

University of Passau



Faculty of Computer Science and Mathematics (FIM)
Chair of Distributed Information Systems

Computer Science

Bridging the Realism Gap for CAD-Based Visual Recognition

Benjamin Planche

A Dissertation Presented to
the Faculty of Computer Science and Mathematics of the University of Passau
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Natural Sciences

- 1. Reviewer* **Prof. Dr. Harald Kosch**
Chair of Distributed Information Systems
University of Passau
- 2. Reviewer* **Prof. Dr. Serge Miguet**
LIRIS Laboratory
Université Lumière Lyon 2

Passau – September 17, 2019

Benjamin Planche

Bridging the Realism Gap for CAD-Based Visual Recognition

Computer Science, September 17, 2019

Reviewers: Prof. Dr. Harald Kosch and Prof. Dr. Serge Miguet

University of Passau

Chair of Distributed Information Systems

Faculty of Computer Science and Mathematics (FIM)

Innstr. 33

94032 Passau

Abstract

Computer vision aims at developing algorithms to extract high-level information from images and videos. In the industry, for instance, such algorithms are applied to guide manufacturing robots, to visually monitor plants, or to assist human operators in recognizing specific components. Recent progress in computer vision has been dominated by deep artificial neural network, *i.e.*, machine learning methods simulating the way that information flows in our biological brains, and the way that our neural networks adapt and learn from experience. For these methods to learn how to accurately perform complex visual tasks, large amounts of annotated images are needed. Collecting and labeling such domain-relevant training datasets is, however, a tedious—sometimes impossible—task. Therefore, it has become common practice to leverage pre-available three-dimensional (3D) models instead, to generate synthetic images for the recognition algorithms to be trained on. However, methods optimized over synthetic data usually suffer a significant performance drop when applied to real target images. This is due to the realism gap, *i.e.*, the discrepancies between synthetic and real images (in terms of noise, clutter, *etc.*). In my work, three main directions were explored to bridge this gap.

First, an innovative end-to-end framework is proposed to render realistic depth images from 3D models, as a growing number of solutions (especially in the industry) are utilizing low-cost depth cameras (*e.g.*, *Microsoft Kinect* and *Intel RealSense*) for recognition tasks. Based on a thorough study of these devices and the different types of noise impairing them, the proposed framework simulates their inner mechanisms, comprehensively modeling vital factors such as sensor noise, material reflectance, surface geometry, *etc.* Able to simulate a wide panel of depth sensors and to quickly generate large datasets, this framework is used to train algorithms for various recognition tasks, consistently and significantly enhancing their performance compared to other state-of-the-art simulation tools.

In some cases, however, relevant 2D or 3D object representations to generate synthetic samples are not available. Considering this different case of data scarcity, a solution is then proposed to incrementally build a representation of visual scenes from partial observations. Provided observations are localized from one to another based on their content and registered in a global memory with spatial properties. Simultaneously, this memory can be queried to render novel views of the scene. Furthermore, unobserved regions can be hallucinated in memory, in consistence with previous observations, hallucinations, and global priors.

The efficacy of the proposed mnemonic and generative system, trainable end-to-end, is demonstrated on various 2D and 3D use-cases.

Finally, an advanced convolutional neural network pipeline is introduced, tackling the realism gap from a novel angle. While most methods addressing this problem focus on bringing synthetic samples—or the knowledge acquired from them—closer to the real target domain, the proposed solution performs the opposite process, mapping unseen target images into controlled synthetic domains. The pre-processed samples can then be handed to downstream recognition methods, themselves purely trained on similar synthetic data, to greatly improve their accuracy.

For each approach, a variety of qualitative and quantitative studies are detailed, providing successful comparisons to state-of-the-art methods. By proposing solutions to bridge the realism gap from either side, as well as a pipeline to improve the acquisition and generation of new visual content, this thesis provides a unique perspective on the challenges of data scarcity when building robust recognition systems.

