduced a new features detector, SURF (Speeded Up Robust Features), to speed up the SIFT detector enabling real-time applications such as tracking and object recognition.

As shown in the previous section, SIFT approximates the Laplacian of Gaussian function with Differences of Gaussian. The core idea behind SURF, instead, is to approximate it with Box Filters. An advantage of this is that convolution with box filters can be easily calculated exploiting the integral images.

An integral image is an image $I_{\sum}$ in which each pixel $p = (x, y)^T$ represents the sum of all pixels in the original image I within the rectangular region formed by the origin and the pixel p. More precisely:

$$I_{\sum}(p) = \sum_{u=0}^{u \leq x} \sum_{v=0}^{v \leq y} I(u, v) \tag{2.5}$$

It allows the fast computation of box convolutional filters and it permits to use only three additions to compute the sum of the intensities over any upright rectangular

area independently from its size.

Indeed, SURF rely on the determinant of the Hessian matrix for both location and scale. In particular, the determinant of the Hessian matrix is used to measure the local change around pixels. Points where the determinant is maximum are chosen as interesting points. SURF uses the determinant of the Hessian matrix also for the scale selection.

Given the pixel $p = (x, y)$, the Hessian matrix $H(p, \sigma)$ in p at scale σ is defined as:

$$H(p, \sigma) = \begin{bmatrix} L_{xx}(p, \sigma) & L_{xy}(p, \sigma) \\ L_{xy}(p, \sigma) & L_{yy}(p, \sigma) \end{bmatrix} \tag{2.6}$$

where $L_{xx}(p, \sigma)$ is the convolution of the Gaussian second order derivative in the horizontal direction, with the image I in p, similarly for $L_{xy}(p, \sigma)$ and $L_{yy}(p, \sigma)$. SURF approximates the second order Gaussian derivatives as in Figure 2.5. This permits to evaluate the convolution with a very low computational effort using integral images. Denoting the approximated second order Gaussian derivatives as

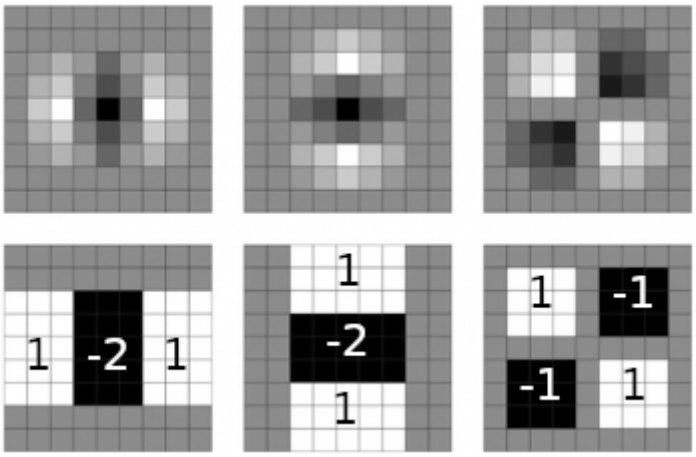


Figure 2.5: Up row: the second order Gaussian derivatives L_{xx}, L_{yy} and L_{xy}. Down row: their respective approximations D_{xx}, D_{yy} and D_{xy}. Here the mid-grey denote the 0 value.

D_{xx}, D_{yy} and D_{xy}, the determinant of the approximated Hessian matrix can be expressed as:

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.7}$$

2.1.4 FAST Detector

As just seen, SURF with its approximations, represent a good features detector with reasonable computational requirements. In computer vision, there are some real-time applications that need to use limited computational resources. One of best example can be SLAM (Simultaneous Localization and Mapping), that often need to be executed on mobile robots hardware. For applications like that, SURF is not fast enough. FAST [7], Features from Accelerates Segment Test, is a corner detector born to face up tasks with high performance requirements.

FAST algorithm is based on comparisons of each pixel with its local neighborhood. In particular, FAST detector uses the pixels on a Bresenham circle [8] of radius $r = 3$ centered on the pixel, considering therefore only 16 pixels. The intensity of each pixel $I(p)$ of the image, is compared with his 16 pixels on his surrounding Bresenham circumference (as shown in Figure 2.6).

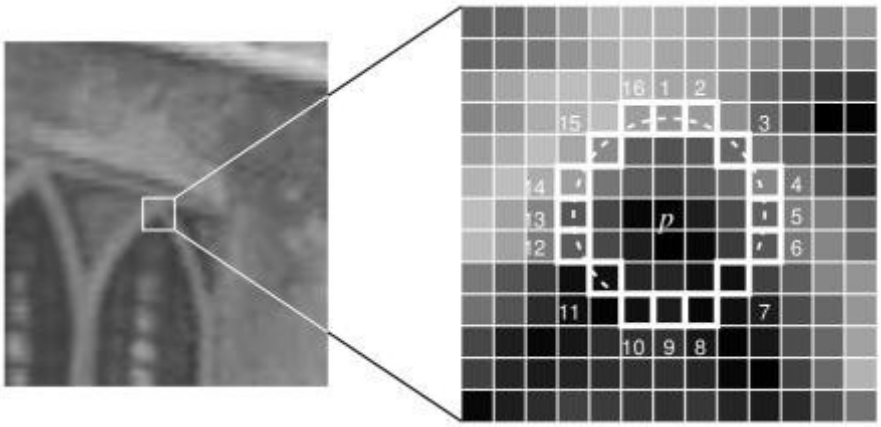


Figure 2.6: FAST corner detector segment-test. The numbered pixels are used in the FAST algorithm. p is the candidate keypoint.

The investigated pixel is a potential keypoint if the intensities of at least 12 contiguous pixels are above $I(p) + t$ or below $I(p) - t$, there t is a threshold. To optimize this test it is possible to first compare $I(p)$ with the opposite vertical and horizontal pixels in the circumference to check if at least 3 of these pixels satisfy the constraint. A non-maximum suppression algorithm, at the end, filter out multiple keypoints on adjacent locations.

2.2 Local Features Description

After the features detection, it is fundamental to describe each feature based on its neighborhood to give a meaning to each of them. This allow, indeed, to match features across images to perform tasks like object recognition, image retrieval, image stitching, etc.

A descriptor is a vector that describes in some way the local appearance around each feature. Ideally it is distinctive and invariant to changes in illumination, scale and rotation. Descriptors can be classified based on:

- **Neighborhood shape:** the shape of the area around the feature considered to calculate the descriptor vector. It can be rectangular or circular;
- **Sampling type:** the sampling technique used to choose pixels in the considered neighborhood in the descriptor vector computation. There are two possible approaches:
  - Dense: all neighboring pixels are considered;
  - Sparse: only some neighboring pixels are considered;
- **Used values:** the values that can be assumed by each component of the descriptor vector. It can be binary or scalar.

In the following subsections, a review of some of the most famous feature descriptor algorithms is proposed.

2.2.1 BRIEF Descriptor

BRIEF (Binary Robust Independent Elementary Features), was introduced in 2010 by Calonder at al. [9] and it is considered the first and simplest binary features descriptor.

BRIEF descriptor is based on comparisons at pixel level, for this reason it is noise sensitive. To reduce this sensitivity, the algorithm uses a Gaussian convolutional filter to smooth the image.

The algorithm compose a binary vector by comparing a relatively small number of pixel intensities. More specifically, authors defines a binary test τ on a patch h of size SxS as:

$$\tau(h; x, y) := \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

where $p(x)$ represents the pixel intensity at $x = (u, v)^T$ location. Choosing a set of n_d location pairs uniquely defines a set of binary tests. The result is an n_d-dimensional bitstring:

$$f_{n_d}(h) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(h; x_i, y_i) \tag{2.9}$$

This approach leaves many options for selecting the n_d location pairs in the SxS patch. Authors propose five possible sampling patterns shown in Figure 2.7.

Let assume the SxS patch h centered in the detected feature:

- **G I:** $(X, Y) \sim$ i.i.d. Uniform$(-\frac{S}{2}, \frac{S}{2})$
- **G II:** $(X, Y) \sim$ i.i.d. Gaussian$(0, \frac{1}{25}S^2)$
- **G III:** $X \sim$ i.i.d. Gaussian$(0, \frac{1}{25}S^2)$, $Y \sim$ i.i.d. Gaussian$(x_i, \frac{1}{100}S^2)$
- **G IV:** (x_i, y_i) are randomly sampled from discrete locations of a coarse polar grid
- **G V:** $\forall i : x_i = (0, 0)^T$ and y_i takes all possible values on a coarse polar grid

BRIEF has the advantage to be a simple and fast to compute since it relies on a small number of intensity difference tests. Indeed, given the binary nature of the resulting descriptor, it is memory efficient and fast to match.

2.2.2 SIFT Descriptor

The SIFT algorithm, after the feature detection, provides also a feature descriptor part. SIFT calculate the descriptor in two steps:

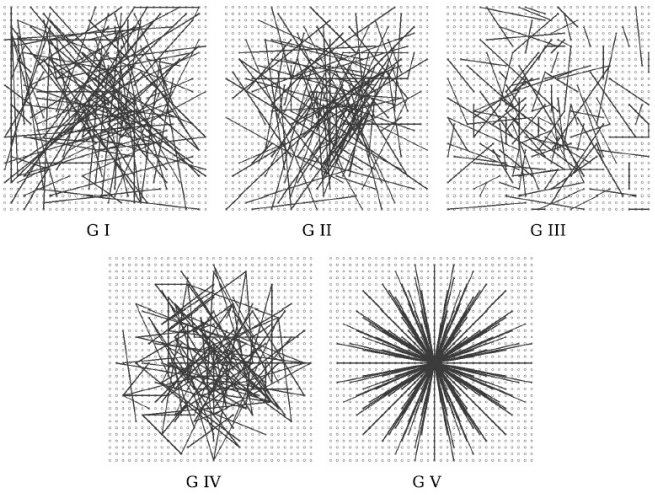


Figure 2.7: Different pattern to choose the test locations.

Orientation assignment

As seen before, the output of the SIFT features detection is a set of local features with a known scale. The goal of this step is to assign an orientation to each detected feature to achieve the rotation invariance.

For this purpose, a neighborhood centered in each detected feature location is considered depending on the scale. In that region, the gradient magnitude and direction is calculated. A 36 bins orientation histogram covering 360° is created.

The highest peak in the histogram describes the orientation of a keypoint and, for any other peak above 80% of it, an additional keypoint with the corresponding orientation is generated.

Generation of keypoint descriptors

The next step involves the computation of a compact and distinctive descriptor vector to each keypoint. At this point, the algorithm knows location, scale and orientation of each keypoint. The aim of this step is to compute an highly distinctive vector invariant (as much as possible) to changes in illumination and viewpoints for each of them.

To this aim, a 16x16 pixel window centered in each keypoint is taken and

it is dividend into 16 4x4 sub-blocks. In order to achieve rotational invariance, it is rotated and aligned with the assigned keypoint orientation. For each 4x4 sub-block, an 8 bin orientation histogram is created. This yields a local image descriptor with 128 elements for each detected feature. Finally, the descriptors are normalized to reduce the effects of illumination changes.

SIFT is one of the most powerful algorithm for features extraction, but it is computationally onerous. SIFT descriptors are scalar, for this reason they generally have higher memory footprint compared with binary ones. To compare SIFT descriptors it is possible to leverage the Euclidean distance.

2.2.3 SURF Descriptor

Exactly as SIFT, also SURF algorithm offers a feature descriptor and also in this case the algorithm cover the same two steps.

Orientation assignment

In order to reach the rotation invariance, SURF descriptors are aligned to the dominant orientation of the detected keypoints. The dominant orientation is calculated using Haar wavelet responses in both x and y direction in a circular region around the keypoint location. The wavelet responses are weighted by a Gaussian centered on the keypoint and represented as 2D points. Then, all points in a sliding orientation window of 60° are summarized in a vector and the orientation that reach the maximum value is chosen as dominant one.

Generation of keypoint descriptors

The orientation calculated in the previous step is used to construct an oriented square region around each keypoint which is then splitted into 4x4 sub-regions. For each sub-region, the Haar wavelet responses at 5x5 regularly spaced sample points are calculated in both x and y local directions, called respectively dx and dy. To increase the robustness both dx and dy are weighted using a Gaussian centered in the keypoint. Then, the sums over dx and dy and the sums over their absolute

Feature Detector	Type	Rotation Inv.	Scale Inv.	Time Eff.
Harris	corner	+	-	++
SIFT	blob (corner)	+	+	++
SURF	blob (corner)	+	+	+++
FAST	corner	+	-	+++

Table 2.1: Qualitative comparison of presented feature detectors

values are calculated to create four characteristics values in each sub-region. The concatenation of these values results in a descriptor vector of lenght 64 which is finally normalized to increase the robustness to illumination changes.

2.3 Comparison of Feature Detectors and Descriptors

In this section a qualitative comparison of analyzed feature detector and descriptor is presented [10].

The performances of feature detectors and descriptors are strongly depending on the specific application, data and requirements. Table 2.1 and Table 2.2 show respectively a qualitative comparison of the presented feature detectors and descriptors.

Feature Descriptor	Size	Descriptor Type	Time Efficiency
BRIEF	32	binary	+++
SIFT	128	scalar	+
SURF	64	scalar	++

Table 2.2: Qualitative comparison of presented feature descriptors

To measure and compare the various approaches the most used metrics are:

- **Repeatability:** measures to what extent the detected regions overlap the same scene region by matching detected features in two images representing

the same scene

- **Matching score:** measures the capability to correctly match features. It also involves the similarity of the respective feature descriptors.

Chapter 3

Deep Learning

Deep Learning (DL) is a subset of Machine Learning (ML) which leverage several representation levels to represent information, where higher level information is defined on the lower one.

The history of deep learning begin in 1943 when Warren McCulloch and Walter Pitts created the first computer model based on neural networks inspired by the human brain. They used a combination of mathematics and algorithms to mimic the thought process, they called it threshold logic. Since then, deep learning has constantly evolved, over the years with two significant breaks in its development. In 1960, Henry J. Kelley laid the foundation of a continuous Back Propagation Model. Stuart Dreyfus, in 1962, simplify Kelley's idea proposing a new version based only on the chain rule.

This chapter will explain the fundamental concepts behind Deep Learning with a particular focus on Convolutional Neural Networks (CNNs).

3.1 Single Neuron Model

Exactly as for the human brain, the basic unit of a neural network is the neuron, also called node or unit. It takes a series of inputs $x_1, x_2, ..., x_n$ from other neurons or from an external source and computes the output. To each input, the single neuron has an associated weight w which represents the importance of that input

in relation to others. The output is given by the function f applied to the weighted sum of inputs. As shown in figure 3.1, the output y is computed as:

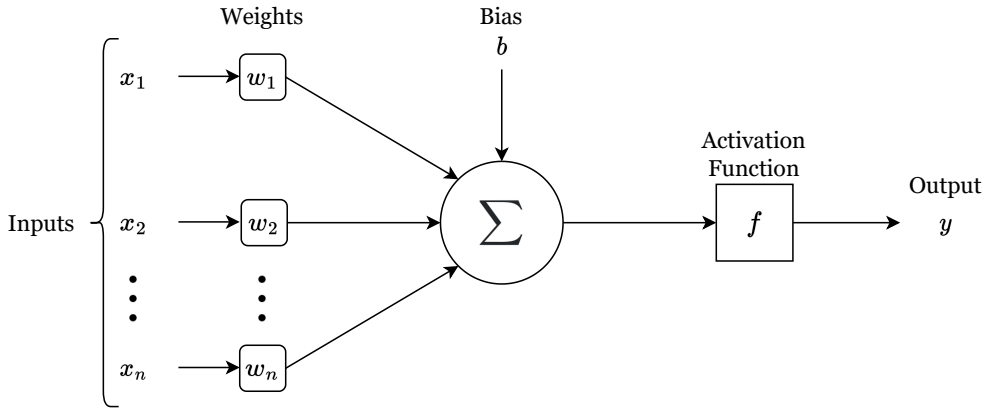


Figure 3.1: Single neuron's structure.

$$y = f(\sum_{i=0}^{n} w_i x_i + b) \tag{3.1}$$

The represented neuron, takes the inputs $x_1, x_2, ..., x_n$ and has weights $w_1, w_2, ..., w_n$ associated with each input. Furthermore, there is another input with value 1 and the associated weight b. This is called bias and its role is to shift the activation function f, to better fit the data.

The activation function f is non-linear and its aim is to introduce non-linearity into the neuron. This is crucial to fit real data because real-world data are always non-linear and the neuron has to learn these non-linear representations.

Different activation functions can be used in neural networks, as shown in Figure 6.3. The most common are:

- **Sigmoid**

  The sigmoid function is defined as:

  $$\phi(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

  It is especially used for models that predict a probability since its codomain exists between 0 to 1. This function is differentiable and monotonic, unlike its derivative.

- **Hyperbolic Tangent**
  The Hyperbolic Tangent ($tanh$) function is defined as:

$$\tanh(x) = \frac{e^{(2x)} - 1}{e^{(2x)} + 1} \tag{3.3}$$

  This function is similar to the sigmoid, but it exists between -1 and 1. It is particularly used in classification tasks. With the $tanh$ function the negative inputs are matted to strongly negative and the zero ones are mapped near zero.

  As sigmoid, $tanh$ function is differentiable and monotonic while its derivative is not monotonic.

- **Rectified Linear Unit**
  The Rectified Linear Unit ($ReLU$) is one of the most used activation functions in Deep Learning.

$$ReLU(x) = max(0, x) \tag{3.4}$$

  $ReLU$ function exists in the range $[0, +\infty)$. It maps all negative values to 0 decreasing the model ability to fit properly from the data. $ReLU$ and its derivative both are monotonic.

- **Leaky Rectified Linear Unit**
  $LeakyReLU$ is similar to $ReLU$ and is defined as:

$$LeakyReLU(x) = max(ax, x) \tag{3.5}$$

  where the a value is 0.01. Contrary to $ReLU$, it is defined in the range $(-\infty, +\infty)$. As $ReLU$, $LeakyReLU$ and its derivative both are monotonic.

3.2 Neural Networks

Neural Networks are mathematical models that use learning algorithms inspired by the biological neural networks of the human brain. Nowadays, they can to reach

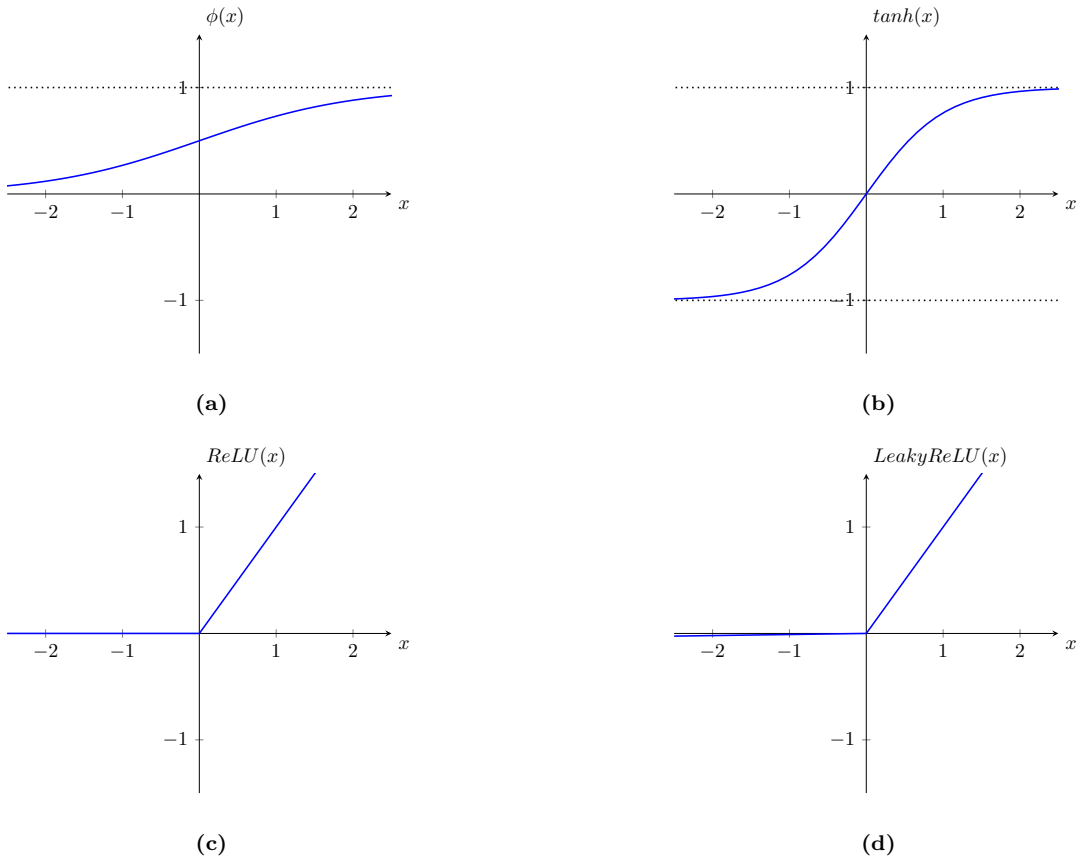


Figure 3.2: Activation functions. (a) Sigmoid; (b) Hyperbolic Tangent; (c) Rectified Linear Unit; (d) Leaky Rectified Linear Unit

impressive results in a wide range of tasks covering a highly variegated range of fields as computer vision, natural language processing and speech recognition.

Artificial Neural Networks (ANN) are composed of a large number of interconnected neurons working together to solve a single task. As human brain, ANNs learn by example to carry out a specific task.

As shown in Figure 3.3 neurons in ANNs are organized in layers with edges between them. Generally there are three type of layers:

- **Input layer:** provides raw information from the external world to the network. It does not have an associated activation function.
- **Hidden layer:** is the layer in the middle, for this reason does not have

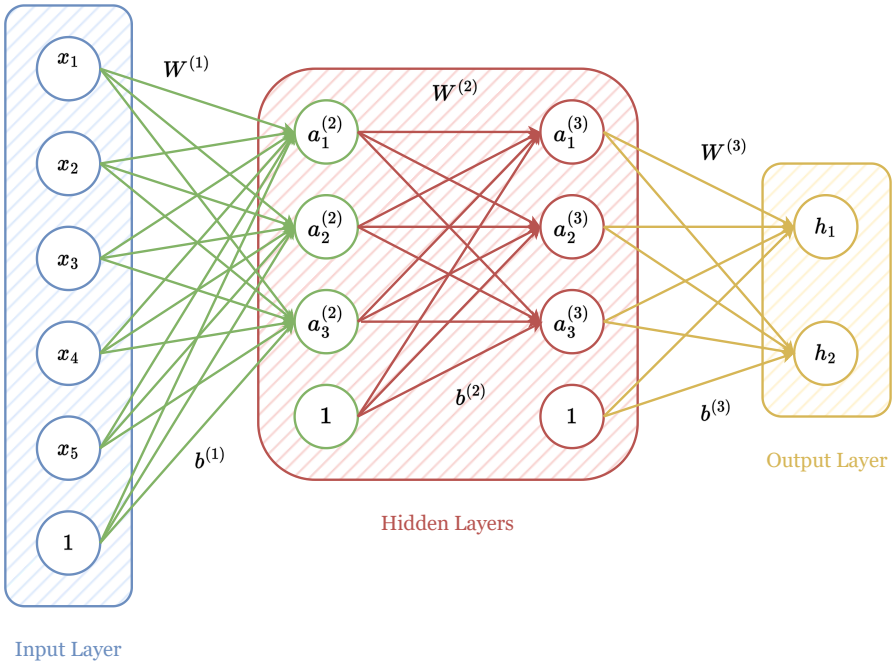


Figure 3.3: Artificial Neural Network structure.

connections with the external world (because of this it is called "Hidden"). It applies weights to the inputs and direct them through an activation function to the output.

- **Output layer:** is responsible for producing and direct out from the network the final result. It takes the result of layer before it, performs the calculation via its neurons and return the final result to the real world.

3.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a particular type of Artificial Neural Network commonly used for computer vision tasks. In fact, CNNs are particularly useful to detect pattern in images in order to recognize objects.

In 1980 Fukushima et al. [11] laid the foundation of the research around CNNs. Then, in 1989 LeCun et al. [12] applied backpropagation to train a CNN to recognize pattern on series of handwritten zip codes. After that, in 1998 he proposed

LeNet-5 [13], a CNN architecture to document recognition. Later, with the introduction of datasets as MNIST and CIFAR10, several new CNNs architectures were introduced.

As shown in Chapter 2, before CNNs development, manual and time-consuming algorithms were used to extract features from images. CNNs provide an innovative approach to computer vision problems which, leveraging linear algebra, can automatically learn features from examples reaching better results.

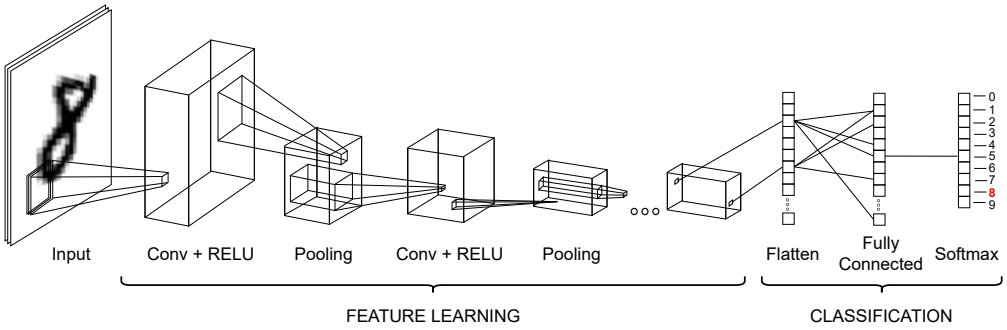


Figure 3.4: Convolutional Neural Network structure. It is an example of a possible CNN structure to classify handwritten digit.

As it is possible to notice in Figure 3.4, commonly a CNN is mainly composed from three different layers:

- **Convolutional layer:** is the core layer in a CNN. It has a series of kernels, with a predefined size, and apply convolution between kernels and input data in order to extract visual features. By stacking up a series of convolutional layers is possible to impose a hierarchical structure in the CNN so that each layer can see pixels within the receptive fields of the previous one. In this way, first layers can detect simpler features, e.g. brightness or edges, while later filters can detect more complex features to uniquely recognize the object.

- **Activation function:** the commonly used activation function after convolutional layer is the ReLU. As shown before, it maps negative values to zero and maintains only positive values from the input data.

- **Pooling layer:** performs a non-linear downsampling, reducing the parame-

ters that the network has to learn. Similar to the convolutional layer, it uses a shifting window on the input to aggregate values inside that. Based on the aggregation function, it is possible to choose different types of pooling layers as max pooling and average pooling.

Chapter 4

Point Clouds

Nowadays, a great number of tools able to improve not only industries and production processes but also everyday life, need a three-dimensional inspection of the environment. With the development of new technologies, several new sensors have been made available.

3D data were first used in robotics where a detailed 3D representation of the surrounding environment is necessary to perform tasks such as self-localization and mapping (SLAM) [14, 15, 16], path planning [17, 18, 19] and obstacle avoidance [20, 21, 22].

After that, these technologies attracted the attention of researchers and companies working on autonomous driving [23, 24] and quality control fields [25, 26]. As for robotics, autonomous cars need to carefully reconstruct the environment to plan the path and to reach the high level of safety needed while driving.

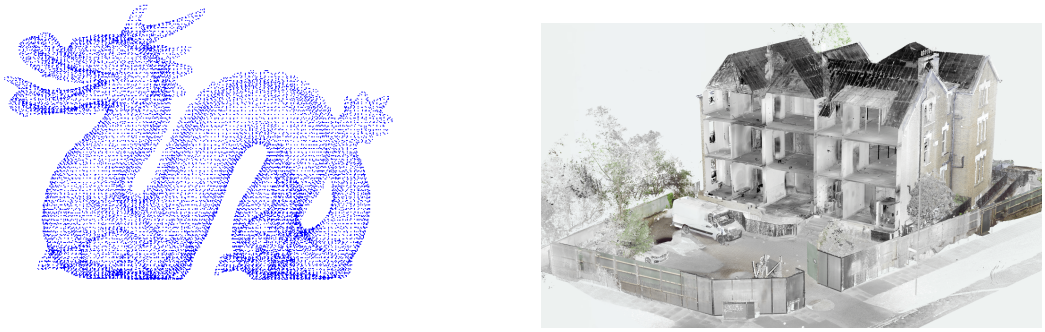
Furthermore, industries take great advantage of 3D data. These technologies can drastically improve manufacturing processes, like quality control processes, providing higher accuracy and performances compared with the human counterpart. Finally, in the last years, other fields have benefited from three-dimensional data, just thinking of all the possible applications in agriculture [27, 28], medicine [29, 30], archaeology [31, 32] and geology [33, 34].

3D data can have really different nature, but the most diffused data commonly returned by the sensors are the so-called point clouds. A point cloud is a set of

points in 3D space, all expressed in the same Cartesian reference system, where each point represents the three-dimensional coordinates of a location on a real-world object's surface.
Using a mathematical definition:

$$P = \{(x_i, y_i, z_i) | i \in N\} \tag{4.1}$$

where P is the point cloud, x, y and z are the coordinates and N is the number of points contained in P. That definition represents the most basic form of point cloud. There are other information that can be stored for each points based on the sensor such as: color, normal to the surface, information about meshes (vertices and edges) and intensity of return. In Figure 4.1 are shown two example of point clouds.



(a) **(b)**

Figure 4.1: Two examples of point clouds. (a) Represents a single object. (b) Point cloud acquired outdoor representing a building.

Based on the organization of data, point clouds can be divided into two categories:

- **Organized point clouds:** data are organized, like an image, in a matrix $MxNxC$ where M is the number of rows, N is the number of columns and C is the number of channels. This format is typically used in stereo cameras and time-of-flight cameras.

- **Unorganized point clouds:** data are not organized in a matrix but are arranged in an unordered list of points. More precisely, data are given as a list of M points with C channels each. This format is typical of LIDAR sensors.

Nowadays, the market offers a great variety of very different sensors for 3D scanning. Basically, all these sensors are based on one of the following principles.

Stereo vision

Stereo vision is a computer vision technique that mimics the human binocular vision to extract 3D information from images. It is based on the use of two or more cameras framing the same scene from different points of view at the same time. The concept can be extended to the case of a single camera moving in the scene acquiring images in different points of space and time (motion stereo, structure from motion). Figure 4.2 illustrates the basic projection principle behind stereo vision.

By comparing two images acquired with two cameras displaced horizontally from one another, the depth information can be deduced in the form of a disparity map. Disparity values are inversely proportional to the depth.

The comparison of the stereoscopic images can be mainly implemented with two different techniques:

- **Feature Matching:** features extracted from the two images are matched without constraints. The method allows to create a sparse reconstruction of the scene.

- **Rectified Stereo:** stereo images are aligned by calibration so that corresponding pixels in the two images are on the same row, simplifying the matching problem. The method allows to create a dense reconstruction of the scene.

A necessary condition to reconstruct the scene using a stereo camera is the knowledge of the intrinsic parameters of the cameras and their relative pose, both estimable with calibration techniques.

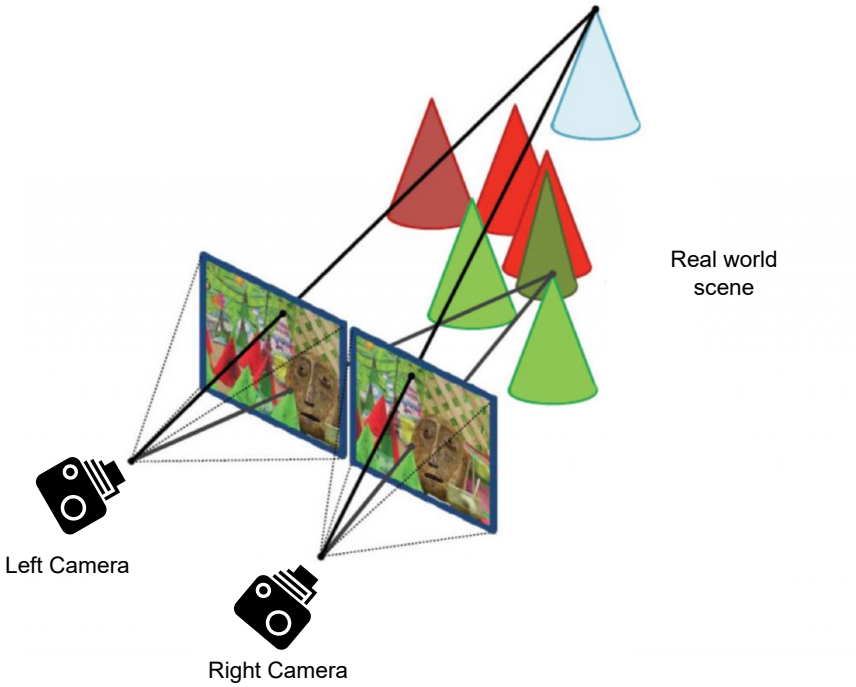


Figure 4.2: Example from the Middlebury Stereo Dataset "cones" where a scene is imaged with a stereo camera.

Structured light

Structured light 3D scanners are based on a quite simple concept: a known pattern projected on a surface.

As shown in Figure 4.3, the system is composed of two main components: a projector and a camera. The projector projects a light with a known pattern on the scanned surface while the camera acquires an image. The vision system is able to extract depth information by analyzing how the scanned surface distorts the projected pattern. Different methods can use different patterns, e.g. stripes, grids, points and others, or a sequence of them. Furthermore, the pattern can be projected using different parts of the light spectrum, visible or invisible light, based on the specific application. Moreover, it is possible to leverage more than a single calibrated camera to reach higher reconstruction accuracy.

Structured light 3D scanners are widely diffused in industries to accomplish tasks such as quality control, reverse engineering, measurements, and videogames.

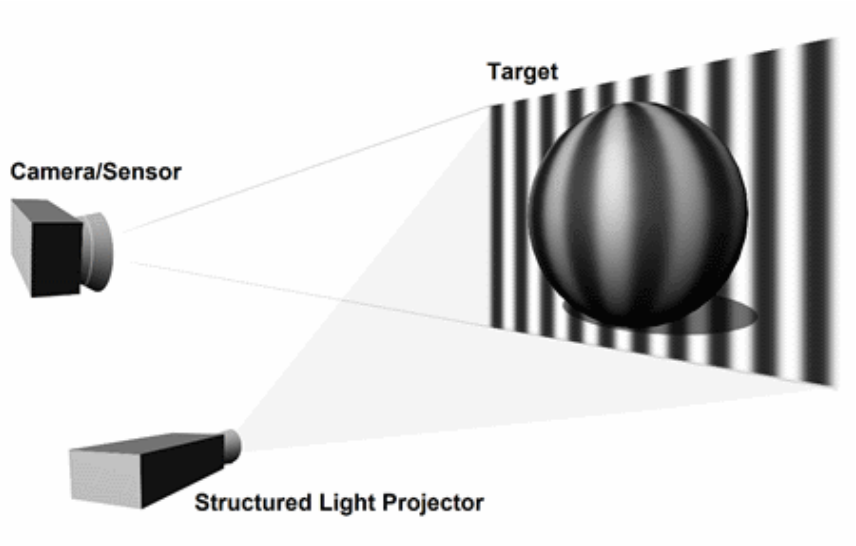


Figure 4.3: A structured light system. The projector project a vertical strips pattern on the scene. The pattern light is distorted by the sphere. The camera acquires an image of the scene and can three-dimensionally reconstruct it leveraging the pattern distortion.

It can offer great precision in 3D reconstruction but its use is limited to indoor scenarios given the nature of this technique. Indeed, the natural light can create problems with the visibility of the projected patterns. The same can happen scanning surfaces with high reflectivity. One of the most diffused structured light sensors is the Microsoft Kinect v1. Born for gaming and consumer electronics, it is used in a wide range of practical applications.

Time-of-Flight

Time-of-Flight (ToF) means the time taken by an object to travel in a medium a particular distance. ToF sensors leverage this concept to measure depth and distances and reconstruct the scene. As shown in Figure 4.4, a ToF camera is composed of two main components: a light source and a ToF sensor. The light source (it can be LED or laser) actively illuminates the scene with a modulated light. The sensor captures the reflected light and measures the time delay ΔT between the light emission and the reception of the backscattered signal. As shown in Equation 4.2, this delay is proportional to twice the distance d between the

camera and the target object:

$$d = \frac{c\Delta T}{2} \tag{4.2}$$

where c is the light velocity. Nowadays, ToF cameras are really diffused, e.g. some smartphones include a ToF sensor in their camera. One of the most diffused ToF camera is the Microsoft Kinect v2, developed for gaming purposes but also used in a wide range of applications.

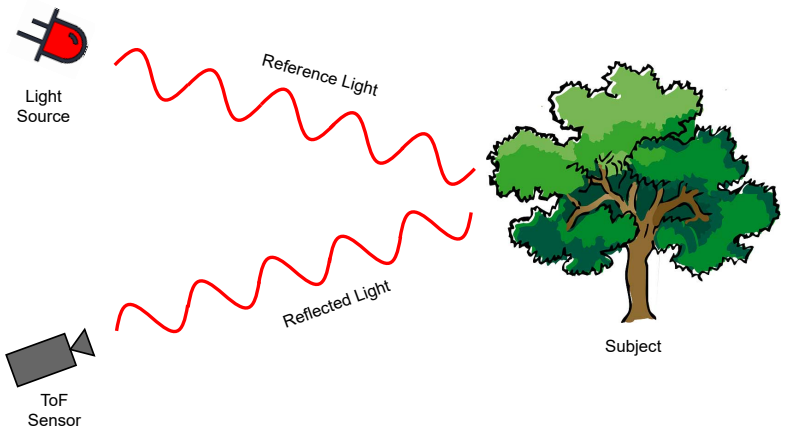


Figure 4.4: A ToF sensor system. The system is composed by a light source and a ToF sensor. The light source illuminates the scene while the ToF sensor acquire the reflected light. The depth information is derived by measuring the time between light emission and its reception.

A particular sensor in this category is the LIDAR which uses laser light. Based on the laser beam steering, it is possible to recognize two different LIDAR systems: mechanical LIDAR and solid-state LIDAR. The farthest deflects the laser beam using rotating mirrors to increase the field of view which usually reach 360° horizontally. The use of rotating mirrors entails a high signal-to-noise ratio (SNR) and a lower frequency in sampling a complete scene. Furthermore, mechanical LIDARs are really expensive. To overcome the mechanical LIDARs issues, solid-state LIDARs have no mechanical parts, and are cheaper but, obviously, have a reduced FOV. The reduced FOV problem can be simply solved by using several calibrated LIDARs to cover 360°. Solid-state LIDARs can be based on different technologies e.g. MEMs LIDAR, Flash LIDAR, OPA LIDAR, FMCW LIDAR. Explaining the working principle of each of these technologies is out of the aim of this thesis.

Another family of sensors based on ToF are Terrestrial Laser Scanners (TLS). These are sensors particularly suitable to scan outdoor environments given their high range. They are often used in geological and archaeological applications. The main issues of these sensors reside in their dimensions, weight, cost and time to execute a scan.

4.1 Local Features on 3D Point Clouds

As shown, also 3D data can be used for a wide range of very different tasks and this means that, as for 2D data, the feature concept is fundamental even here. Obviously, the algorithms described in Chapter 2 are not suitable for point clouds since the different nature of data and their arbitrary topology.

In the last decades, with the spread of 3D sensors, new algorithms for feature detection and description on point clouds were developed with the purpose of addressing tasks such as point cloud registration, object recognition and localization. Here a brief overview of algorithms for both keypoint detection and feature description is reported. The described algorithms are all available in the Point Cloud Library (PCL) which represents one of the more complete and used C++ library for point cloud processing.

4.1.1 Keypoints detection

Harris3D

Harris3D [35] detector is an extension of the 2D Harris corner detector to three-dimensional data. This method is based on the evaluation of the Harris operator in Equation 2.4 which is based on the auto-correlation matrix.

More precisely, for each point v in the point cloud P is considered a neighborhood on which a quadratic surface is fitted. First-term derivatives on x and y directions in the point v are then calculated to compose the auto-correlation matrix M. Finally, the Harris operator is evaluated at each point of P. The output of these steps is a point cloud with a Harris operator value associated with each

point. The authors propose two different ways to choose the keypoints. Firstly, all local maxima points - i.e. all points with the maximum Harris operator value in their neighborhood - are preserved. Then, one of the two following approaches can be used:

- *Select the points with the highest Harris response:* it preserves a constant fraction of points with the highest Harris operator value. This approach keeps the most salient points, thus some portions of the point cloud do not have keypoints.
- *Representatives of Interest Points Clusters:* it permits to have a better keypoints distribution. It consists of two steps: first points are sorted by their Harris operator value, then they are clustered and keypoints are selected.

Normal Aligned Radial Feature

Normal Aligned Radial Feature (NARF) [36] is a keypoint detector designed to work on range images. This is because on this data it is possible to translate 2D computer vision methods in the 3D domain.

It is designed to detect keypoints that are able to take peculiar information about borders and surface structure, easily recognizable from different points of view and that are in positions able to provide a stable normal estimation.

The detection is composed of two steps. Firstly, the algorithm detects object borders based on the distance between each point and its neighborhood. Then, to achieve position stability, for each point in the range image, a score indicating the number of surface changes and the dominant direction is calculated. Afterwards, an interest value is calculated representing changes of directions and surface in the local neighborhood. In the end, final interest points are chosen by smoothing the interest values and performing non-maxima suppression.