Zusammenfassung (Abstract in German)

Die Computer Vision strebt an, Algorithmen zum Extrahieren hochwertiger Informationen von Bildern und Videos zu entwickeln. In der Industrie werden solche Algorithmen beispielsweise angewendet, um Fertigungsroboter zu steuern, um Betriebe visuell zu überwachen, oder um Mitarbeiter bei der Erkennung bestimmter Komponenten zu unterstützen. Die kürzlichen Fortschritte im Bereich Computer Vision wurden von tiefen künstlichen neuronalen Netzen dominiert. Diese Methoden des maschinellen Lernens (*Machine Learning*) simulieren die Art und Weise, in der die Information in unseren biologischen Gehirnen verarbeitet wird und in der unsere neuronale Netze sich anpassen und aus Erfahrung lernen. Damit diese Methoden zur genauen Ausführung komplexer visueller Aufgaben befähigt werden, müssen sie mit einer großen Anzahl von annotierten Bildern trainiert werden. Die Erhebung und Kennzeichnung entsprechender Trainingsdatensätze ist jedoch eine langwierige und manchmal sogar unmögliche Aufgabe. Deswegen ist es zur gängigen Praxis geworden, stattdessen die vorhandenen 3D-Modelle zur Generierung synthetischer Bilder einzusetzen, damit die Erkennungsalgorithmen mit Hilfe dieser Bilder trainiert werden. Allerdings, bei der Anwendung auf die realen Zielbilder, erleiden die Methoden, die durch synthetische Daten angepasst wurden, einen erheblichen Leistungsabfall. Dies geschieht aufgrund der Realismuskluft (*Realism Gap*), das heißt durch die Diskrepanzen zwischen synthetischen und realen Bildern (hinsichtlich von Rauschen, Störungen *usw.*). In meiner Arbeit wurden drei Hauptrichtungen untersucht, um diese Lücke zu schließen.

Zuerst wird ein innovatives End-to-End-Framework vorgeschlagen, um realistische Tiefenbilder von 3D-Modellen zu rendern, denn immer mehr Lösungen (insbesondere in der Industrie) verwenden kostengünstige Tiefen-Kameras (z. B. *Microsoft Kinect* und *Intel RealSense*) für die Erkennungsaufgaben. Aufgrund einer gründlichen Untersuchung dieser Geräte und der verschiedenen Arten von Rauschen, die dem Aufnahmen beeinträchtigen, simuliert das vorgeschlagene Framework deren innere Mechanismen, indem Schlüsselfaktoren wie Sensorrauschen, Reflektionsgrade der Materialien, Oberflächengeometrie *usw.* umfassend modelliert werden. Dieses Framework ist in der Lage eine breite Palette von Tiefensensoren zu simulieren und schnell große Datensätze zu generieren. Dies wird eingesetzt, um die Algorithmen für verschiedene Erkennungsaufgaben zu trainieren und deren Leistung im Vergleich zu anderen hochmodernen Simulationmethoden konsistent und erheblich zu verbessern.

In manchen Fällen sind jedoch keine relevanten 2D- oder 3D-Objektdarstellungen zur Erzeugung von synthetischen Bildern verfügbar. Ausgehend von dieser Problematik des Datenmangels wurde eine Lösung vorgeschlagen, in der die Rekonstruktion von visuellen Szenen aus Teilbeobachtungen schrittweise durchgeführt wird. Die Bilder werden anhand

ihres Inhalts in Bezug zueinander lokalisiert und in einer globalen Gedächtnisstruktur mit räumlichen Eigenschaften registriert. Gleichzeitig kann dieses Gedächtnis abgerufen werden, um neuen Ansichten der Szene zu rendern. Darüber hinaus können bisher unbeobachtete Regionen in Übereinstimmung mit früheren Beobachtungen, Halluzinationen und globalen Vorwissen im Gedächtnis halluziniert werden. Die Wirksamkeit des vorgeschlagenen, durchgehend trainierbaren mnemonischen und generativen Systems, wird anhand von verschiedenen 2D- und 3D-Anwendungsfällen demonstriert.