Intrinsic Shape Signatures 3D

Intrinsic Shape Signatures detector was introduced by Zhong in 2009 [37].

The method is based on a saliency measure, for each point in the point cloud, based on the Eigenvalue Decomposition (EVD) of a weighted scatter matrix $\Sigma(p_i)$. More precisely, for each point p_i in the point cloud P, weight w_i inversely proportional to the number of points in its spherical neighborhood of radius r is associated:

$$w_i = \frac{1}{\|p_j : |p_j - p_i| < r\|} \tag{4.3}$$

this point-wise weight is useful to compensate for the irregular sampling of the point cloud. Then, $\Sigma(p_i)$ is calculated as:

$$\Sigma(p_i) = \frac{\sum_{|p_j - p_i| < r} w_j (p_j - p_i)(p_j - p_i)^T}{\sum_{|p_j - p_i| < r} w_j} \tag{4.4}$$

Let $\lambda_1(p_i)$, $\lambda_2(p_i)$, $\lambda_3(p_i)$ its eigenvalues in decreasing order of magnitude, a pruning phase is provided. All points whose ratio between two successive eigenvalues is below a given threshold are retained:

$$\frac{\lambda_2(p_i)}{\lambda_1(p_i)} < Thr_{12} \wedge \frac{\lambda_3(p_i)}{\lambda_2(p_i)} < Thr_{23} \tag{4.5}$$

Among other points, the saliency is definite as the magnitude of the smallest eigenvalue $\lambda_3(p_i)$ with the aim to choose points with a high variation on the other two principal directions.

4.1.2 Features description

After the keypoint detection, a features description step is mandatory to create a descriptor useful to allow tasks such as registration or object detection.

Usually, 3D local feature descriptors leverage histograms to summarize different features in the local neighborhood of each keypoint. The used features can be a combination of both topological and geometrical ones. In [38] authors classify 3D feature descriptors into two categories: *spatial distribution histogram* and *geometric attribute histogram* based descriptors. The farthest represents the local surface by creating a histogram using the spatial point distribution. Usually, they start with the construction of a Local Reference Frame (LRF) with the origin in the keypoint and they divide the space around it. Then the histogram is created by

accumulating points in each sub-region. The geometric attribute histogram-based descriptors represent the local surface by creating a histogram using geometric features, such as normals or curvatures, of the points in the keypoint neighborhood.

3D Shape Context

The 3D Shape Context (3DSC) descriptor [39] is the 3D extension of the 2D version [40]. It is a spatial distribution histogram-based descriptor.

The algorithm takes a spherical region around the keypoint p with its north pole aligned to the estimated surface normal in p. This region is then equally partitioned in the azimuth and elevation dimensions and logarithmically in the radial one. From the resulting grid, a histogram is created associating each grid sub-region to a bin. Each bin of the histogram contains a value calculated as the weighted sum of the points falling in the corresponding sub-region. The weight associated with the keypoint is inversely proportional to the local point density and the sub-region volume to compensate the large variation in volume sizes with radius and elevation. As said before, the spherical grid is aligned with the normal in the keypoint, but there is no alignment on the normal plane. This is managed by considering all the possible orientations as the azimuth subdivisions, resulting in multiple descriptions of each keypoint.

Unique Shape Context

The Unique Shape Context (USC) descriptor [41] is an extension of the 3DSC and, as it, it belongs to the spatial distribution histogram-based descriptor category.

The main difference between USC and 3DSC is that USC defines an LRF for each keypoint with an unambiguous orientation. It is possible by yielding not only the surface normal in the keypoint but also a unique couple of directions on the tangent plane. This reduces the memory footprint and the ambiguity in matching resulting in higher accuracy and efficiency. When the LRF is uniquely fixed, the spherical grid is created and the descriptor is computed in the same way as 3DSC.

Point Feature Histogram

Point Feature Histogram (PFH) [42] is a geometric attribute histogram-based descriptor introduced in 2008 by Rusu et al.

It works with point pairs in a point neighborhood. For each couple of points in the k-neighborhood of the point p, the Darboux frame is created leveraging the surface normals and points' position. Then, for each couple of points, four features are calculated based on the Darboux frame, surface normals and point positions. Thereafter, these features are accumulated in a histogram to create the descriptor. The PFH descriptor dimension is dependent on the number of histogram bins along each dimension.

Fast Point Feature Histogram

PFH descriptor is a good 3D feature descriptor, but its computational complexity doesn't make it suitable for real-time or near-real-time applications. For this reason, in 2009 Rusu et al. [43] proposed the Fast Point Feature Histogram (FPFH) descriptor, a more efficient version of the PFH.

The FPFH descriptor algorithm consists of two steps:

1. for each point, a Simplified Point Feature Histogram (SPFH) is created by comparing it with his neighborhood exactly as in PFH. Unlike PFH, SPFH uses only three features to create the descriptor.

2. FPFH descriptors are then created as a weighted sum of the pre-computed SPFH descriptors of the keypoint and its neighbor points.

In this way, the complexity of FPFH is greatly reduced compared with the PFH making it suitable for real-time applications.

Signature of Histograms of Orientations

Signature of Histograms of Orientations (SHOT) was first introduced in 2010 by Tobari et al. [41] and finalized in 2014 by the same authors [44]. It is inspired by the 2D SIFT descriptor.

The algorithm defines an LRF invariant to rotations and translations and robust to noise and clutter in each keypoint. Then a spherical region is defined and neighboring points inside the sphere are aligned to the respective LRF. The sphere is divided into 32 bins: 8 along the azimuth, 2 along the radius and 2 along the elevation. In each sub-region, a one-dimensional local histogram is computed using as features the cosine of the angle between the normal of the keypoint and the neighboring points in that volume. The final descriptor is created by stitching all the computed local histograms and has a length of 352.

4.2 Deep Learning on Point Clouds

The keypoint detector and feature descriptor algorithms described before are generally handcrafted for specific tasks. With the diffusion of deep learning and the success of CNN with images, several deep learning approaches to deal with point clouds have been proposed.

Convolutional architectures require a regular data structure such as images or 3D voxels, in which data are organized respectively in a bi-dimensional or three-dimensional grid. 3D point cloud data, on the contrary, are unorganized since they are usually given as a set of measurements in a continuous domain. In [45], authors individuate three main proprieties of 3D point clouds that need to be considered in a deep learning approach:

- **Unordered:** a point cloud is a set of points without a specific order. For this reason, a neural network that takes in input N points has to be invariant to the $N!$ permutation of the same points.
- **Interaction between points:** The points measured in the space are not isolated, but neighboring points form meaningful sets. This means that the network has to be able to capture these structures since containing meaningful information.
- **Invariance under transformations:** since data represent 3D objects, the extracted features should be invariant to several transformations, such as

changes of point of view.

In literature, several surveys reviewing deep learning approaches for 3D point clouds were proposed [46, 47, 48, 49]. A diffused classification method is based on the input data representation and network structure [50].

Based on the input data representation, it is possible to divide it in:

- **Projection-based Networks:** these methods project the unstructured point cloud on an intermediate representation and leverage 2D or 3D convolution to learn features. These methods can be subdivided into:
  - *Multi-view representation:* [51, 52, 53, 54, 55] these methods project the point cloud or the three-dimensional mesh into multiple 2D images acquired from different points of view and use standard CNNs to extract features from that. These methods achieved good results in shape classification and retrieval tasks but it is difficult to extend them to scene understanding.
  - *Volumetric representation:* [56, 57, 58] these methods leverage the voxelization operation - which transforms a raw point cloud in an organized 3D voxel grid - and use 3D convolutions to extract features. They suffer from data sparsity and are computationally expensive.
- **Point-based Networks:** these networks are able to directly consume raw point clouds and can use different methods to extract features from a single point. Methods in this category can be divided into:
  - *Point-wise Multi-Layer Perceptron Networks:* [59, 60, 61] these methods extract features from each point using MLP layers and then aggregate point-wise feature vectors to obtain a global descriptor. They can guarantee permutation invariance but not all the relationships among points are considered. Pioneer works in this category are PointNet [45] and its extension PointNet++ [62].

- *Convolution-based Networks:* these methods compute 3D convolutions directly on the points. Approaches of this type can use continuous [63, 64, 65] or discrete [66, 67, 68] convolutional kernels.
- *Graph-based Networks:* these methods model the point cloud as a graph in which each point represents a node and edges are generated by defining a neighborhood function. Features can be learned in the spatial [69, 70, 71] or spectral [72, 73, 74] domain.
- *Data-indexing based Networks:* [57, 75, 76] these methods leverage different data indexing structures - as kd-tree or octree - to learn features in a hierarchical way.

Chapter 5

Computer Vision for Autonomous Vehicles

The development of driverless cars has the potential to revolutionize the future mobility of people and goods.

Self-driving cars represent a field in which we are witnessing rapid evolution in the last decades. This evolution affects not only academic research - in which is a very hot topic - but also car manufacturers and mass production. A driverless car means a car with a given level of automation can move with low or no human interaction. To this aim, driverless cars are equipped with a great variety of sensors, such as cameras, lidar, radar, GPS, IMU. An advanced control system fuses the data from all sensors to perceive the surrounding environment extracting higher level information to identify the more appropriate navigation path and react to outside situations. In this scenario, it is easy to imagine how computer vision has a fundamental role.

Nowadays, all cars on the market are equipped with many ADAS (Advanced Driver Assistance Systems). These are systems able to assist humans while driving with the aim to make cars safer. The Society of Automotive Engineers (SAE) defines 6 levels to categorize vehicle autonomy. These levels can be understood as: *Level 0* - no driving automation; *Level 1* - driver assistance; *Level 2* - partial automation; *Level 3* - conditional automation; *Level 4* - high driving automation

and *Level 5* - full driving automation.

This chapter will present two works in the autonomous vehicles field. Both will exploit neural networks to handle two different computer vision tasks. In particular, the first one [77] faces up the problem of tracking a car on point clouds while the second one [78] will treat the monocular depth estimation task.

5.1 Car Tracking on Lidar Point Clouds

The introduction of driverless cars on today's cities implies the sharing of roads with many other elements such us pedestrians, cyclists and other not autonomous vehicles. For this reason, understanding the behaviour of each element in the road is a fundamental requirement for an autonomous vehicle to reach higher safety level. To drive in a correct and safe way, autonomous vehicles need to perform several tasks. For instance, road detection and road sign recognition are useful to understand where and how to drive, object detection is crucial to collision-free path planning while object tracking serves to predict other road agents' motion to avoid dangerous situations.

Driverless cars need to leverage both geometrical and appearance features of the surrounding environment to extract high level and precise information from the scene. Most autonomous driving applications leverage RGB images because these contain important appearance features and it is easy to recover the geometric ones by inferring depth using stereo rings or a single camera. RGB cameras, however, are really susceptible to unfavorable weather and conditions. On the other hand, lidars are able to produce point clouds with really accurate geometric features, they have a larger FOV and are more robust to light and weather untoward conditions but at a cost of loss in appearance features.

Object tracking is a long-studied problem in computer vision and robotics fields because it allows some more complex tasks such as multi-robot cooperation and human-robot collaboration. Object tracking goal is the estimation of a target object state through time. It is usually modeled as a search for the candidate that best match a model.

All the object tracking algorithms are susceptible to error drift over time. Usually, they are modelled as feed-forward algorithms since there is no way to correct the estimation. Several object tracking using point cloud data have been proposed in literature. Most state-of-the-art algorithms are on tracking-by-detection methods, which work by detecting the object first and then solving a data association problem. The aim of this work is to leverage this idea to increase the accuracy of an out-of-shell 3D car tracker.

This section aims to explain a possible method to mitigate the consequences of the error drift over time in a 3D car tracker developed to track a single car in a point cloud tracklet. The proposed solution uses a detection algorithm to correct the tracking estimation with a given frequency that can be flexibly tuned by the system designer. In particular, a new system which introduces a detection branch to an out-of-shell 3D car tracking algorithm is proposed. The experiments will demonstrate how this approach can mitigate the tracker error drift. The introduction of the detection branch implies the tuning of a new parameter, but it is indispensable for real applications.

5.1.1 Related Work

3D Object Detection

A 3D object detector is a system that aims to detect all the objects of a given category in an input point cloud. The output of this system is a list of oriented 3D bounding boxes, each containing one of the detected objects.

Yulan Guo et al. [50] divide 3D object detection methods into two categories: region proposal-based and single-shot methods. The former methods propose several regions (called proposals) that probably contain objects and then classify objects contained in every proposal. Usually, these methods are time-consuming and greedy for resources. Single-shot methods, instead, directly calculate probabilities and regress 3D bounding boxes, without the need for the region proposal step. MV3D [79] represents a milestone in 3D Object Detection. This approach leverage both RGB images and lidar point clouds. It uses a multiview encoding of the

input point cloud by projecting it on both Bird's Eye View (BEV) and front view images. BEV image is used to create the 3D proposals which are then projected in all other views. Then, a deep fusion network is used to combine region-wise features which are finally used to predict the oriented 3D bounding box and the object class.

In [80] authors proposed an object detection method based on RGB-D data. The algorithm leverage the best features of both RGB images and 3D data. The authors exploit a mature 2D object detection method based on CNN to generate proposals and classify objects. Then, the proposals' projection on 3D space defines a frustum that delineates the 3D searching space. Finally, using a PointNet-like [45] neural network authors perform an instance segmentation of points inside the searching area and estimate the 3D bounding box.

VoteNet [81] is another approach proposed by Qi et al. that uses Hough voting to detect objects in indoor scenarios. Authors propose a 3D extension of Hough voting based on neural networks to estimate objects' virtual center points from point clouds and generate proposals.

3D Object Tracking

Based on the addressed problem, it is possible to recognize two different categories of object tracking algorithms: Single-Object Tracking (SOT) and Multiple-Object Tracking (MOT). SOT methods' goal is to estimate - in each frame - the state of a single specific object given its state in the first frame. MOT methods, on the contrary, points to estimating - in each frame - the state of all the objects of the same category in the scene. This task is usually tackled with the before mentioned tracking-by-detection.

In 2018, Lou et al. proposed Fast and Furious [82], an approach able to jointly handle 3D object detection, tracking and motion forecasting on 3D data. The authors used a deep neural network that performs 3D convolutions across space and time on BEV projections of 3D data. The resulting method is very fast and robust to occlusions and data density.

Complexer-YOLO [83] fuses deep-learning-based 3D object detector and se-

mantic segmentation to perform 3D object detection and tracking in autonomous driving scenarios. It leverages a voxel-based version of the previous work Complex-Yolo [84] which is a 3D extension of YOLOv2 [85]. Complexer-YOLO is a MOT method which leverages both RGB images and point clouds acquired in the same scene with temporal information to increase accuracy and robustness.

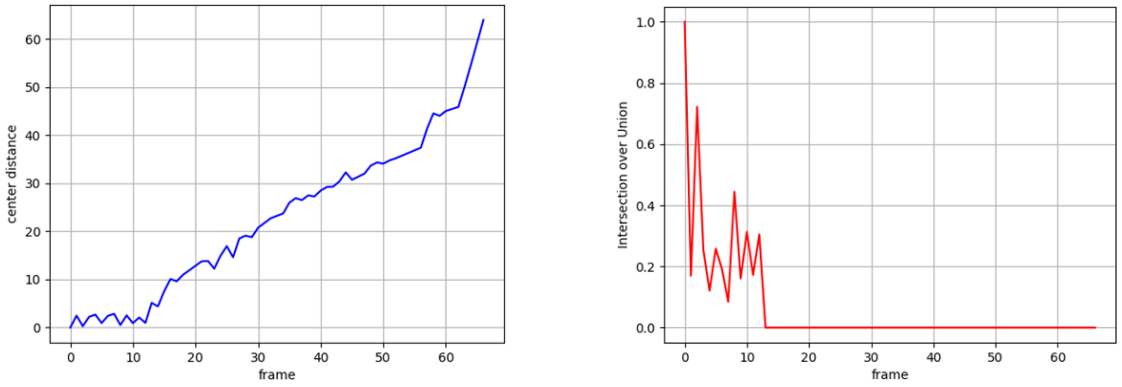
To reach a higher level of efficiency, in 2020 qi et al proposed Point-to-box (P2B) [86]. This method uses Hough voting to localize potential targets by centers, avoiding Kalman filtering or the exhaustive search that represents the bottleneck of most tracking methods in terms of performance. Then it performs jointly target proposal and verification by clustering detected centers neighborhoods.

In [87] a more probabilistic approach to MOT problem is proposed. It follows the classic tracking-by-detection approach. It uses a Kalman filter in which the observations are given by the detection. For data association, the authors propose to use the Mahalanobis distance because more suitable than 3D-IoU.

3D-SiamRPN [88] is a SOT system based on a Siamese Network. The network is composed by two sub-networks. The first sub-network uses PointNet++ [62] architecture to extract features from the input point cloud and fuses them using two cross-correlation modules. The second one is a region proposal network (RPN) and uses the features from the previous step to regress a 3D bounding box.

5.1.2 Proposed Approach

The error drift problem is common to all object tracking algorithms. To better highlight this, in Figure 5.1 the center location error and the IoU of a state-of-the-art 3D car tracking algorithm [89] - on a single point clouds sequence - are plotted. The center location error is defined as the Euclidean distance between the centers of the ground truth bounding box and the estimated one. It is easy to note that the error increases over time. As explained before, here a new 3D SOT system is proposed. It add a detection branch to an out-of-shell 3D tracking algorithm with the aim of reduce the error accumulated over time. Figure 5.3 shows the proposed system pipeline which performs 3D car tracking on point clouds and is composed by two main modules: the tracker and the detector.



(a) (b)

Figure 5.1: Example of error drift of a state-of-the-art object tracking algorithm in a single sequence. (a) Plot of the distance between the center of the estimated object's bounding box and the ground truth one. It increases with the increasing of the frames. (b) Intersection over Union between the estimated object's bounding box and the ground truth one. It decreases with the increasing of the frames reaching zero.

Tracking branch: This branch has the aim to perform tracking of the specific car in the frame f_t given his state (expressed as a bounding box) given his state in the frame f_{t-1}. In particular, let x_{t-1} the bounding box estimated in previous frame, $\hat{x}_{t-1}$ the model point cloud obtained by aggregating all cropped points inside $x_{0,\dots,t}$ and p_t the point cloud acquired at time t. The bounding box x_0 is assumed to be known. Given pt there are several techniques - such as: Bayesian filters (Kalman filter or Particle filter), Exhaustive Search or RPN - to regress a given number of proposal bounding boxes around x_{t-1}. The points inside each proposal bounding box are cropped and features are extracted. By comparing the features extracted from each proposal with which extracted from the model $\hat{x}_{t-1}$ it is possible to choose the best match. The bounding box of the best match is chosen as the tracking result x_t and its relative points are used to update the model $\hat{x}_t$.

Detection branch: This branch has the aim to detect all cars in the scene and choose the tracked one. As shown it contemplate two steps: detection and data association. First, the detection step regress a bounding box around each car in the point cloud p_t. To do it is possible to use a state-of-the-art detector. Then it is possible to use a metric to choose which of these bounding boxes is more

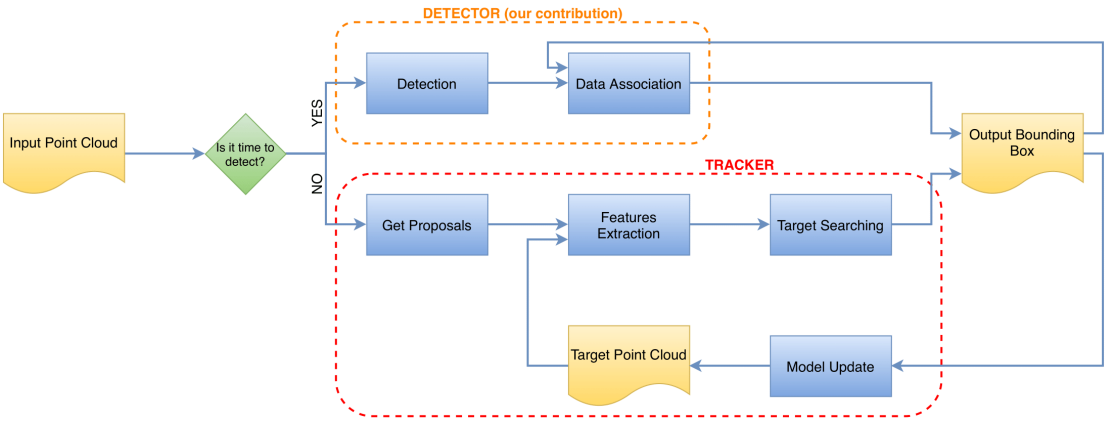


Figure 5.2: A flowchart representing the pipeline of the proposed system.

probably to be the target one x_t. Then the points inside x_t are aggregated in the the model $\hat{x}_t$. Given the independence of this branch from the estimation x_{t-1} it reinitialize the tracker reducing the accumulated error drift.