Schließlich wird eine auf Convolutional Neural Networks (CNNs) basierte weiter entwickelte Pipeline vorgestellt, die die Realismislücke aus einem neuen Blickwinkel angeht. Während die meisten Methoden, die sich mit diesem Problem befassen, sich darauf konzentrieren, synthetische Datenproben (*bzw.* daraus erworbenes Wissen) näher an die echte/reale Zieldomäne zu bringen, führt die vorgeschlagene Lösung den umgekehrten Prozess durch, indem ungesehene Zielbilder in den kontrollierten synthetischen Domänen abgebildet werden. Die vorbehandelten Datenproben können dann für die nachgeschalteten Erkennungsalgorithmen übergeben werden, die selbst anhand der ähnlichen synthetischen Daten trainiert wurden, um deren Genauigkeit deutlich zu verbessern.

Für jeden Ansatz werden verschiedene qualitative und quantitative Studien durchgeführt, um mit sie den neuesten Methoden zu vergleichen. Insgesamt werden in dieser Arbeit Methoden zur Überbrückung der Realismislücke auf beiden Seiten sowie eine Lösung zur Verbesserung der Erfassung und Generierung neuer visueller Inhalte beschrieben. Daher bietet diese Dissertation eine neuartige Perspektive auf die Herausforderungen der Datenknappheit bei der Entwicklung robuster Erkennungssysteme.

Acknowledgment

Looking back at the past years, I realize how long the list of people that I have to thank for the course of my Ph.D. is. My first thoughts go to my supervisors, Prof. Dr. Harald Kosch and Dr.-Ing. Andreas Hutter. Since I joined the University of Passau, Harald has been a constant source of warm support and life-changing opportunities, and I will be forever thankful for the path that he set me on. Andreas has been as supportive, supervising me since my first days at Siemens as a graduate student. Like everyone in our team, I owe him the human context and material resources to work and grow. I could not have wished for better conditions to do my research.

I am also deeply grateful to Dr. Ziyang Wu, who has been my most frequent collaborator and unofficial adviser. Besides being the nicest person to talk with, Ziyang is a brilliant and enthusiastic researcher, a real model to follow. Along with Dr. Srikrishna Karanam, Dr. Jan Ernst, Dr. Erhan Batuhan Arisoy, and so many more, the Princeton-based team has provided me with invaluable scientific guidance. I cannot wait to meet them again in person, to properly express my gratitude! Pursuing with scientific guidance, I would also wish to thank P.D. Dr. Slobodan Ilic, Dr. Peter Amon, and all my co-workers in Munich.

A special thought goes to Sergey Zakharov—a Master student I co-supervised, now a friend and fellow Ph.D. student—for our fruitful collaboration and fun discussions. I am also grateful to all the other Siemens students for the vibrant research atmosphere and relaxing meals together. Similar thoughts go to Prof Dr. Lionel Bruni, Prof Dr. Michael Granitzer, and all the fellow members of IRIXYS (International Research and Innovation Center for Intelligent Digital Systems). The scientific workshops that we did together helped me gain confidence as a researcher and provided me with valuable feedback.

I wholeheartedly thank my family for their love and support throughout my life. My parents, Michèle and Jean-Christophe, continuously encouraged me to follow my passions, even though it meant having me roaming the world far from the homeland. My sister, Clémentine, and my brother, Jérémy, also ended up in various continents, following their own paths, and I am looking forward to visiting them and catching up.

I owe my sanity to my friends from France and elsewhere. Céline, Jessie, Jorsi, MC, Nicolas, Patsy, Samuel, Victor, *etc.* They were there for me, even though I have not been the most present friend in recent times. I cannot wait to celebrate the summer with them all.

Last and foremost, all my love goes to Varia. Her warm encouragement and presence got me through the darkest storms, and my best memories are with her. I am deeply grateful for the colors that she brings to my everyday life. She is an amazingly smart and kind person, and a true source of inspiration. May the winds carry us to new adventures together!