At each time instant t, only one branch is executed. Which of the two branch execute is given by a predefined *reset frequency*.

5.1.3 Used Architecture

Both tracking and detection branches can be made in really different ways and can follow very different logic. To implement and validate the proposed method, two out-of-shell approaches present in literature have been used. For both branches, methods using only point clouds as input data have been chosen.

The approach proposed by Giancola et al. [89] is used for the tracking branch. It is a SOT method based on a PointNet-like Siamese Network. The system proposed takes as input a tracklet in which in each frame the target object exists. Given the point cloud p_t, it uses an Exhaustive Search (also Bayesian filters can be used) to sample proposals around the bounding box x_{t-1}. Points inside each proposal are cropped and encoded in a latent vector z_c. The model $\hat{x}$ is maintained over time. It is obtained by accumulating points of the estimated bounding box in each time instant. The cosine similarity is used to compare each encoded proposal with the encoded model $\hat{z}$ in the latent space. The bounding box relative to en-

coded vector z_c that maximizes the cosine similarity is chosen to be the output of the system at time t. Moreover, this method leverages shape-completion regularization to embed generative features in the latent vectors, useful in discrimination.

For the detection branch, PointVoxel-RCNN (PV-RCNN) [90] is used. This approach integrates both 3D voxel CNN and PointNet++[62] set abstraction to increase detection performance. It leverages a 3D voxel CNN with sparse convolution to multi-scale semantic features encoding and proposal generation. The proposal generation follows an anchor approach by projecting the output of the 3D sparse convolution-based encoder into a BEV feature map. The features extracted by the encoder are then summarized in a small set of keypoints through the voxel set abstraction (VSA) module. VSA acts as a bridge between the 3D voxel CNN features encoder and the proposal refinement network. With an approach similar to set abstraction layer of PointNet++ [62], VSA can represent the scene thought local features. After the scene is encoded using a small set of keypoints, in order to perform proposal refinement, they are weighted. Intuitively, keypoints belonging from background should contribute less than ones belonging from foreground. For this reason, Predicted Keypoint Weighting (PKW) module is used to weight keypoints by segmentation extra-supervision. At this point, a RoI-grid pooling module is used to aggregate keypoints features to the RoI-grid points with multiple receptive fields by set abstraction. Given the RoI features of each bounding box, the proposal refinement network learns to predict bounding box residuals and a confidence score based on the IoU between the estimated and ground-truth 3D bounding box.

5.1.4 Experiments

Here the experimental setup used to test the proposed approach is described. **Dataset.** All the experiments fundamental to testing the developed system were executed on KITTI dataset [91]. As in [89] only its training set was used splitting it in the following way: scenes from 0 to 16 to training the system, scenes 17-18 for validation and scenes 19-20 as test set. The dataset was adapted for SOT systems creating a single tracklet for each car identified in the scene.

Reset Frequency	Success	Precision	Cost
1	45.53	59.547	98.132
1/2	42.6813	55.2459	48.6457
1/5	42.5988	54.4742	18.9134
1/10	36.6999	45.5273	9.0286
1/15	38.6761	49.9222	5.7752
1/25	31.32	39.1131	3.0199
1/50	26.7987	33.3134	1.183
1/100	25.2444	31.7466	0.4202
Only tracking	24.0823	30.007	0

Table 5.1: Experimental results. Success and Precision are defined as in the One-Pass Evaluation. The Cost is defined as the percentage of the number of frames in which detection is used on the total number of frames.

Evaluation Metrics. To evaluate the complete system, as common in literature, the One Pass Evaluation (OPE) is used [92]. It uses the IoU to measure the overlap between the estimated and the ground truth bounding boxes, and the error as the center location error between the same bounding boxes. Success and Precision metrics are defined as the Area Under the Curve (AUC) of respectively IoU and distance error (from 0 to 2m). Furthermore, a normalized Cost is defined by the number of frames in which the detection was used. It is useful to express, in a simple way, a measure of the efficiency of each experiment to perform a comparison between them.

5.1.5 Results Discussion

In Table 5.1 are reported the experimental results. It can be noted that both success and precision metrics are directly proportional to the reset frequency. Indeed, an increase in the reset frequency corresponds to an increase in performance. This result is interesting because confirms the importance of reinitialization in the tracking task to reduce the error drift. On the other hand, as shown by the cost metric, the detection branch can introduce a cost in terms of efficiency that ob-

viously increase with the reset frequency. For this reason, for real applications, the reset frequency has to be accurately chosen as trade-off between performance and efficiency. It is, thus, strictly dependent on the specific task and the available computational resources. This commonly involves the investigation of several parameters such as the inference time of both tracking and detection modules, their performances, the task requirements and the used hardware.

Just as an example, based only on the experiments performed on KITTI with the described setup, it is possible to identify the reset frequency of 1/15 as a good trade-off. In fact, the performance slightly increases compared with 1/10 and the cost is lower (the detection is used only on about 5% of the total frames).

5.1.6 Summary

In this work, the importance of reinitialization in a tracking algorithm is shown. First, the problem is highlighted by testing a 3D car tracking algorithm on a dataset with long sequences. Then an approach to solving it by using a detection branch is proposed. In particular, the proposed method leverage a 3D point cloud car detector to reset the tracking module with a given frequency. With large-scale experiments on KITTI dataset, the advantages of this approach are shown by reaching good improvements compared with the baseline. Finally, the importance of reinitialization in real applications is highlighted and the way to choose a good reset frequency is explained.

Future work will include the investigation of methods to execute detection only when strictly necessary with the aim of increasing system efficiency. Moreover, given the generality of the proposed pipeline, further works will also include the use of different sensor data to take advantage of the different features of each of them.

5.2 Monocular Depth Estimation

In Chapter 4 several sensor families to perceive 3D data are described. An accurate environment perception is at the earth of all applications characterized by a system

that moves in the 3D real-world. All sensors shown before have advantages and disadvantages. Because of this, the system designer has to carefully choose the sensor that better fits requirements. For instance, lidars are really accurate sensors, but they are generally expensive and do not represent a good choice for low-cost vehicles and robots. Moreover, for autonomous driving applications, they have to be mounted in a particular position that does not permit integration on the car changing both the appearance and the aerodynamics. For this reason, most autonomous driving applications are based on other sensors such as RGB cameras or radars.

There are several methods to infer depth from RGB cameras such as stereo cameras and structure from motion. It is obvious that the depth estimation from a single image is an ill-posed problem since the decay of the epipolar geometry constraints. Humans are able to perform well this task by unconsciously making assumptions to respect objects' dimensions [93].

Recently, a huge number of works has been proposed in literature that aims to use deep learning to address the monocular depth estimation problem. Many of these are supervised approaches [94, 95] which rely on annotated datasets. Even if they are able to reach amazing results, collecting a large and varied dataset with precise depth annotation is, in practice, challenging and labor-expensive. For this reason, some works propose leveraging the photometric reprojection constraints to develop self-supervised methods. Unlike supervised approaches, self-supervised ones need multi-views images - e.g. acquired with stereo cameras - of the same scene and thus do not need precise round truth depth data. This type of data is simple to acquire and its annotation requires less effort.

Most of the methods present in literature use very deep CNN that can learn a high number of parameters to learn important generative features. However, these methods are power- and resource- consuming and need high-end GPUs to produce results in real time.

In this section, a new self-supervised approach based on knowledge distillation is proposed. The aim is to exploit the knowledge guarded in a deep pre-trained CNN to transfer it to a smaller one.

5.2.1 Related Work

Estimation of depth from a single image represents an ill-posed problem since a single image is available at inference time and it is not possible to rely on geometrical constraints. For this reason, it is a deeply studied topic for several years. Most works focused on the use of stereo images [96], multiple monocular images acquired from different points of view [97], at different time instants [98] or making some assumption about the scene, e.g. static scene [99] and viewpoint with different light [100]. Here a brief review of Supervised and Self-supervised methods is reported.

Supervised Monocular Depth Estimation

Supervised learning for monocular depth estimation need ground truth depth maps to train the model. The aim of the training is to minimize a loss function mainly based on the similarity between the ground truth and estimated depth maps.

One of the first models proposed in this category is Make3D [101]. It leverages the over-segmentation of the input image to estimate 3D local location and orientation of local planes that compose the scene. This method, however, tends to fail with thin structures and does not consider the global structure of the scene. In [95] a CNN model was proposed to deal this task, while Ladicky et al. [102] leverage semantic information to improve the results. Karsch et al. [103] account the problem as an image matching problem using temporal information to refine results.

Eigen et al. [94] proposed a multi-scale CNN trained following the supervised paradigm. Differently from most of work explained until now, this approach infers depth from the raw pixels without leverage any segmentation or hand-crafted features. Relying on Deep3D [104], Luo et al. [105] modeled the monocular depth estimation problem leveraging stereo geometry, creating the right view in a synthetic way. DepthNet [106] rely recurrent neural networks (RNN) to learn spatio-temporal information to predict depth maps from monocular video sequences.

In [107], authors aim to tackle the problem for a real world applications point of view. They account the task as a image-to-image translation problem. The

proposed method is based on an adversarial network trained on synthetic data able to generalize enough to be domain independent.

Other proposed approaches aim to jointly face up monocular depth estimation with other tasks. DeMoN [108] is a model based on a chain of encoder-decoder networks able to jointly estimate both depth and ego-motion from monocular videos. Chen et al. [109] developed a model to jointly estimate depth and semantic segmentation while ViP-DeepLab [110] estimate it jointly with panoptic segmentation.

All this reviewed methods need large datasets composed by images with their respective ground truth depth maps. The creation of these datasets is not trivial, therefore often it is preferable to explore other possibilities.

Self-supervised Monocular Depth Estimation

To overcome the problem of dataset creation of supervised learning methods and utilize relative cheap data, many works proposed to self-supervised methods. In 2016, Flynn et al. proposed DeepStereo [111] which is trained in an unsupervised way relying images representing the scene from different points of view. Since it needs multiple images also in inference phase, it is not suitable for monocular depth estimation task. Deep3D [104] meets the novel view estimation problem in a binocular setup. It leverages an image reconstruction loss generate the right view from the left one in the 3D movies domain.

Zhan et al.[112] proposed an encoder-decoder network for monocular depth estimation on a stereo setup trained to minimize a reconstruction loss. Their image synthesis is not fully differentiable, for this reason they linearized the loss using a Taylor approximation making the training more difficult. This problem is overcome in Monodepth [113] by using bi-linear sampling. It consist in a model that, at training time, estimates depth from both left and right images in a stereo setup to rely on left-right consistency. Inspired by it, Aleotti et al. [114] proposed an adversarial training to face up this task, while Poggi et al. [115] proposed a thin network to deal the same task using the CPU in a trinocular configuration.

Zhou et al. [116] proposed an approach to synthesize depth-maps on monocular videos. In the training it aims to minimize a reconstruction loss between temporal

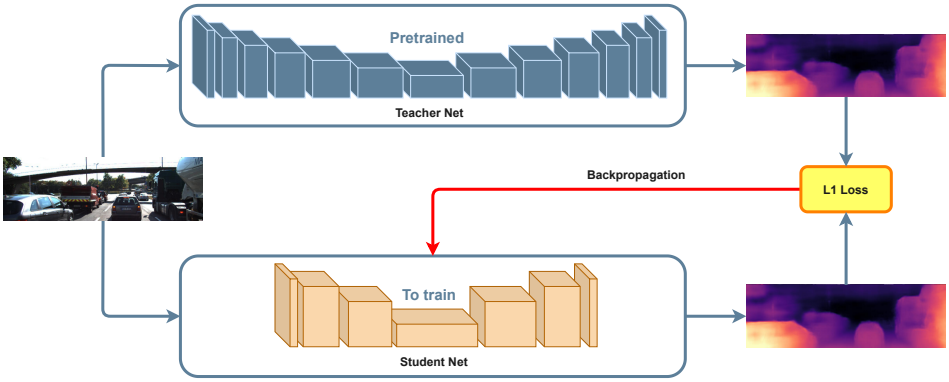


Figure 5.3: Proposed method for Knowledge Distillation to monocular depth estimation. The method uses the output of a pre-trained network as supervision signal. In particular, a pre-trained model of Monodepth2 was used as teacher network. The student network (DeepLabv3+) was trained from scratch relying on a simple L1-loss between the output of the teacher network and the one of the student network.

subsequent frames. Furthermore, it train a pose network to estimate the relative camera pose between them.

With a similar goal, Mahjourian et al. [117] introduced an ICP-based loss to jointly estimate depth and ego-motion from monocular videos.

Later, Godard et al. [118] extended their previous work. Here, they use a model trainable on both stereo pairs and monocular video configuration. The focus of this work is on artifacts mainly due to occlusions and moving objects.

5.2.2 Proposed Method

Here the knowledge distillation [119] concept is explained with a particular focus on vanilla distillation which is used in the proposed method. Then the two networks used respectively as teacher and student models will be reviewed. Figure 5.3 represents the pipeline used to train the student model.

Knowledge Distillation

Knowledge distillation represents one of the most effective techniques to compress and accelerate deep-learning models. It is based on the idea of inferring the knowl-

edge acquired by a very deep neural network - the teacher model - in a smaller one, the so-called student model. This technique is particularly useful when there is the need to use deep models on devices with limited resources such as mobile devices or embedded systems.

Gou et al. [119] classify knowledge distillation algorithms from the perspective of knowledge categories, training schemes, teacher-student architecture, and distillation algorithms.

In this work, an offline distillation scheme is used where the knowledge is transferred from a pre-trained teacher network to a student one. The training process consists of two steps:

- *Teacher training:* the teacher is trained on a set of training data.
- *Student training:* the student model is trained from scratch with the supervision of the teacher model. The knowledge extracted from the teacher model can be in the form of intermediate features or the output of the model.

In this work, the vanilla knowledge distillation scheme is adopted which provides the use of the teacher output as supervision signal to train the student model. In particular, the student network is trained to minimize the L1 loss between the output depth map of the teacher and student networks.

Let F_T the teacher model, F_S the student model and I the input image, the L1-loss is defined as:

$$L_1 = \frac{1}{N} \sum_{p=0}^{N} |F_T(I)_p - F_S(I)_p| \tag{5.1}$$

where p is the pixel index, N is the total number of pixels in the image, $F_T(I)_p$ and $F_S(I)_p$ represent the p-th pixel of the output image respectively of the teacher and the student model.

Teacher Network

As mentioned in the previous sections, the adopted knowledge distillation pipeline uses an offline training approach transferring the teacher's knowledge to the stu-

dent network. Here as teacher network, a pre-trained model of Monodepth2 [118] is used.

Monodepth2 was presented in 2019 by Godard et al. as an extension of their previous work [113] reaching good results on the monocular depth estimation task. The aim is to train a CNN in a self-supervised setup to estimate an accurate depth map given a single image in both stereo and monocular setups. The approach accounts the learning problem as a new view-synthesis task, in which the network is trained to synthesize the appearance of a target image that represents the same scene from a different point of view. In other words, the method accounts the monocular depth estimation task as a photo-metric reprojection error minimization problem.

Let $T_{t\to t'}$ the relative pose between the input image I'_t pose w.r.t. the target image I_t one. The model has to predict the depth map D_t that minimizes the photometric reprojection error L_p defined as:

$$L_p = \min_{t'} pe(I_t, I_{t\to t'}) \tag{5.2}$$

with

$$I_{t\to t'} = I_{t'}\langle proj(D_t, T_{t\to t'}, K)\rangle \tag{5.3}$$

where pe is the photo-metric reconstruction error, $proj(.)$ is the projection of the depth map D_t in $I_{t'}$, the $\langle.\rangle$ operator indicates the bi-linear sampling operator [120] while K is the camera intrinsics matrix. To calculate pe, authors rely on L1 and SSIM [121] errors:

$$pe(I_a, I_b) = \frac{\alpha}{2}(1 - SSIM((I_a, I_b)) + (1-\alpha)\|I_a - I_b\|_1 \tag{5.4}$$

where $\alpha = 0.85$. Furthermore, in the training authors used a smoothness loss L_s defined as:

$$L_s = |\delta_x d_t^*|e^{-|\delta_x I_t|} + |\delta_y d_t^*|e^{-|\delta_y I_t|} \tag{5.5}$$

where d_t^* represent the mean-normalized inverse depth [122]. The final objective function used during the training is given by a weighted sum of the two defined terms:

$$L = \mu L_p + \lambda L_s \tag{5.6}$$

The proposed architecture supports three different training modalities. It can be trained with stereo images, where $I_{t'}$ and I_t are the two images of the stereo pair. It can also be trained using a monocular setup, in which the two frames adjacent to I_t are used as source images. Since, in this case, the relative pose between frames $T_{t\to t'}$ is not known, a pose estimation network is used to predict it. Finally, a mixed training setup is possible, where $I_{t'}$ includes both temporally adjacent images and the second view of the stereo setup.

Student Network

As student network, to reach the aim of realizing a model to tackle monocular depth estimation task using limited computational resources, DeepLabv3+ [123] has been chosen. It is a network specially designed for mobile systems born to deal with the image semantic segmentation task. This network extends DeepLabv3 [124] by adding a simple decoder to it in order to obtain an encoder-decoder structure, useful to refine results, especially on boundaries.

The main intuition behind Deeplabv3 is the use of the so-called *Atrous Spatial Pyramid Pooling (ASPP)*. It is composed of several parallel atrous convolutions with different rates to extract features at multiple scales.

Atrous convolution is a generalization of the standard convolution operation which allows to control the resolution of features computed by CNN and adjust filter's field-of-view to capture multi-scale information. More precisely, in the case of bi-dimensional signals, for each position i on the output features map y, given the input x and the kernel w, atrous convolution is calculated as:

$$y[i] = \sum_k x[i + r \cdot k]w[k] \tag{5.7}$$

where r is the atrous stride, that determines the stride with which the input features map is sampled.

In DeepLabv3, ASPP is implemented after a backbone network that is used to extract features. In DeepLabv3+ the last feature map before logits of DeepLabv3 is used as encoder output; this feature map contains 256 channels. The decoder is

very simple; it first upsamples by a factor of 4 the encoder output and then concatenates it with the corresponding low-level features from the backbone network with the same resolution. Before the concatenation, a point-wise convolution is applied to the low-level features from the backbone just to reduce the channels number. After the concatenation, a few 3x3 convolutions are applied, and finally a 4x bilinear upsampling.

This architecture leverages both the spatial pyramid pooling module and the encoder-decoder structure. The former is useful to capture contextual information at different resolutions while the latter permits to obtain sharper boundaries.

5.2.3 Experiments

Here the experimental setup used to test the proposed approach is described. In all experiments, the pre-trained model of Monodepth2 [118] trained with the monocular and stereo setup (MS) was used as teacher network. The choice fell on this model because it achieved the best results on the selected dataset. All models were trained and tested on KITTI 2015 [125] dataset using an image resolution of 1024x320 by resizing the input images.

Dataset. As in [113], here the KITTI dataset was used. It contains 61 scenes with a total of 42.382 rectified stereo pairs 1242x375 pixels. All experiments were performed using two different splits:

- **Eigen split** [94] uses a test split of 697 images from 29 different scenes. The remaining 32 scenes are divided using 22.600 stereo images for training and 888 for evaluation. To generate the ground truth, in this split, the reprojection of lidar point clouds on the left RGB camera plane image was used.

- **Eigen Zhou** [116] is suited for monocular training and removes from the dataset all static frames. This split contains a total of 39.810 monocular images for training and 4,424 for validation.

In all experiments, the same intrinsics were used. In all images, the principal point has been assumed in the image center and the focal length has been set as

the average of all the focal lengths on the KITTI dataset. The maximum admitted depth was cropped to 80m, as a standard practice.

Evaluation Metrics. To evaluate the proposed approach, all trained models were evaluated for both performance and efficiency. For the evaluation, the metrics proposed by Eigen et al. [94] were used. This work uses different metrics like Abs Relative difference, Squared Relative difference, RMSE, and RMSE (log) which measure the difference in meter with the ground truth depth and other metrics based on the percentage of depths that are within some threshold from the ground truth value. This is useful because the non-thresholded metrics can be sensitive to large errors caused by prediction errors in depth caused by predictions at small disparity values. Furthermore, efficiency was evaluated by measuring the inference time of each model on the different hardware setups. The inference time was calculated as the average of times on 10 different executions. As a general-purpose system, a PC with an Intel i7-5720K CPU which has 6 Cores with 3.30 GHz each, and a GPU Nvidia GTX 970 was used. To test the performances on mobile, a Xiaomi Poco X3 NFC - an Android 10 smartphone equipped with a Qualcomm SM7150-AC Snapdragon 732G Octa-core - was used. For mobile experiments, a simple application based on Pytorch Mobile [126] was created and the model uses only the CPU with multi-threading.