Contents

List of Figures	xiii
List of Tables	xv
List of Acronyms	xvii
List of Symbols	xix
1 Introduction	1
1.1 Motivation	1
1.1.1 Big Data, Deep Learning, High Expectations	2
1.1.2 Computer Vision for a Smarter Industry	3
1.2 Problem Statement	5
1.2.1 Data Scarcity in Modern Computer Vision	5
1.2.2 CAD-based Recognition and Realism Gap	6
1.3 Thesis Overview	7
1.3.1 Contributions	7
1.3.2 Outline	8
2 Background on Visual Recognition from Scarce Training Data	11
2.1 Conceptualization of Computer Vision	11
2.1.1 Introduction to Computer Vision	11
2.1.2 Development of the Field	14
2.1.3 Formalization of Computer Vision Models	18
2.2 Prevalence and Limits of Deep Learning	20
2.2.1 Development of Artificial Neural Networks	20
2.2.2 Rationale for the Success of Deep Learning	28
2.2.3 Acknowledgment of CNN Limitations	31
2.3 Data Scarcity and its Ramifications	33
2.3.1 Learnability and Data Dependencies	33
2.3.2 Dealing with Data Scarcity in Industrial Computer Vision	42
3 Realistic Depth Sensor Simulation	47
3.1 Motivation	48
3.1.1 Rise of Depth-based Computer Vision	48

3.1.2	Call for Realistic Depth Simulation Tools	49
3.2	Related Work	50
3.2.1	Recognition Algorithms and Synthetic Data	51
3.2.2	Generation of Synthetic Images	53
3.3	Methodology: Simulation of 2.5D Sensors	54
3.3.1	Study of Structured-Light Depth Sensors	55
3.3.2	End-to-End Simulation of Depth Sensors	58
3.4	Experiments and Results	62
3.4.1	Implementation Details	64
3.4.2	Depth Error Evaluation	64
3.4.3	Application to Recognition Tasks	66
3.5	Discussion	69
3.5.1	Contributions	69
3.5.2	Limitations	70
4	Novel View Synthesis through Incremental Scene Learning	73
4.1	Motivation	74
4.1.1	Visual Understanding for Autonomous Agents	74
4.1.2	Novel Image Synthesis	76
4.2	Related Work	77
4.2.1	Simultaneous Localization and Mapping	77
4.2.2	Incremental Scene Sampling	78
4.3	Methodology: Neural Pipeline for Incremental Scene Synthesis	80
4.3.1	Localization and Memorization	80
4.3.2	Anamnesis	83
4.3.3	Mnemonic Hallucination	85
4.4	Experiments and Results	86
4.4.1	Implementation Details	88
4.4.2	Navigation in 2D Images	90
4.4.3	Exploring Virtual and Real 3D Scenes	94
4.5	Discussion	96
4.5.1	Contributions	96
4.5.2	Limitations	97
5	Reversed Domain Adaptation Scheme for CAD-based Learning	99
5.1	Motivation	100
5.1.1	Bridging the Realism Gap from the Other Side	100
5.1.2	Hardening the Data Scarcity Constraint	102
5.2	Related Work	103
5.2.1	Domain Adaptation to Bridge the Realism Gap	103
5.2.2	Regression of Discriminative Representations	106
5.3	Methodology: Real-to-Synthetic Mapping through Multi-Modal Distillation	107

5.3.1	Cross-Domain Mapping via Multi-Modal Distillation	109
5.3.2	Learning from Purely Geometrical CAD Data	112
5.4	Experiments and Results	116
5.4.1	Implementation Details	116
5.4.2	Experimental Setups	118
5.4.3	Qualitative Observations	120
5.4.4	Quantitative Evaluation on Recognition Tasks	121
5.5	Discussion	123
5.5.1	Contributions	123
5.5.2	Limitations	124
6	Discussion and Future Work	127
6.1	Summary	127
6.1.1	Significance	127
6.1.2	Overall Contribution	128
6.2	Future Work	129
6.2.1	Integration of Novel Differentiable Image Generators	129
6.2.2	Beyond Image Domains	131
6.3	Epilogue	132
	Bibliography	135