5.2.4 Results and Discussion

In Table 5.2 are reported all the results obtained with the presented setup in comparison with the baseline that is represented by the pre-trained model used as teacher network. The quantitative results show that using the proposed approach with DeepLabv3+ [123] with ResNet101 [127] as backbone trained on Eigen split [94] the results are very similar to the baseline overcoming it in some metrics. Furthermore, as imaginable, using MobileNet [128] as backbone results are worse than other architectures - because it uses fewer parameters - but they still remain quite good. Table 5.2 compares all the analyzed architecture from an efficiency point of view. In particular, there are shown, for each network architecture, the number of trainable parameters and the inference time on different hardware ar-

Architecture	Split	Abs Rel	Sq Rel	RMSE	RMSE log	δ<1.25	δ<1.25^2	δ<1.25^3
DeepLabv3+ w. MobileNet	Zhou	0.108	**0.746**	4.657	**0.193**	0.865	0.957	**0.981**
DeepLabv3+ w. ResNet50	Zhou	0.121	0.975	5.051	0.216	0.841	0.937	0.967
DeepLabv3+ w. ResNet101	Zhou	0.109	0.869	4.649	0.250	0.873	**0.958**	0.980
DeepLabv3+ w. MobileNet	Eigen	0.110	0.871	5.139	0.333	0.863	0.955	0.980
DeepLabv3+ w. ResNet50	Eigen	0.107	0.771	4.666	0.199	0.865	0.953	0.978
DeepLabv3+ w. ResNet101	Eigen	**0.104**	0.752	**4.552**	**0.193**	0.875	**0.958**	0.980
Monodepth2 (baseline)	Eigen	0.105	0.790	4.608	**0.193**	**0.876**	**0.958**	0.680

Table 5.2: Comparison of different models performance. Results on KITTI 2015 stereo dataset with two different splits: Eigen [94] and Eigen Zhou [116]. Here, for the first four metrics lower is better and for the last three higher is better.

Architecture	**# of trainable parameters**	**Single CPU [s]**	**Multi CPU [s]**	**GPU [s]**	**Mobile [s]**
DeepLabv3+ w. MobileNet	**2643281**	**0.4231**	**0.1445**	0.0161	1.429
DeepLabv3+ w. ResNet50	26609233	1.3956	0.3881	0,0369	4.65
DeepLabv3+ w. ResNet101	45601361	1.9663	0.5139	0.06051	9.425
Monodepth2 (baseline)	14842236	0.7365	0.2611	**0.0123**	**1.099**

Table 5.3: Inference time. The table shows the number of trainable parameters and the time (in seconds) necessary to infer a single image with each architecture. The time is calculated as the average of 10 consecutive executions.

chitectures. DeepLabv3+ with MobileNet as the backbone is the network with the lower number of parameters. It has about six times fewer parameters with respect to Monodepth2 architecture. As possible to see in the table, the DeepLabv3+ with MobileNet model is the fastest on single and multiple CPU setups. On the contrary, on GPU and Mobile setup the faster architecture is Monodepth2 [118]

5.2.5 Summary

Here a technique to deal with the monocular depth estimation task with a self-supervised approach on mobile systems was proposed. An out-of-shell pre-trained model was exploited to train a simpler network in a knowledge distillation setup. DeepLabv3+ architecture - which was born for semantic segmentation tasks - was adapted to solve a monocular depth estimation problem in an automotive scenario.

Large-scale experiments were executed on different splits of KITTI dataset [125] to compare several architectures from both performance and efficiency points of view. It is possible to conclude that it is possible to deal with the monocular depth estimation problem using the DeepLabv3+ architecture and the reached results are comparable with the baseline. Unexpectedly, even if DeepLabv3+ model contains

a lower number of parameters, it results slower on Mobile setup compared with Monodepth2 because the latter is more optimizable for Mobile.

In the future will be useful to study a more complex method of knowledge distillation between these two networks leveraging on connections between middle layers and introducing other loss functions typical of this domain based on image appearance and geometrical constraints. Furthermore, it can be interesting to study a new custom model to deal with the monocular depth estimation task able to run in real-time on mobile hardware and test it on different indoor and outdoor datasets.

Chapter 6

Computer Vision and AR/VR in Manufacturing Industry

Nowadays, in the Industry 4.0 era, industries are revolutionizing their processes to make them automated and connect. Industries are investing to achieve leaner and faster processes.

Emerging digital technologies are fundamental in this scenario since they are exploited to improve industries' processes. It is possible to recognize four different development lines: the first one regards the use of data and connectivity which covers big data, IoT, and cloud computing techniques; the second one regards the analytics which means how to extract information from data; the third regards the human-machine interaction; and finally the fourth digital and real worlds interaction covering robotics, machine-machine interaction, additive manufacturing and 3D printing.

Manufacturing industries are the ones that can take more advantage of the new technologies to improve their production process, especially from the last two development lines. Some sectors, like automotive, have achieved leaner and better processes faster than other ones, like the aerospace industry. Furthermore, in the production chain, a few stages gained more automation earlier than others. For instance, assembly and inspection processes present some aspects that make them risky to deal with them automatically. Moreover, with the last years' advancements

in mobile robotics, one of the processes that can gain high automation is surely the warehouse management one.

This chapter will present three different works that exploit new technologies to automatize different industrial processes. [129] and [130] will present two approaches for quality control in the aerospace field. The farthest will exploits robotics and 3D computer vision methods to the control quality of a tail stabilizer aircraft component. The latest will leverage AR and VR techniques for the validation process of control quality results of aircraft interiors. Even if AR and VR seem weakly correlated with computer vision, they often require a 3D scan of the environment and the use of computer vision techniques for localization. Finally, [131] will review a deep learning approach to mobile robot localization in a structured environment. An approach like this can strongly automatize a wide range of processes in manufacturing industries. These three works are all part of different research projects.

6.1 Hardware Calibration and Point Clouds Stitching in Aerospace Manufacturing

This section will present an innovative approach to solving a robot-sensor calibration problem. It is preparatory to the acquisition and stitching of point clouds to the automatic control quality in the aerospace manufacturing industry.

This work is part of the "Integrated Smart Assembly Factory" (ISAF) project aims to integrate innovative technologies in aerospace factories to improve processes' efficiency and effectiveness.

Production processes in aerospace manufacturing can involve a wide range of really different components. At the same time, it is easy to imagine the high safety requirements and thus the high accuracy standards that must meet. Today, the typical approach to quality control of these components is to manually-wise inspect them only after their final assembly. This inspection is fundamental, indeed, for instance, checking the gaps between some adjacent components is necessary to avoid high pre-tension between them and to ensure the reliability of the whole

structure. Possible gaps between assembled components are manually measured and filled with special shims. Obviously, this procedure can be time-consuming and costly, as well as weighing down the system.

The conducted analysis concerns the assembly of a horizontal stabilizer and the measurement of the gaps between its components. The current procedure provides that components are loaded and measured on a special jig, several times before the stabilizer is completed. The proposed method aims to automatize this process by adopting 3D scanning technologies. These solutions are easily available [132], and they are already highly applied in manufacturing environments [133, 134, 135, 136, 137]. The idea is to use a mobile automated guided vehicle (AGV) with a robotic arm mounted on the top equipped with a 3D sensor. The system will move around the jig to scan the interesting areas. The acquired 3D data will be then used to automatically measure the gaps and assess the need for shims.

This work focuses on the acquisition and stitching of the point clouds coming from the 3D sensor. While the acquiring phase regards the robot's programming to perform the right movements and coordinate with the 3D sensor, the stitching phase is more complex. Many different works are available in the literature on this topic [138]. Given the constraints to be met, the system characteristics, and the features of the components to scan, a suitable approach has been developed. Specifically, a robot-sensor calibration problem has been solved to move the acquired point clouds in a global reference system and stitch them together.

6.1.1 Materials and Methods

Hardware Setup

The adopted methodology provides the use of a mobile automated guided vehicle (AGV) which mounts the Universal Robot UR10e [139] robotic arm and the LMI Gocator 3210 [140] 3D sensor.

The AGV is an omnidirectional mobile platform, capable of navigating through the space by localizing and following a QR-code tape properly placed on the ground. It can be managed through an industrial PC installed on board.

The UR10e [139] is a 6DOF robot arm, ideal for a wide range of applications and widely used in industrial contexts. It has a payload of 12.5 kg and a 1300 mm reach. The UR10e is mounted on the top of AGV through a lift, that allows placing the UR10e base at different heights.

An LMI Gocator 3210 [140] is fixed to the wrist of the robot. It is a 3D snapshot sensor that leverages a stereo camera and a blue light projector to reconstruct the scanned surface in the form of 3D point cloud data. It has a large field of view and can acquire an area of 154x100 mm with a resolution of 34 µm returning a point cloud of about 2.5 million of points in a single snapshot.

Methodology

The proposed methodology can be divided into three steps: calibration of the system, acquisition of the components, and stitching of the point clouds.

The robot-sensor calibration aims to estimate all the 3D transformations to express all the acquired point clouds, that are in the sensor reference system, in a unique global reference system in order to stitch them together. Assuming the UR10e base is stationary, it is possible to express them in a frame integral to it.

Figure 6.1 shows the hardware setup - without the AGV, which position is assumed fixed in this phase - and all reference systems involved in calibration:

- {B} is the robot frame
- {F} is the UR10e flange frame
- {G} is the Gocator frame
- {T} is the target frame, an external reference system used in the Gocator alignment procedure

In particular, {G} is the Gocator uncalibrated reference system, while the target frame T is the calibrated one. It is found by leveraging a specific alignment procedure.

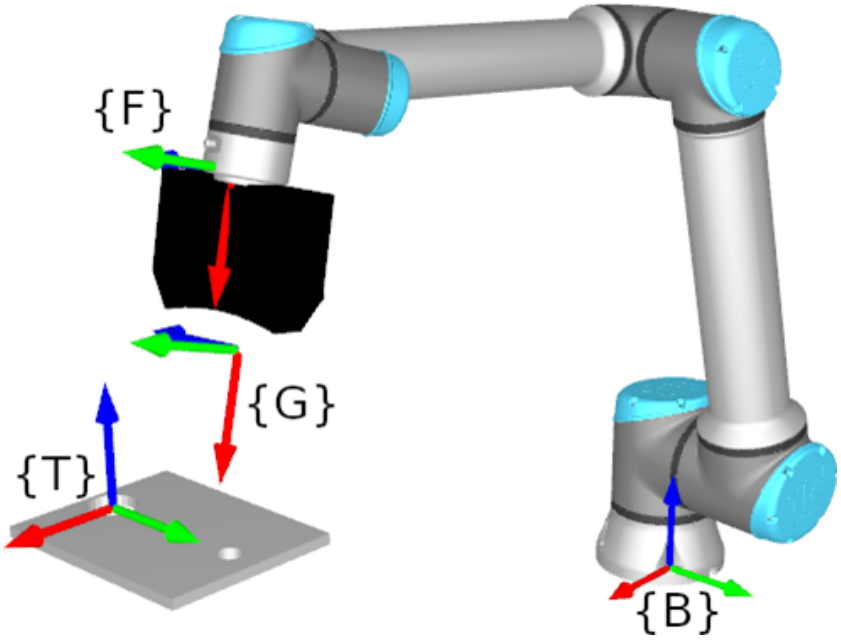


Figure 6.1: The hardware setup with the reference frames involved in the robot-sensor calibration step.

In this specific setup, the aim is the estimation of the fixed 3D transformation ${}^{B}_{G}T$, i.e. the transformation matrix that expresses points from the Gocator uncalibrated frame {G} in the robot base one {B}. The proposed estimation method leverages the whole transformation matrix and a global iterative approach.

Let ${}^{T}P$ a point in {T}, it is possible to express it in {B} using the following equivalence:

$$ {}^{B}P = {}^{B}_{F}T \cdot {}^{F}_{G}T \cdot {}^{G}_{T}T \cdot {}^{T}P \tag{6.1} $$

It is important to note that, given the hardware setup, ${}^{F}_{G}T$ is the same in every robot configuration. On the contrary, ${}^{B}_{F}T$ and ${}^{G}_{T}T$ change as the robot moves. The approach for solving the problem requires performing several measurements of the transformation pair $({}^{B}_{F}T_i, {}^{G}_{T}T_i)$ with $i = 1, \ldots, k$ to iteratively compute the transformation matrix ${}^{F}_{G}T$ that minimizes a custom-defined error function. The estimation of $({}^{B}_{F}T_i, {}^{G}_{T}T_i)$ pairs is straightforward: ${}^{B}_{F}T_i$ is given by the robot odometry, while, as mentioned before, ${}^{G}_{T}T_i$ can be estimated using the Gocator alignment procedure. The alignment step is accomplished by using a metal calibration plate

having two reference holes, one larger than the other. The Gocator system identifies the center of the wider hole as the origin of the new coordinate system, while the direction connecting the two centers of the holes determines the angle from the Z axis.

After a set of $({}^{B}_{F}T_i, {}^{G}_{T}T_i)$ pairs are acquired it is possible to define a grid of N 3D points in {T} and perform a global minimization algorithm to find the only unknown transformation ${}^{F}_{G}T$, based on basin hopping [141] method. In particular, let ${}^{T}X \in \mathbb{R}^{(4xN)}$ be the matrix where each column represents one point of the defined grid in homogeneous coordinates expressed in {T}, for each of the calibration pair $({}^{B}_{F}T_i, {}^{G}_{T}T_i)$ the projection of the grid in {B}, ${}^{B}X_i$, is calculated using the same ${}^{F}_{G}T$. The result of this operation is a tensor $Y \in \mathbb{R}^{(4xNxk)}$. The error function used in the global minimization method is defined as the Frobenius norm of $\sigma(Y)$, where the function $\sigma(.)$ represents the standard deviation on the third axis. In each iteration, the matrix ${}^{F}_{G}T$ is recalculated to minimize the error.

After the calibration step, possible workplans for performing the acquisitions were defined. Workplan means the sequence of movements that the robot must execute to correctly acquire a component of interest and its coordination with the Gocator sensor. Obviously, given the limited scan area of the Gocator, most of the components need more scans to be fully acquired. At first, the workplans have been simulated in a virtual environment. This was useful to understand where to locate the AGV with respect to the component, to define the initial joint configuration, and thus to avoid singularity and collisions. Afterward, the movements have been tested on the real setup. The results of this phase are the distinct point clouds describing the component, and the related robot configurations.

To stitch together the acquired point clouds, for each of them ${}^{G}_{T}T_i$ is retrieved from the UR10e inverse kinematics related to the i-th acquisition. Therefore, the i-th point cloud can be moved into the UR10e base frame {B}, using Equation 6.1. After this procedure for all the point clouds, the stitching phase is complete. The result consists of the comprehensive point cloud representing the whole acquired component. Finally, this is subject to an additional post-processing procedure, in order to make the point cloud lighter in terms of number of points. Specifically,

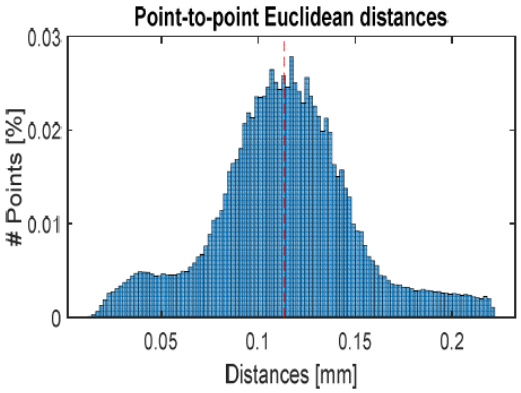


Figure 6.2: Histogram of the point-to-point Euclidean distances between the points of source and the related closest points of target.

closest vertices are merged, outliers are detected through the local outlier probability (LoOP) approach [142] and then removed. After that, the resulting point cloud is further filtered.

6.1.2 Experiments and Results

To evaluate the effectiveness of the proposed method, an analysis of results obtained while stitching two consecutive point clouds is performed. The point distance in the overlapping area can be used to measure the stitching goodness. Indeed, in case the point clouds are well overlapped, the point distance in that area is expected to be the lowest possible, thus proving that the applied approach is robust and effective. Figure 6.2 shows the achieved results for the considered example. As one can notice, the distribution of the point-to-point distances is similar to a Gaussian. In this specific case, it has a mean value of 0.1133 mm and a standard deviation of 0.0372 mm. Looking at the cumulative distribution, the 90% of the point distances are lower than 0.15 mm, while the remaining 10% are lower than 0.22 mm.

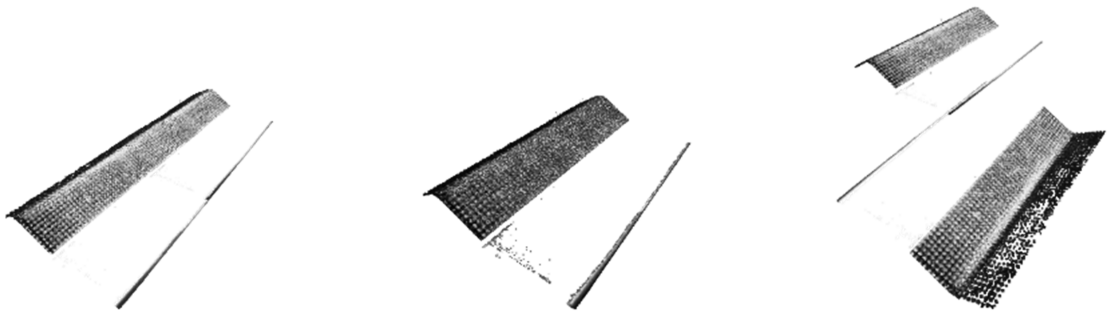
To analyze the results from a qualitative point of view, in Figure 6.3b the comparison of the complete acquisition of a carbon angle profile with its CAD model is shown. The acquisition of the entire component provided several single snapshots. The resulting stitched model perfectly matches its CAD reference.

Comparing the sizes of the two bounding boxes, they differ by 0.5 mm at most.

(a) (b)

Figure 6.3: Comparison between the acquisition (a) and the CAD model (b) of a carbon angle profile

Given the nature of the components to be acquired - carbon fiber components with a very regular surface, without any texture or particular features - no other stitching approaches could be considered. For instance, the diffused ICP algorithm would have failed. Just to completeness, in Figure 6.4 results obtained applying the ICP algorithm with point-to-point [143] and point-to-plane [144] are reported.



(a) (b) (c)

Figure 6.4: Examples of: correct stitching achieved by adopting the proposed strategy (a), mistaken stitching achieved by applying ICP without initial transformation (b), mistaken stitching achieved by applying ICP point-to-point with initial transformation (c).

Without specifying any initial transformation, the ICP fails. Adopting both approaches, itcompletely overlaps the two point clouds, ignoring that one is shifted

with respect to the other. The situation improves using the UR10e kinematics as ICP initial transformation.

However, given the nature of the algorithm, the results are strongly input-dependent and their repeatability is not guaranteed [145]. Moreover, the goal of the quality control process is to identify gaps in the components. This information could be lost by ICP algorithm. Finally, by knowing all the needed requirements, the proposed approach is definitely more robust.

6.1.3 Summary

In this work, a strategy to point clouds acquisition and stitching for quality control in aerospace manufacturing is proposed. To stitch the acquired point clouds, a preliminary robot-sensor calibration step is required. When all the needed transformations are known, several workplans to scan each different component have been defined. Once the acquisitions are completed, single scans point clouds are stitched by moving each of them in the UR10e base reference system leveraging the transformation estimated during calibration.

As future work, it is possible to deeply investigate the calibration problem (for instance about the ideal number of the plate acquisitions or the optimization algorithm). On the other hand, the point clouds acquired will be used for detecting possible gaps and measuring the shims needed to fill them automatically.

6.2 AR/VR for Quality Control of Aircraft Interiors

Automated processes are at the base of modern production processes. The quality control process is often human-performed with a low degree of automation. Usually, it is a time-consuming task requiring inspection officers to keep a high level of attention. For instance, when inspecting the aircraft cargo and cabin lining, workers are required to evaluate both the geometric and appearance aspects of all visible panels. It means that, performing it manually, workers have to perform measure-

ments between panels, like establishing that gap, step, and parallelism respect the requirements. Cognitive fatigue is a common issue of quality control tasks, as well as humans are not able to keep the same attention level for several hours with the consequence of different judging at different times. Other factors have to be considered such as the ergonomic condition forced by some environments. For instance, in the cargo area, inspection workers have to work in a crouched position given the low ceiling. These and other reasons justify the need for automation in the control quality process for aircraft interiors. On the other hand, some tasks, such as judging the appearance of panels, with all nuances intrinsic in this task, suggest that keeping the human in the loop is important because it allows joining the precision and accuracy of robotic inspection systems with the supervision of highly skilled inspection officers.