List of Figures

1.1	Visualization of the exponential trends in machine learning and computer vision.	3
1.2	Screenshot of the product webpage of Easy Spares IDea®.	4
1.3	Illustration of the <i>realism gap</i> for industrial applications.	6
2.1	Usage of Eigenface features [269].	16
2.2	Representation of SIFT key points extracted from a given image (using OpenCV [28]).	17
2.3	Biological neuron and its artificial counterpart.	21
2.4	Simple representation of a gradient descent.	23
2.5	Representation of common activation functions used in neural networks (NNs).	24
2.6	<i>LeNet-5</i> architecture for hand-written digit recognition [141, 142].	27
2.7	Example of a recognition model trained on a biased dataset.	39
2.8	Pairs of color and depth images, from T-LESS [102] and LineMOD [99] datasets.	43
3.1	Real color and depth pictures, opposed to the corresponding CAD model and simulated depth image.	48
3.2	Illustration of a structured-light sensor.	56
3.3	<i>DepthSynth</i> pipeline and results for the simulation of multi-shot depth sensors.	59
3.4	Effects of material specularity on the simulation.	61
3.5	Effects of surface conditions on the simulation.	61
3.6	Detailed visual comparison with <i>BlenSor</i> [88].	63
3.7	Detailed visual comparison with Landau's solution [138, 139].	63
3.8	Standard depth error as a function of the distance and the tilt angle.	65
3.9	Standard depth error as a function of the radial distance.	65
3.10	CAD models, sample real images, and <i>DepthSynth</i> images used in the experiments.	67
3.11	Real data acquisition and processing.	67
3.12	Cumulative distribution functions on errors in translation and in rotation for pose estimation.	68
4.1	Proposed solution for scene understanding and novel view synthesis.	74
4.2	Detailed pipeline for incremental view synthesis.	75
4.3	Localization and memorization, based on MapNet [98].	81
4.4	Training of the memorization and anamnesis modules.	82
4.5	Geometrical Memory Culling.	84

4.6	Training of the hallucination module.	85
4.7	Synthesis of memorized and novel views from 2D scenes.	87
4.8	Novel view synthesis from 3D scenes.	87
4.9	Incremental and direct memory sampling of complete environments	91
4.10	Incremental exploration and hallucination.	92
5.1	Usage and results of the proposed domain adaptation method	100
5.2	Qualitative results for <i>PixelDA</i> [26] trained with or without realistic texturing.	104
5.3	Training of the proposed network h_G	108
5.4	Detailed architecture of the proposed network h_G	110
5.5	Augmentation and training results.	111
5.6	Qualitative results of the proposed generator on LineMOD [99].	113
5.7	Qualitative results of the proposed generator on T-LESS [102].	118

List of Tables

3.1	Comparison of <i>BlenSor</i> [88], Landau’s pipeline [138, 139] and <i>DepthSynth</i> w.r.t. sensor noise types.	58
4.1	Quantitative comparison on 2D and 3D scenes.	94
4.2	Ablation study on CelebA dataset [155].	94
5.1	Visual comparison of recognition training schemes.	105
5.2	Quantitative comparison of recognition pipelines, depending on the available training data, on T-LESS [102].	120
5.3	Quantitative comparison of recognition pipelines, depending on the available training data, on LineMOD [99].	121
5.4	Architectural ablation study, considering the instance classification task on LineMOD.	122

List of Acronyms

ERF	effective receptive field
2.5D	depth
2D	two-dimensional
3D	three-dimensional
6-DOF	six degrees of freedom
AE	auto-encoder
AGI	artificial general intelligence
AI	artificial intelligence
ANN	artificial neural network
APE	average position error
app	mobile application
AR	augmented reality
ATE	absolute trajectory error
AUC	area-under-the-curve
BRDF	bidirectional reflectance distribution function
CAD	computer-aided design
CCTV	closed-circuit television
CDF	cumulative distribution function
CNN	convolutional neural network
CRF	conditional random field
DND	differentiable neural dictionary
GAN	generative adversarial network
GPS	global positioning system
GPU	graphics processing unit
GQN	generative query network
GTM-SM	generative temporal model with spatial memory
HOG	histogram of oriented gradients
IC	instance classification
ICP	iterative closest point
ICPE	instance classification and pose estimation
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