This section will analyze an automatic inspection of aircraft interiors panel assembly leveraging both human and robotic skills. This work is part of the "Vision-based Inspection Systems for automated Testing of Aircraft interiors" (VISTA) project which required the development of an autonomous robot for scanning cargo and cabin of aircraft and a method to process the 3D acquired data to detect defects. The geometrical defects detection approach was analyzed in [26, 146] while surface defects detection was proposed in [147]. In the following sections, an approach to facilitate the validation process of reported measurements and detected defects is presented. The approach exploits virtual and augmented reality to make the inspection officers' validation more efficient and effective.

AR and VR are promising technologies allowing to improve the efficiency and accuracy of several processes in which human work remains fundamental. These technologies can help human work by offering a higher level of immersion improving human-computer interaction. As explained in [148], these technologies joined with artificial intelligence represent a very powerful tool applicable in a wide range of fields.

Proposed solutions are developed to be suitable for different scenarios. In particular, the proposed approach provides two different solutions: the VR-based one which is more suitable for off-site applications while the AR-based one is

applicable to the on-site validation process.

6.2.1 Related Work

Recently several literature reviews about augmented reality in the manufacturing industry were proposed [149, 150]. In [151], papers published from 2006 to 2017 in this field are reviewed. As shown in that review, most of the performed research work is connected to the automotive industry with 11 papers, as opposed to just 3 papers dedicated to the aircraft/aerospace industry. This review gives a look at the industrial field in opposition to the AR application. It shows that two of these papers [152, 153] are related to the maintenance sector while one is related to assembly operations [154]. But there are no papers about post-assembly control quality.

In [155], Eschen et al., analyzed four use cases to use AR and VR for inspection and maintenance in the aerospace field. An AR approach to process guidance in inspecting and repairing scarf profiles of fiber composite parts through milling is presented. It uses an Hololens headset using an image marker to register real and virtual worlds.

An AR-assisted system for the inspection of aviation connectors has been proposed by Li et al. [156]. Here, the users leverage AR glasses to inspect connectors to detect mismatched or missing pins. It uses a client-server approach in which images are sent to a separate computer for processing and results are sent back as a list of misplaced-wire pins and missing-wire pins.

In summary, there are no previous works on AR/VR semi-automated quality control of aircraft interior lining.

6.2.2 Geometrical and Surface Defects

In this project, two categories of defects that may be present while inspecting aircraft interiors are considered: geometrical and surface defects.

Geometrical defects are related to the panels' arrangement and to deformities on the edges of single panels. Figure 6.5 illustrates geometrical defects considered

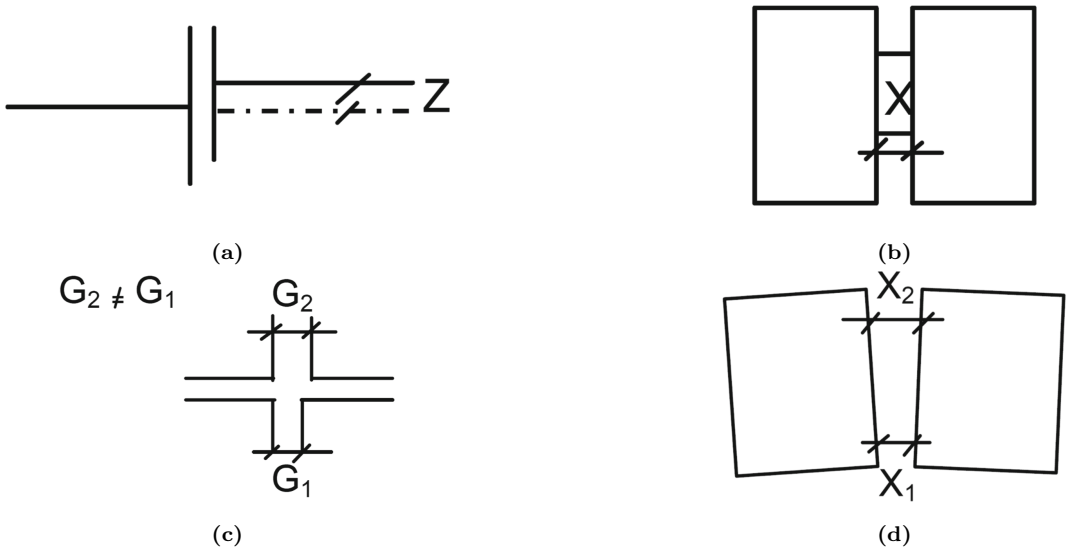


Figure 6.5: Geometrical defects: (a) step, (b) gap, (c) mismatch, (d) parallelism

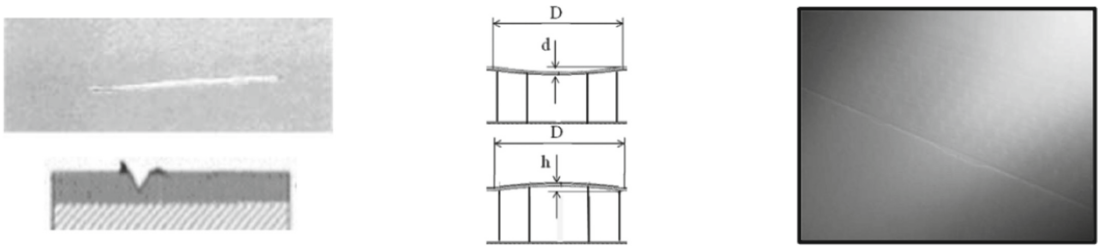
in the project, which are:

- **Step:** two adjacent panels, that should appear as continuous, have a misalignment on the z-plane exceeding a given threshold.
- **Gap:** it is the distance between the closest ends of adjacent panels that exceed a given tolerance.
- **Mismatch of tolerances:** it occurs in the proximity of multiple panels (usually four) and it is related to different gaps between any adjacent pairs.
- **Parallelism:** it is related to the lack of parallelism between adjacent panels. it appears as gap differences at extreme ends of the shared edge.

Surface defects are related to a single panel and appear as colors and roughness changes on the surface. They are usually present before assembly. In this work, only three types of surface defects are considered (which are shown in Figure 6.6):

- **Scratches:** linear damage of the surface panel, usually given by contact with sharp objects.

- **Bumps/Dents:** local concavity/convexity, usually rounded, which could be caused by inappropriate assembly or impact with a non-sharp object causing a local panel deformation.
- **Color/Texture inhomogeneities:** a change in color or texture on an otherwise uniform panel part.



(a) (b) (c)

Figure 6.6: Surface defects: a) scratches, b) bumps/dents, c) color/texture inhomogeneities

Most of the presented defects must be corrected for safety, aesthetic, and comfort issues. However, it is important to keep a human inspector in the loop since, often, the aesthetic considerations can involve more than automated measurements for a correct judgment.

6.2.3 Reporting Sub-system

The proposed solution is based on a client-server architecture where the main system - called "supervisor" - handles the interactions with all the sub-systems distributed among different devices. The supervisor has the aim to handle access to a centralized database containing the state of the automated inspections, including setup, configuration, acquired data, and processed data. The latter ones are accessed through reporting clients, with the supervisor acting as the "middleman" for data access.

It is possible to access the processing results through different means, including but not limited to:

- **Tablet APP:** reports are accessed through an Android or iOS application with 3D representation of the aircraft interiors offered as CAD or scanned data.
- **Windows APP:** reports are accessed through a Universal Windows Platform (UWP) on Windows 10+.
- **Hololens APP:** reports are accessed through a UWP application developed for Hololens 1, with the possibility of overlay measurements of specific defects.

It is important to highlight that, notwithstanding the project is focused on the development of a Virtual Reality application using a mobile or desktop application, it is preferred not to use specific VR devices as VR headsets. This is because the immersion offered by these devices was perceived more akin to a "disconnection" from the physical environment in the specific application. On the other hand, providing contextual information in the real environment was perceived as useful for the on-site scenario, leading to the development of the Mixed Reality solution.

All developed applications work in a similar way, but some differences are present to leverage the different technologies' features.

For instance, the mobile application exploits the inspection environment for the camera setup, in order that the user can closely observe the panels to inspect. Exploiting the landscape mode, the application can present a list of defects on a left pane while, on the right pane, the 3D environment with the selected defect properly highlighted. Users can move between different defects and move in the virtual environment as well.

The selection of the inspection area (cargo or cabin) is handled differently in the AR application. The Hololens app has to overlay synthetic information on the real environment. For this aim, an image marker is used. This permits the alignment of the real and virtual worlds for tracking initialization. If a 3D scan of the scene is available, it can be shown in the Hololens view as well. A set of graphical panes permits to modify color and transparency of the shown synthetic data to improve the user experience.

In all reporting clients, the user can express a judgment on each defect, validating the results and potentially improving classification results in subsequent inspections. The system is designed to be multi-user so that many opinions can be considered. The communication between the server and clients is based on a REST API using the JSON format for message exchange.

All the applications, developed for different devices, have been developed in Unity to enable 3D environment visualization. However, Hololens app required a more customized solution since the device nature. Indeed, the user interface had to manage head-mounted display gestures and gaze. Moreover, it was important to register the virtual representation of the scene with the real one for achieving data overlay over panels, exploiting Hololens sensors for interpreting the scene and tracking object positions.

Given the structured environment in which the proposed solution will be used, the solution based on the image marker has been considered advantageous. Thus, two different markers, one for the cargo and one for the cabin, were used at a known distance from the inspected panels. Once the marker is identified, the Hololens can track user pose leveraging its inertial sensors, depth, and color camera even with the marker out of sight. A commercial off-the-shelf solution, Vuforia Engine [157], which can be integrated into Unity, was used for handling the AR scenario, while Vuforia Studio was exploited for creating the user interface based on the floating window. Three ways are available to list the defects: as text; on a 3d model; or over-imposed in augmented reality.

6.2.4 Results

The focus of this section is on facilitating reporting of quality control issues for interior lining in single-aisle aircraft. The project includes the inspection of lining and panels after mounting sidewalls and hatracks, but before installing airplane seats, as Figures 6.7 and 6.8 of VR-like and AR prototypes show.

Reporting is weakly coupled with the inspection phase: it enables to visualize inspection results and to express a judgment on them to permit officers to choose what is correctly considered a defect and what was misclassified.

Figure 6.7: Mobile devices interface. The left pane shows the list of defects and measurements and on the right the 3D representation of the scene

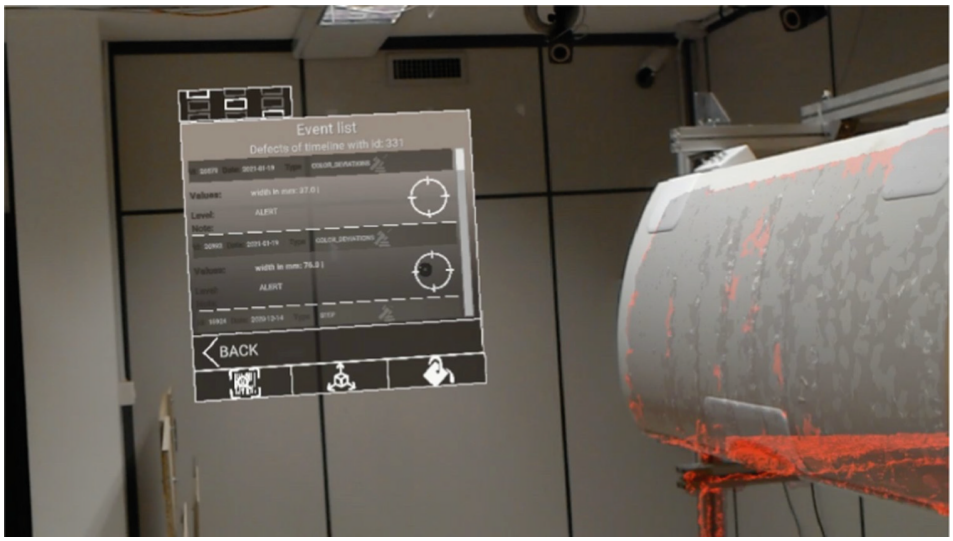


Figure 6.8: AR interface. Floating window reporting the list of measurements in the event list in the Hololens application

All the developed applications provide a list of defects offered as a list view reported after the user has selected a processing timeline to inspect. The user can also enter into a 3D view mode that splits the screen into two panes showing on the left the defects list and on the right the VR environment representation (as shown in Figure 6.7). This VR-based application has been implemented for analyzing the measurements when the user is away from the production site.

A mixed reality application was instead developed for on-site situations. This leverage mixed reality devices, such as Microsoft Hololens, allowing the user to freely move in the area to inspect and see in an augmented reality setup the position of the possible defects over imposed on the real environment. Using the application on Hololens, it is possible to visualize all information on the same scene, projecting virtual objects floating in the space at a specified distance. As shown in Figure 6.8, a floating window shows measurement data, while leaving most of the view "free" for showing virtual markers and objects overlaid on the real environment and panels.

6.2.5 Summary

Here a novel approach based on mixed reality was proposed for enabling the inspection officers to work better while analyzing geometrical and surface defects in aircraft interiors. This is part of a bigger project that aims to improve the aircraft lining assembly process using an autonomous platform to automatically perform measurements and detect defects. The proposed approach enables inspection officers to validate them in both on-site and off-site scenarios allowing, moreover, officers to avoid non-ergonomic work positions.

In the future, there are many points that can be investigated. It will be interesting to experimentally quantify how the proposed approach can improve working situations from both quality of life and productivity points of view. Furthermore, the user interface can be improved after an experimental study of user experiences. Finally, it can be useful to explore possible improvements in developing the AR application on the new generation Microsoft Hololens 2.

6.3 Optical Encoder for Robot Localization

Mobile robots are fundamental to accomplish several tasks in the manufacturing industry. They are particularly useful to automatically move objects that can also be really weighty. Mobile robots can automatize, for instance, warehouse management [158] - which can be an heavy work for humans - making this process really efficient and accurate. Obviously, robots, to autonomously coexist and navigate an environment in a safe way, need accurate and reliable localization.

This section tackles the robot localization problem by deep learning means. Accurate localization is fundamental for mobile robots since it is at the base of most mobile robot applications. Autonomous navigation, which is one of the most challenging requirements for mobile robots [159, 160], needs a reliable localization.

In literature, many solutions for robot localization have been proposed, using different strategies [161] and sensors [162]. Localization methods are usually classified into two main categories: absolute and relative localization. In the farthest, the robot pose is estimated with respect to a global reference system, while, in the latest, the robot pose is derived by knowing its initial pose, and accumulating its relative displacement over time. Here, an innovative relative localization approach is discussed.

6.3.1 Related Work

In the last few years, advances in computer vision have favorite the emergence of new localization methods for robot navigation and localization [163, 164, 165]. One of the most popular computer vision methods for relative robot localization is, without any doubt, visual odometry [166, 167, 168]. It leverages one or multiple cameras to estimate robot movements.

Given the deep learning success in all computer vision tasks, recently some deep learning approaches for visual odometry have been proposed [169, 170]. In [171], a CNN was proposed with the aim of predicting velocity and direction changes from stereo images. The synchrony detection method aids in detecting visual motion and depth representations, then the CNN can correlate these local depths and

motion representations to indirectly perform visual odometry. In 2017, Muller at al. proposed Flowodometry [172], which developed three different network architectures for robust ego-motion estimation. They are able to extract new visual features starting from a dense optical flow as input. Moreover, robustness against blur, luminance, and contrast anomalies is proven. A similar approach was proposed in [173], where a CNN was used to estimate the rotation and displacement information between frames from optical flow data. Posenet [174], uses a CNN to estimate, in real-time, the 6-DOF camera pose in both indoor and outdoor environments.

Recurrent Neural Networks (RNN), were also used for visual odometry [175]. They allow both to learn an effective feature representation for visual odometry applications and to model sequential relations and dynamics implicitly. In [176] an unsupervised learning approach was proposed. Moreover, it uses additional depth information to reach better results since classic monocular odometry needs prior information about the absolute scale. [177] uses a combination of CNN and RNN to retrieve the monocular camera ego-motion from optical flow data. This method can learn the scale without having knowledge about the intrinsic camera parameters. In [178] an unsupervised approach for visual odometry using a downward-facing camera is proposed. Nevertheless, the used loss function does not permit to estimate rotations with good accuracy.

Analyzing the literature, it is possible to notice the important deep learning role for the visual odometry task. However, in some cases, the networks need to be retrained or fine-tuned to be used in a different environment. In some cases, additional data or an image pre-processing step are required to correctly estimate the movements.

6.3.2 Proposed Approach

Here OE-Net - the proposed CNN framework for monocular visual odometry - is presented. It is inspired by FlowNetS [179]. The proposed CNN is able to extract visual features for the relative motion estimation of robot.

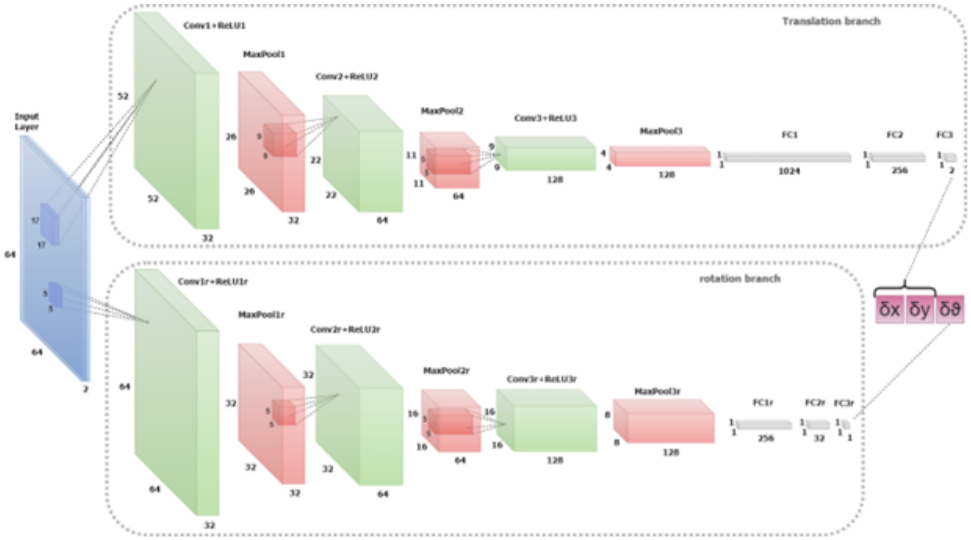


Figure 6.9: OE-Net architecture. It takes in input a tensor of two temporally following images and estimates the relative pose composed of translation and rotation. OE-Net is composed of two branches: the first one estimates the translational components, and the second one estimates the rotational one.

OE-Net Architecture

OE-Net is a CNN architecture proposed with the aim of performing monocular visual odometry tasks. It leverages images of a camera oriented toward the ground floor fastened on a robot.

OE-Net architecture is shown in Figure 6.9. It takes in input two temporally-consecutive images. It is composed of two branches: the first one estimates δx and δy, the planar translational information, while the second branch estimates the rotational information $\delta\vartheta$. Before it, a network with a single branch solving for both translation and rotation was developed. This solution was not able to provide satisfying results.

Table 6.1 reports all OE-Net's configuration parameters. The two branches work in parallel on the same input tensor. The translational branch is composed of three convolutional layers followed, each followed by a RELU. Their kernel size gradually decreases from 17x17 to 9x9 and then to 5x5. It was observed that using smaller kernels, such as 3x3, provides worse results in motion estimation.

Indeed, in the case of relatively large motions, the features to be correlated over the images might be scattered over different receptive fields of the convolutional layer, thus the accurate motion estimation could not be attained effectively. In effect, small kernels are less receptive than large kernels. When small kernels are used, large motions might be detected only by the inner layers of the network as the output tensor from previous layers decreases its spatial size by means of the pooling layers. The first layers could not be able to detect the motion, and part of the information could be lost. Therefore, 17x17 kernels having higher receptive fields than 3x3 kernels have been introduced in the first convolutional layer to ensure more accurate estimates.

The channels number of convolutional layers increases according to multiples of base 2. The zero-paddings and the strides are introduced to change the spatial dimension of activation tensors after the convolution step. All the max-pooling layers halve the spatial dimension of input tensors preserving the cardinality of channels. The output of the translation branch is a tensor of two elements containing the relative planar translation between the two input images.

The rotational branch, as the translational one, contains three convolutional layers followed by RELUs. The main difference is that here the kernel size is kept fixed to 5x5. It was experimentally proven that an higher kernel size does not provide significant improvements. This is because, in the experiments, a small receptive field is usually sufficient to capture relevant information about the rotation adequately. For this reason, since the different kernel sizes between the two branches enables a better estimation of the different motion components, the choice to use two sub-networks.

The concatenation layer allows merging the results of the two branches while the regression layer provides the loss error to backpropagate to train the network. As loss function, an MSE is considered. It is defined as:

$$MSE = \frac{1}{N}\sum_{k=1}^{n}(v_k^e - v_k^{gt})^2 \tag{6.2}$$

where N is the batch size, $v_k^e = [\delta x_k^e, \delta y_k^e, \delta \vartheta_k^e]^T$ is the estimate relative motion vector while $v_k^{gt} = [\delta x_k^{gt}, \delta y_k^{gt}, \delta \vartheta_k^{gt}]^T$ is the ground truth one.

	Layer name	Kernel size	Stride	Padding	Channels	Activation
	InputLayer	-	-	-	-	64x64x2
Translational Branch	Conv1+ReLU1	17x17	1	2	32	52x52x32
	MaxPool1	2x2	2	0	-	26x26x32
	Conv2+ReLU2	9x9	1	2	64	22x22x64
	MaxPool2	2x2	2	0	-	11x11x64
	Conv3+ReLU3	5x5	1	1	128	9x9x128
	MaxPool3	2x2	2	0	-	4x4x128
	FC1	-	-	-	1024	1x1x1024
	FC2	-	-	-	256	1x1x256
	FC3	-	-	-	2	1x1x2
Rotational Branch	Conv1r+ReLU1r	5x5	1	2	32	64x64x32
	MaxPool1r	2x2	2	0	-	32x32x32
	Conv2r+ReLU2r	5x5	1	2	64	32x32x64
	MaxPool2r	2x2	2	0	-	16x16x64
	Conv3r+ReLU3r	5x5	1	2	128	16x16x128
	MaxPool3r	2x2	2	0	-	8x8x128
	FC1r	-	-	-	256	1x1x256
	FC2r	-	-	-	32	1x1x32
	FC3r	-	-	-	1	1x1x1
	Concat(FC3+FC3r)	-	-	-	-	1x1x3
	RegressionLayer	-	-	-	-	1x1x3

Table 6.1: OE-Net configuration.

Experimental Setup

Here the strategies to collect the synthetic data and train the proposed network are discussed.

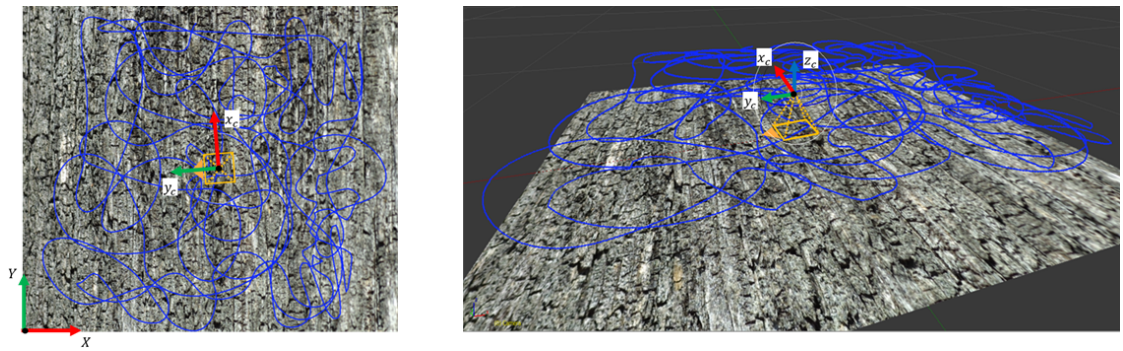
Dataset. Although the proposed OE-Net model will necessarily have to work in a real environment, preliminary experiments on synthetic data are done. Several advantages justify this choice. In particular, it is simpler to obtain ground truth data, and it permits to study how variations in illumination, speed of movements, floor materials, blurs, etc., affect the performance.

Synthetic data are generated using Blensor [180], an open-source simulation toolbox. It is based on Blender, a 3D modeling, animation, and rendering software. The Blensor package adds capabilities for exporting the simulated world as it can simulate various types of range scanners and color cameras. The data are collected by simulating a monocular camera moving in an environment with a texture on the floor, as shown in Figure 6.10. Floor images and ground truth motion are collected and used to train the model. The camera frame rate is set to 60 fps, which represents a good trade-off between motion accuracy and computational costs. The camera focal length f is equal to 3.7 mm, whereas the diagonal field of view (FOV) is $dFOV = 189.7$ mm by considering the distance of the camera from the floor of $h = 110$ mm. The collected dataset includes a total of 10k samples.

Training. For the OE-Net training, the collected dataset had been divided using the 80% of randomly chosen samples for the training and the remaining 20% was halved between validation and test sets. Stochastic gradient descent with momentum (SGDM) has been used as solver [181] with the momentum equal to 0.9. The batch size was 256 m with shuffle mode activated. The system was implemented using the Matlab framework [182] and trained on an NVIDIA GeForce RTX 3080 with 10 GB of memory.

6.3.3 Results

The results obtained performing the trajectory shown in Figure 6.10 are shown in Figure 6.11. It is clear that OE-net is able to accurately estimate the right



(a) **(b)**

Figure 6.10: Top view (a) and side view (b) of the simulated trajectory (in blue) that the camera (in orange) follows in the Blensor framework. The images and the ground truth of motion are collected during the simulation.

trajectory. On the test set, the RMSE was calculated to evaluate the system performance. It was $6.49 \cdot 10^{-2}$ mm and $6.03 \cdot 10^{-}2$ mm for δx and δy, respectively. While for the orientation $\delta\vartheta$ is $0.51°$ These results suggest that OE-Net is very accurate in translational estimation, but it makes more errors in estimating the rotational component.

Figure 6.11 shows the comparison of the OE-Net estimated trajectory compared with the ground truth one. As can be noticed, the two trajectories are very similar. Obviously, since OE-Net estimates the relative pose between two time instants, it suffers from cumulative errors. However, the two signals seem nearly equivalent. This suggests that the rotational error involves this divergence. To demonstrate this, the OE-Net projected trajectory using the ground truth rotational values is also plotted in Figure 6.12. This trajectory perfectly overlaps the ground truth. The trajectory total length is equal to 28.5783 m while the OE-Net estimated one is 28.5741 m with an absolute difference of $4.19 \cdot 10^{-}3$ m. The final absolute positioning error is equal to $3.03 \cdot 10^{-}2$ m.

Finally, in light of these results, it is possible to affirm that OE-Net is an accurate and reliable system for visual odometry tasks.

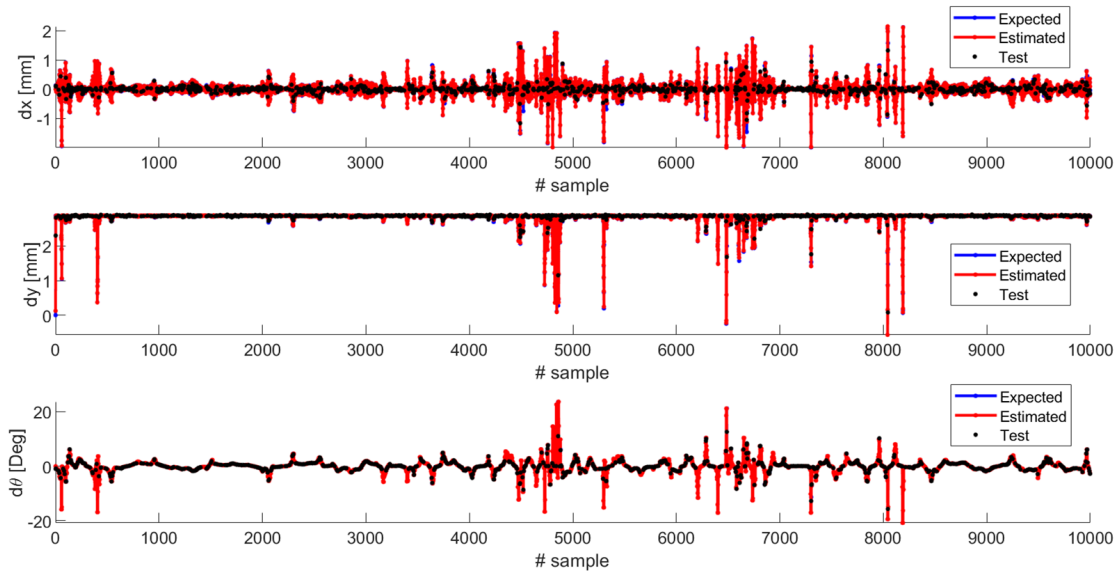


Figure 6.11: Relative pose estimates (in red) compared with the expected estimates (in blue) for all the samples. The black dots indicate the samples belonging to the testing set.

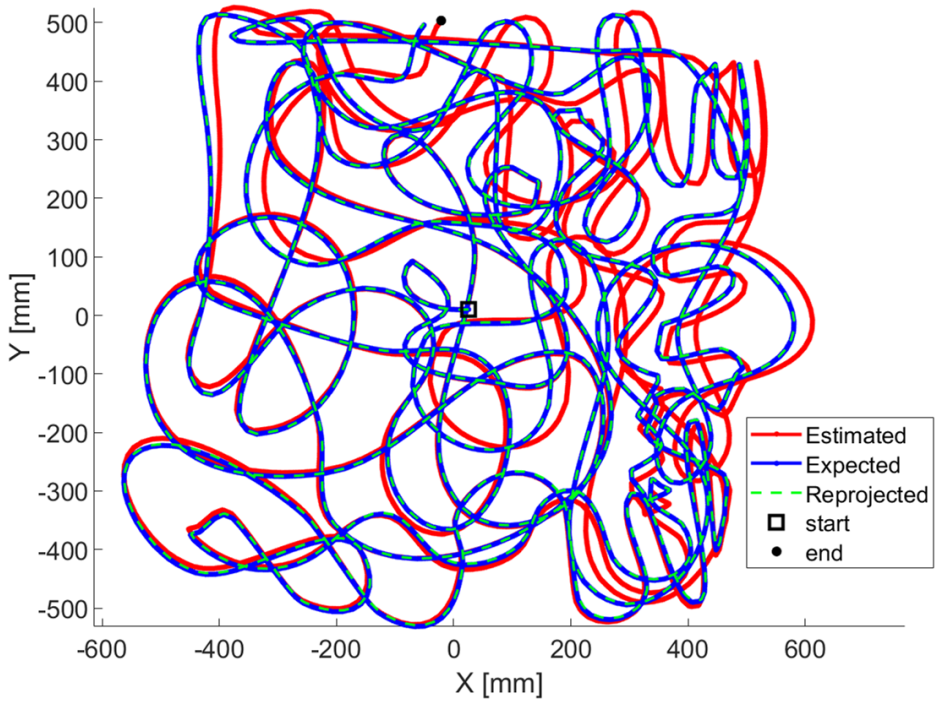


Figure 6.12: Estimated trajectory (in red) obtained by cumulating all the relative poses computed by the OE-net, compared with the expected trajectory (in blue). The green dashed signal represents the reprojection of OE-net measurements by using the expected orientations.

6.3.4 Summary

In this section, an innovative deep learning approach to visual odometry tasks has been presented. OE-Net is able to estimate the relative planar camera movement by analyzing two consecutive floor images. It is straightforward that by combining these relative movements, it is possible to localize the camera. The proposed network is composed of two branches that respectively estimate translational and rotational movements in parallel. A synthetic dataset was created to train the model. Both qualitative and quantitative results confirm the effectiveness of the proposed method.

Future work will regard the improvement of the rotational component estimation by investigating new network configurations and architectures. Other important aspects, such as variations in illumination, speed of movements, defocus effects, and camera parameters will also be considered and evaluated using synthetic data. Finally, real environment images will also be considered to validate the OE-Net in real applications.

Chapter 7

Conclusions and Future Work

In this thesis, the problem of the reconstruction and analysis of 3D models was analyzed in two different domains as autonomous vehicles and the manufacturing industry. Several tasks are analyzed and faced up in the two domains, aiming to provide new efficient and reliable methods as well as to improve of those already present in the literature to face up the identified problems.

The work has been focused on the acquisition, processing, and exploitation of both 2D and 3D data to extract important information useful for each analyzed task. In particular, in both domains, new approaches based on both 2D and 3D data are proposed to address different problems typical of the specific domain.

In detail, first in the autonomous driving domain has been analyzed setting two main objectives:

- Design and development of a deep learning car tracking algorithm exploiting 3D data.
- Design and development of a deep learning approach to the monocular depth estimation problem.

While, in the manufacturing industry domain, three main objectives were fixed aiming to automate and facilitate as many industrial processes. In detail, the following objectives were pursued:

- Design and development of a robot sensor calibration and point cloud stitching method to automate control quality process in aerospace industries.
- Design and development of AR/VR based applications to facilitate the aircraft interiors control quality keeping humans in the loop.
- Design and development of OE-net, a CNN model to address the visual odometry problem for mobile robot localization.

Chapter 5 analyzes the autonomous driving domain with its open problems that take advantage of computer vision. Here, two main problems are taken into account proposing deep learning based approaches for both of them.

First, the single car tracking (SOT) problem was analyzed. It was noticed the error drift over time problem typical of tracking systems. An approach to mitigate it by adding a detection branch to that pipeline with the aim of resetting the tracking branch with a given frequency was proposed. Both tracking and detection branches consist of models previously proposed in the literature. The method has been validated on a standard dataset and results confirm the effectiveness of the method. Some insights on how to choose the reset frequency are given. Future work will regard the investigation of methods to choose the best frames to execute detection and the exploitation of different types of data/sensors to leverage their different peculiarities.

Secondly, the monocular depth estimation task was considered. An efficient CNN model to estimate depth from a single view image with a knowledge distillation approach was proposed. Here the aim was to develop a model able to perform this task in systems with limited resources. Experiments tested the model from both performance and efficiency points of view analyzing the latest on several hardware architectures. Results confirmed the effectiveness of the approach since the trained model reached performances comparable with the baseline. Future work will regard the development of more advanced knowledge distillation pipelines to facilitate the training and the design of a new customized deep learning model able to run in real-time also with limited computational resources.

Chapter 6 inspects the manufacturing industry domain identifying several processes that can be automated by leveraging emerging technologies.

Here, first, a new approach to quality control of specific aircraft components is proposed. The approach uses a robotic platform composed of a mobile automated guided vehicle (AGV), a robotic arm, and a 3D sensor to scan components and stitch resulting point clouds. The aim is to measurements of possible gaps in the assembly of an aircraft horizontal stabilizer. A preliminary robot-sensor calibration step is required. Then, several workplans to scan each different component have been defined. Once the acquisitions are completed, single scans point clouds are stitched by moving each of them in a global reference frame. Results confirm the effectiveness of the proposed approach that, in the specific scenario, results more reliable than the baseline. In the future can be useful to investigate the calibration problem more.

Then, a semi-automatic approach for quality control of aircraft interiors is proposed. It leverages AR/VR technologies to facilitate the inspector officers' defects validation work. Different AR and VR based applications are proposed to account for both on-site and off-site scenarios. Future work provides the experimental quantification of how the proposed approach can improve working situations from both quality of life and productivity points of view, and the improvement of AR applications leveraging new generation helmets.

Finally, a deep learning approach to visual odometry based localization of autonomous robots is proposed. Accurate localization is fundamental for mobile robots since it is at the base of most mobile robot applications. OE-Net estimates the relative planar camera movement by analyzing two consecutive floor images. A synthetic dataset was created to train the model. Experimental results confirm the effectiveness of the method. Future work regards the investigation of new architectures to improve performances and the validation of the model on real-world applications.

Bibliography

[1] Mohamed Elgendy. *Deep learning for vision systems*. Simon and Schuster, 2020.

[2] Chris Harris, Mike Stephens, et al. "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[3] Hans Moravec. "Towards Automatic Visual Obstacle Avoidance". In: *Proceedings of 5th International Joint Conference on Artificial Intelligence (IJCAI '77)*. Vol. 1. Aug. 1977, p. 584.

[4] David G Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[5] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[7] Edward Rosten and Tom Drummond. "Fusing points and lines for high performance tracking". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. Ieee. 2005, pp. 1508–1515.

[8] Jack E Bresenham. "Algorithm for computer control of a digital plotter". In: *IBM Systems journal* 4.1 (1965), pp. 25–30.

[9] Michael Calonder et al. “Brief: Binary robust independent elementary features”. In: *European conference on computer vision*. Springer. 2010, pp. 778–792.

[10] Martin Weinmann. “Visual features—From early concepts to modern computer vision”. In: *Advanced topics in computer vision*. Springer, 2013, pp. 1–34.

[11] Kunihiko Fukushima and Sei Miyake. “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[12] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[13] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[14] Albert Palomer, Pere Ridao, and David Ribas. “Inspection of an underwater structure using point-cloud SLAM with an AUV and a laser scanner”. In: *Journal of field robotics* 36.8 (2019), pp. 1333–1344.

[15] Waqas Ali et al. “A Feature Based Laser SLAM Using Rasterized Images of 3D Point Cloud”. In: *IEEE Sensors Journal* 21.21 (2021), pp. 24422–24430.

[16] Ji Zhang and Sanjiv Singh. “LOAM: Lidar odometry and mapping in real-time.” In: *Robotics: Science and Systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.

[17] Lei Yang et al. “A novel system for off-line 3D seam extraction and path planning based on point cloud segmentation for arc welding robot”. In: *Robotics and Computer-Integrated Manufacturing* 64 (2020), p. 101929.

[18] Iago Z Biundini et al. “A framework for coverage path planning optimization based on point cloud for structural inspection”. In: *Sensors* 21.2 (2021), p. 570.

[19] Xiangfei Wang et al. “Point cloud 3D parent surface reconstruction and weld seam feature extraction for robotic grinding path planning”. In: *The International Journal of Advanced Manufacturing Technology* 107.1 (2020), pp. 827–841.

[20] Knut B Kaldestad et al. “Collision avoidance with potential fields based on parallel processing of 3D-point cloud data on the GPU”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 3250–3257.

[21] Haeyeon Gim, Minwook Jeong, and Soohee Han. “Autonomous Navigation System with Obstacle Avoidance using 2.5 D Map Generated by Point Cloud”. In: *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. IEEE. 2021, pp. 749–752.

[22] Enrique Aldao et al. “UAV Obstacle Avoidance Algorithm to Navigate in Dynamic Building Environments”. In: *Drones* 6.1 (2022), p. 16.

[23] You Li and Javier Ibanez-Guzman. “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems”. In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61.

[24] Georgios Zamanakos et al. “A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving”. In: *Computers & Graphics* 99 (2021), pp. 153–181.

[25] Delin Huang et al. “Detection and monitoring of defects on three-dimensional curved surfaces based on high-density point cloud data”. In: *Precision Engineering* 53 (2018), pp. 79–95.

[26] Nicola Mosca et al. “Post assembly quality inspection using multimodal sensing in aircraft manufacturing”. In: *Multimodal Sensing and Artificial Intelligence: Technologies and Applications II*. Vol. 11785. SPIE. 2021, pp. 188–195.

[27] Takeshi Masuda. “Leaf area estimation by semantic segmentation of point cloud of tomato plants”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1381–1389.

[28] Thibault Clamens et al. “Real-time Multispectral Image Processing and Registration on 3D Point Cloud for Vineyard Analysis.” In: *VISIGRAPP (4: VISAPP)*. 2021, pp. 388–398.

[29] Xiaoyuan Luo et al. “Prediction of cerebral aneurysm rupture using a point cloud neural network”. In: *Journal of NeuroInterventional Surgery* (2022).

[30] Qi Cai and Yuan Feng. “Semantic Segmentation of Dental Point Cloud Based on Pointnet++”. In: *Proceedings of the International Conference on Information Economy, Data Modeling and Cloud Computing, ICIDC 2022, 17-19 June 2022, Qingdao, China*. 2022.

[31] Valeria Croce et al. “From the semantic point cloud to heritage-building information modeling: A semiautomatic approach exploiting machine learning”. In: *Remote Sensing* 13.3 (2021), p. 461.

[32] Roberto Pierdicca et al. “Point cloud semantic segmentation using a deep learning framework for cultural heritage”. In: *Remote Sensing* 12.6 (2020), p. 1005.

[33] Luke Weidner, Gabriel Walton, and Ryan Kromer. “Classification methods for point clouds in rock slope monitoring: A novel machine learning approach and comparative analysis”. In: *Engineering Geology* 263 (2019), p. 105326.

[34] Adrián J Riquelme, Antonio Abellán, and Roberto Tomás. “Discontinuity spacing analysis in rock masses using 3D point clouds”. In: *Engineering Geology* 195 (2015), pp. 185–195.

[35] Ivan Sipiran and Benjamin Bustos. “Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes”. In: *The Visual Computer* 27.11 (2011), pp. 963–976.

[36] Bastian Steder et al. “Point feature extraction on 3D range scans taking into account object boundaries”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 2601–2608.

[37] Yu Zhong. "Intrinsic shape signatures: A shape descriptor for 3d object recognition". In: *2009 IEEE 12th international conference on computer vision workshops, ICCV Workshops*. IEEE. 2009, pp. 689–696.

[38] Paula Stancelova, Elena Sikudova, and Zuzana Cernekova. "3D Feature detector-descriptor pair evaluation on point clouds". In: *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 590–594.

[39] Andrea Frome et al. "Recognizing objects in range data using regional point descriptors". In: *European conference on computer vision*. Springer. 2004, pp. 224–237.

[40] Serge Belongie, Jitendra Malik, and Jan Puzicha. "Shape matching and object recognition using shape contexts". In: *IEEE transactions on pattern analysis and machine intelligence* 24.4 (2002), pp. 509–522.

[41] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Unique shape context for 3D data description". In: *Proceedings of the ACM workshop on 3D object retrieval*. 2010, pp. 57–62.

[42] Radu Bogdan Rusu et al. "Aligning point cloud views using persistent feature histograms". In: *2008 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2008, pp. 3384–3391.

[43] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. "Fast point feature histograms (FPFH) for 3D registration". In: *2009 IEEE international conference on robotics and automation*. IEEE. 2009, pp. 3212–3217.

[44] Samuele Salti, Federico Tombari, and Luigi Di Stefano. "SHOT: Unique signatures of histograms for surface and texture description". In: *Computer Vision and Image Understanding* 125 (2014), pp. 251–264.

[45] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[46] Anastasia Ioannidou et al. “Deep learning advances in computer vision with 3d data: A survey”. In: *ACM computing surveys (CSUR)* 50.2 (2017), pp. 1–38.

[47] E Ahmed et al. “Deep learning advances on different 3D data representations: A survey. arXiv 2018”. In: *arXiv preprint arXiv:1808.01462* ().

[48] Weiping Liu et al. “Deep learning on point clouds and its application: A survey”. In: *Sensors* 19.19 (2019), p. 4188.

[49] Nan Li, Olaf Kähler, and Norbert Pfeifer. “A comparison of deep learning methods for airborne lidar point clouds classification”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2021), pp. 6467–6486.

[50] Yulan Guo et al. “Deep learning for 3d point clouds: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.12 (2020), pp. 4338–4364.

[51] Hang Su et al. “Multi-view convolutional neural networks for 3d shape recognition”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 945–953.

[52] 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*. 2018, pp. 186–194.

[53] Charles R Qi et al. “Volumetric and multi-view cnns for object classification on 3d data”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5648–5656.

[54] Yifan Feng et al. “Gvcnn: Group-view convolutional neural networks for 3d shape recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 264–272.

[55] Felix Järemo Lawin et al. “Deep projective 3D semantic segmentation”. In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2017, pp. 95–107.

[56] 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)*. IEEE. 2015, pp. 922–928.

[57] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. "Octnet: Learning deep 3d representations at high resolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3577–3586.

[58] Truc Le and Ye Duan. "Pointgrid: A deep network for 3d shape understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9204–9214.

[59] Mor Joseph-Rivlin, Alon Zvirin, and Ron Kimmel. "Momen (e) t: Flavor the moments in learning to classify shapes". In: *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 2019, pp. 0–0.

[60] Jiancheng Yang et al. "Modeling point clouds with self-attention and gumbel subset sampling". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3323–3332.

[61] Hengshuang Zhao et al. "Pointweb: Enhancing local neighborhood features for point cloud processing". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5565–5573.

[62] Charles Ruizhongtai Qi et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems* 30 (2017).

[63] Yongcheng Liu et al. "Relation-shape convolutional neural network for point cloud analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8895–8904.

[64] Alexandre Boulch. "ConvPoint: Continuous convolutions for point cloud processing". In: *Computers & Graphics* 88 (2020), pp. 24–34.

[65] Wenxuan Wu, Zhongang Qi, and Li Fuxin. "Pointconv: Deep convolutional networks on 3d point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9621–9630.

[66] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. "Pointwise convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 984–993.

[67] Artem Komarichev, Zichun Zhong, and Jing Hua. "A-cnn: Annularly convolutional neural networks on point clouds". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 7421–7430.

[68] Sudhakar Kumawat and Shanmuganathan Raman. "Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4903–4912.

[69] Martin Simonovsky and Nikos Komodakis. "Dynamic edge-conditioned filters in convolutional neural networks on graphs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3693–3702.

[70] Yue Wang et al. "Dynamic graph cnn for learning on point clouds". In: *Acm Transactions On Graphics (tog)* 38.5 (2019), pp. 1–12.

[71] Chao Chen et al. "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4994–5002.

[72] Gusi Te et al. "Rgcnn: Regularized graph cnn for point cloud segmentation". In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 746–754.

[73] Yifan Feng et al. "Hypergraph neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 3558–3565.

[74] Chu Wang, Babak Samari, and Kaleem Siddiqi. "Local spectral graph convolution for point set feature learning". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 52–66.

[75] Peng-Shuai Wang et al. "O-cnn: Octree-based convolutional neural networks for 3d shape analysis". In: *ACM Transactions On Graphics (TOG)* 36.4 (2017), pp. 1–11.

[76] 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*. 2018, pp. 9397–9406.

[77] Yashar Deldjoo et al. "Towards Improving Car Point-Cloud Tracking Via Detection Updates". In: *2021 3rd International Conference on Image Processing and Machine Vision (IPMV)*. 2021, pp. 30–34.

[78] Yashar Deldjoo et al. "Towards real-time monocular depth estimation for mobile systems". In: *Multimodal Sensing and Artificial Intelligence: Technologies and Applications II*. Vol. 11785. SPIE. 2021, pp. 117–125.

[79] Xiaozhi Chen et al. "Multi-view 3d object detection network for autonomous driving". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 1907–1915.

[80] Charles R Qi et al. "Frustum pointnets for 3d object detection from rgb-d data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 918–927.

[81] Charles R Qi et al. "Deep hough voting for 3d object detection in point clouds". In: *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9277–9286.

[82] Wenjie Luo, Bin Yang, and Raquel Urtasun. "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 3569–3577.

[83] Martin Simon et al. “Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[84] Martin Simony et al. “Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[85] Jun Sang et al. “An improved YOLOv2 for vehicle detection”. In: *Sensors* 18.12 (2018), p. 4272.

[86] Haozhe Qi et al. “P2b: Point-to-box network for 3d object tracking in point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6329–6338.

[87] Hsu-kuang Chiu et al. “Probabilistic 3d multi-modal, multi-object tracking for autonomous driving”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 14227–14233.

[88] Zheng Fang et al. “3d-siamrpn: An end-to-end learning method for real-time 3d single object tracking using raw point cloud”. In: *IEEE Sensors Journal* 21.4 (2020), pp. 4995–5011.

[89] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. “Leveraging shape completion for 3d siamese tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1359–1368.

[90] Shaoshuai Shi et al. “Pv-rcnn: Point-voxel feature set abstraction for 3d object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10529–10538.

[91] Andreas Geiger et al. “Vision meets robotics: The kitti dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[92] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Online object tracking: A benchmark". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 2411–2418.

[93] Ian P Howard. *Perceiving in depth, volume 1: basic mechanisms*. Oxford University Press, 2012.

[94] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network". In: *Advances in neural information processing systems* 27 (2014).

[95] Fayao Liu et al. "Learning depth from single monocular images using deep convolutional neural fields". In: *IEEE transactions on pattern analysis and machine intelligence* 38.10 (2015), pp. 2024–2039.

[96] Daniel Scharstein and Richard Szeliski. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". In: *International journal of computer vision* 47.1 (2002), pp. 7–42.

[97] Yasutaka Furukawa, Carlos Hernández, et al. "Multi-view stereo: A tutorial". In: *Foundations and Trends® in Computer Graphics and Vision* 9.1-2 (2015), pp. 1–148.

[98] Rene Ranftl et al. "Dense monocular depth estimation in complex dynamic scenes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4058–4066.

[99] Robert J Woodham. "Photometric method for determining surface orientation from multiple images". In: *Optical engineering* 19.1 (1980), pp. 139–144.

[100] Austin Abrams, Christopher Hawley, and Robert Pless. "Heliometric stereo: Shape from sun position". In: *European conference on computer vision*. Springer. 2012, pp. 357–370.

[101] Ashutosh Saxena, Min Sun, and Andrew Y Ng. "Make3d: Learning 3d scene structure from a single still image". In: *IEEE transactions on pattern analysis and machine intelligence* 31.5 (2008), pp. 824–840.

[102] Lubor Ladicky, Jianbo Shi, and Marc Pollefeys. "Pulling things out of perspective". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 89–96.

[103] Kevin Karsch, Ce Liu, and Sing Bing Kang. "Depth transfer: Depth extraction from video using non-parametric sampling". In: *IEEE transactions on pattern analysis and machine intelligence* 36.11 (2014), pp. 2144–2158.

[104] Junyuan Xie, Ross Girshick, and Ali Farhadi. "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks". In: *European conference on computer vision*. Springer. 2016, pp. 842–857.

[105] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. "Efficient deep learning for stereo matching". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5695–5703.

[106] Arun CS Kumar, Suchendra M Bhandarkar, and Mukta Prasad. "Depthnet: A recurrent neural network architecture for monocular depth prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 283–291.

[107] Amir Atapour-Abarghouei and Toby P Breckon. "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2800–2810.

[108] Benjamin Ummenhofer et al. "Demon: Depth and motion network for learning monocular stereo". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5038–5047.

[109] Yuhua Chen et al. "Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1841–1850.

[110] Siyuan Qiao et al. "Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3997–4008.

[111] John Flynn et al. "Deepstereo: Learning to predict new views from the world's imagery". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5515–5524.

[112] Huangying Zhan et al. "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 340–349.

[113] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. "Unsupervised monocular depth estimation with left-right consistency". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 270–279.

[114] Filippo Aleotti et al. "Generative adversarial networks for unsupervised monocular depth prediction". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[115] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. "Learning monocular depth estimation with unsupervised trinocular assumptions". In: *2018 International conference on 3d vision (3DV)*. IEEE. 2018, pp. 324–333.

[116] Tinghui Zhou et al. "Unsupervised learning of depth and ego-motion from video". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1851–1858.

[117] Reza Mahjourian, Martin Wicke, and Anelia Angelova. "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5667–5675.

[118] Clément Godard et al. “Digging into self-supervised monocular depth estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3828–3838.

[119] Jianping Gou et al. “Knowledge distillation: A survey”. In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819.

[120] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. “Spatial transformer networks”. In: *Advances in neural information processing systems* 28 (2015).

[121] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[122] Chaoyang Wang et al. “Learning depth from monocular videos using direct methods”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2022–2030.

[123] Liang-Chieh Chen Yukun Zhu George, Papandreou Florian Schroff, and Hartwig Adam. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *ECCV. Liang-Chieh Chen Yukun Zhu George Papandreou Florian Schroff and Hartwig Adam* (2018).

[124] L-CCGP Florian and Schroff Hartwig Adam. “Rethinking atrous convolution for semantic image segmentation”. In: *Conference on computer vision and pattern recognition (CVPR). IEEE/CVF*. Vol. 6. 2017.

[125] Moritz Menze, Christian Heipke, and Andreas Geiger. “Joint 3d estimation of vehicles and scene flow”. In: *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences* 2 (2015), p. 427.

[126] *Pytorch Mobile*. `https://pytorch.org/mobile/home/`. (Accessed: 26 October 2022).

[127] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[128] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[129] Pierluigi Dibari et al. “An accurate hardware calibration and 3D point cloud stitching towards automatic quality control in aerospace manufacturing”. In: *2022 IEEE 9th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE. 2022, pp. 265–269.

[130] Nicola Mosca et al. “Virtual and Augmented Reality for Quality Control of Aircraft Interiors”. In: *International Conference on Image Analysis and Processing*. Springer. 2022, pp. 225–234.

[131] Cosimo Patruno et al. “Optical encoder neural network: a CNN-based optical encoder for robot localization”. In: *Optical Engineering* 62.3 (2022), p. 031204.

[132] Song Zhang. “High-speed 3D shape measurement with structured light methods: A review”. In: *Optics and Lasers in Engineering* 106 (2018), pp. 119–131.

[133] Rui Lv et al. “WPMAVM: Weighted plus-and-minus allowance variance minimization algorithm for solving matching distortion”. In: *Robotics and Computer-Integrated Manufacturing* 76 (2022), p. 102320.

[134] Alessandro Paoli and Armando V Razionale. “Large yacht hull measurement by integrating optical scanning with mechanical tracking-based methodologies”. In: *Robotics and Computer-Integrated Manufacturing* 28.5 (2012), pp. 592–601.

[135] Jinshan Wang et al. “A mobile robotic measurement system for large-scale complex components based on optical scanning and visual tracking”. In: *Robotics and Computer-Integrated Manufacturing* 67 (2021), p. 102010.

[136] S Ravishankar, HNV Dutt, and B Gurumoorthy. “AIWIN—a fast and jigless inspection technique for machined parts”. In: *The International Journal of Advanced Manufacturing Technology* 62.1 (2012), pp. 231–240.

[137] Cosimo Patruno et al. “An effective approach for 3D point cloud registration in railway contexts”. In: *Multimodal Sensing: Technologies and Applications*. Vol. 11059. SPIE. 2019, pp. 248–263.

[138] François Pomerleau, Francis Colas, Roland Siegwart, et al. “A review of point cloud registration algorithms for mobile robotics”. In: *Foundations and Trends® in Robotics* 4.1 (2015), pp. 1–104.

[139] *UR10e*. `https://www.universal-robots.com/products/ur10-robot/`. (Accessed: 27 October 2022).

[140] *Gocator 3210*. `https://lmi3d.com/resource/gocator-3210-datasheet-large-field-view-3d-snapshot-sensor/`. (Accessed: 27 October 2022).

[141] David J Wales and Jonathan PK Doye. “Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms”. In: *The Journal of Physical Chemistry A* 101.28 (1997), pp. 5111–5116.

[142] Hans-Peter Kriegel et al. “LoOP: local outlier probabilities”. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009, pp. 1649–1652.

[143] Paul J Besl and Neil D McKay. “Method for registration of 3-D shapes”. In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606.

[144] Yang Chen and Gérard Medioni. “Object modelling by registration of multiple range images”. In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[145] Peng Li et al. “Evaluation of the ICP algorithm in 3D point cloud registration”. In: *IEEE Access* 8 (2020), pp. 68030–68048.

[146] Nicola Mosca et al. “A RANSAC-based method for detecting post-assembly defects in aircraft interiors”. In: *2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE. 2020, pp. 403–408.

[147] Nicola Mosca et al. “Qualitative comparison of methodologies for detecting surface defects in aircraft interiors”. In: *2021 IEEE 8th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE. 2021, pp. 215–220.

[148] Maria di Summa et al. “Extended reality and artificial intelligence: Synergic approaches in real world applications”. In: *Roadmapping Extended Reality: Fundamentals and Applications* (2022), pp. 183–192.

[149] Joel Murithi Runji, Yun-Ju Lee, and Chih-Hsing Chu. “Systematic Literature Review on Augmented Reality-Based Maintenance Applications in Manufacturing Centered on Operator Needs”. In: *International Journal of Precision Engineering and Manufacturing-Green Technology* (2022), pp. 1–19.

[150] Phuong Thao Ho et al. “Study of Augmented Reality Based Manufacturing for Further Integration of Quality Control 4.0: A Systematic Literature Review”. In: *Applied Sciences* 12.4 (2022), p. 1961.

[151] Eleonora Bottani and Giuseppe Vignali. “Augmented reality technology in the manufacturing industry: A review of the last decade”. In: *Iise Transactions* 51.3 (2019), pp. 284–310.

[152] Francesca De Crescenzio et al. “Augmented reality for aircraft maintenance training and operations support”. In: *IEEE Computer Graphics and Applications* 31.1 (2010), pp. 96–101.

[153] Piotr Golanski, Malgorzata Perz-Osowska, and Marek Szczekala. “A Demonstration Model of a Mobile Expert System with Augmented Reality User Interface Supporting M-28 Aircraft Maintenace”. In: *Journal of KONBIN* 31.1 (2014), p. 23.

[154] Sabine Webel et al. “An augmented reality training platform for assembly and maintenance skills”. In: *Robotics and autonomous systems* 61.4 (2013), pp. 398–403.

[155] Henrik Eschen et al. “Augmented and virtual reality for inspection and maintenance processes in the aviation industry”. In: *Procedia manufacturing* 19 (2018), pp. 156–163.

[156] Shufei Li, Pai Zheng, and Lianyu Zheng. “An AR-assisted deep learning-based approach for automatic inspection of aviation connectors”. In: *IEEE Transactions on Industrial Informatics* 17.3 (2020), pp. 1721–1731.

[157] *Vuforia enterprise augmented reality (AR) software*. `https://www.ptc.com/en/products/vuforia`. (Accessed: 02 November 2022).

[158] Michela Zaccaria et al. “Multi-Robot Multiple Camera People Detection and Tracking in Automated Warehouses”. In: *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*. IEEE. 2021, pp. 1–6.

[159] Mary B Alatise and Gerhard P Hancke. “A review on challenges of autonomous mobile robot and sensor fusion methods”. In: *IEEE Access* 8 (2020), pp. 39830–39846.

[160] Sean Campbell et al. “Where am I? localization techniques for mobile robots a Review”. In: *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. IEEE. 2020, pp. 43–47.

[161] Prabin Kumar Panigrahi and Sukant Kishoro Bisoy. “Localization strategies for autonomous mobile robots: A review”. In: *Journal of King Saud University-Computer and Information Sciences* (2021).

[162] Johann Borenstein, HR Everett, Liqiang Feng, et al. “Where am I? Sensors and methods for mobile robot positioning”. In: *University of Michigan* 119.120 (1996), p. 27.

[163] Luis Paya, Arturo Gil, and Oscar Reinoso. “A state-of-the-art review on mapping and localization of mobile robots using omnidirectional vision sensors”. In: *Journal of Sensors* 2017 (2017).

[164] Mohammad OA Aqel et al. “Review of visual odometry: types, approaches, challenges, and applications”. In: *SpringerPlus* 5.1 (2016), pp. 1–26.

[165] Yusra Alkendi, Lakmal Seneviratne, and Yahya Zweiri. "State of the art in vision-based localization techniques for autonomous navigation systems". In: *IEEE Access* 9 (2021), pp. 76847–76874.

[166] Davide Scaramuzza and Friedrich Fraundorfer. "Visual odometry [tutorial]". In: *IEEE robotics & automation magazine* 18.4 (2011), pp. 80–92.

[167] Friedrich Fraundorfer and Davide Scaramuzza. "Visual odometry: Part ii: Matching, robustness, optimization, and applications". In: *IEEE Robotics & Automation Magazine* 19.2 (2012), pp. 78–90.

[168] Cosimo Patruno et al. "A vision-based odometer for localization of omnidirectional indoor robots". In: *Sensors* 20.3 (2020), p. 875.

[169] Ke Wang et al. "Approaches challenges and applications for deep visual odometry toward to complicated and emerging areas". In: *IEEE Transactions on Cognitive and Developmental Systems* (2020).

[170] Hao Liu, Dan Dan Huang, and Zhen Ye Geng. "Visual Odometry Algorithm Based on Deep Learning". In: *2021 6th International Conference on Image, Vision and Computing (ICIVC)*. IEEE. 2021, pp. 322–327.

[171] Kishore Reddy Konda and Roland Memisevic. "Learning Visual Odometry with a Convolutional Network". In: *VISAPP*. 2015.

[172] Peter Muller and Andreas Savakis. "Flowdometry: An optical flow and deep learning based approach to visual odometry". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 624–631.

[173] Gabriele Costante et al. "Exploring representation learning with cnns for frame-to-frame ego-motion estimation". In: *IEEE robotics and automation letters* 1.1 (2015), pp. 18–25.

[174] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946.

[175] Sen Wang et al. “Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks”. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 2043–2050.

[176] Qiang Liu et al. “Unsupervised Deep Learning-Based RGB-D Visual Odometry”. In: *Applied Sciences* 10.16 (2020), p. 5426.

[177] Tejas Pandey et al. “Leveraging deep learning for visual odometry using optical flow”. In: *Sensors* 21.4 (2021), p. 1313.

[178] Maximilian Gilles and Sascha Ibrahimpasic. “Unsupervised deep learning based ego motion estimation with a downward facing camera”. In: *The Visual Computer* (2021), pp. 1–14.

[179] Alexey Dosovitskiy et al. “Flownet: Learning optical flow with convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.

[180] Michael Gschwandtner et al. “BlenSor: Blender sensor simulation toolbox”. In: *International Symposium on Visual Computing*. Springer. 2011, pp. 199–208.

[181] Ning Qian. “On the momentum term in gradient descent learning algorithms”. In: *Neural networks* 12.1 (1999), pp. 145–151.

[182] *Matlab*. `https://www.mathworks.com/products/matlab-online.html`. (Accessed: 31 October 2022).