IoT	Internet of things
IR	infrared
IT	information technology
LESH	local energy-based shape histogram
LSTM	long short-term memory
NN	neural network
NSS	normalized scanpath saliency
PCA	principal component analysis
RGB	red-green-blue
RGB-D	red-green-blue and depth
RNN	recurrent neural network
SAD	sum of absolute differences
SfM	structure-from-motion
SGD	stochastic gradient descent
SIFT	scale invariant feature transform
SLAM	simultaneous localization and mapping
SSIM	structural similarity
SURF	speeded up robust features
SVM	support vector machine
ToF	time-of-flight
VAE	variational auto-encoder

List of Symbols

A	augmentation pipeline
a	encoding of an agent's action
atan2	2-argument arctangent
b	bias value(s) parameterizing an artificial neuron or neuronal layer ($b = (b_0, b_1, \dots, b_n) \in \theta$)
c	number of channels (or <i>depth</i>)
d_{base}	baseline distance (usually in millimeters)
$d_{\mathcal{H}\Delta\mathcal{H}}$	expected $\mathcal{H}\Delta\mathcal{H}$ -divergence of two input marginal distributions w.r.t. to the hypothesis space \mathcal{H} [16]
$\hat{d}_{\mathcal{H}\Delta\mathcal{H}}$	empirical $\mathcal{H}\Delta\mathcal{H}$ -divergence of two input marginal distributions w.r.t. to the hypothesis space \mathcal{H} [16]
d_{disp}	disparity (in pixels)
$d_{disp,i}$	vertical displacement/disparity (in pixels)
dim	dimensionality operator
e	base of the natural logarithm
ϵ	learning rate
f_{ac}	activation function applied by a neuron or neuronal layer
f_{px}	focal length (in pixels)
$\Pi_{\mathcal{H}}$	growth function of a hypothesis space \mathcal{H} [45, 274]
\mathcal{H}	hypothesis class
h	$h \in \mathcal{H}$ machine learning model/hypothesis
h	height (usually in pixels)
k	kernel size in convolutional layers, window size
\mathcal{L}	loss function, applied to the training of machine learning models
O	<i>Big O</i> notation (asymptotic behavior)
o	projected feature map representing an agent's neighborhood
P	probability distribution
ψ	global memory structure
ReLU	rectified linear unit
$\hat{\mathcal{R}}$	empirical risk
\mathcal{R}	expected risk
s_L	sigmoid

σ	softmax
s	stride hyper-parameter in convolutional layers
t	time step
tanh	hyperbolic tangent
\mathcal{T}	<i>task</i> a model is set to tackle
Θ	ensemble of all possible parameter sets a machine learning model/hypothesis can have
θ	$\theta \in \Theta$ specific set of parameters defining a machine learning model/hypothesis
<i>vc</i>	Vapnik-Chervonenkis (VC) dimension of a hypothesis space \mathcal{H} [45, 274]
\mathbf{W}	weight tensor parameterizing a neuronal layer ($\mathbf{W} = (\omega_0, \omega_1, \dots, \omega_n) \in \theta$)
ω	weight vector parameterizing an artificial neuron
w	width (usually in pixels)
X	finite set of input elements
x	input element (<i>e.g.</i> , image or its features)
x	X-position in three-dimensional (3D) space
\mathcal{X}	input space
T	set of target elements (typically either unavailable at training time or available but unlabeled)
\tilde{x}	element (<i>e.g.</i> , image or its features) synthesized by a generative model
S	finite set of input elements and their labels available for training
Y	finite set of task-specific labels
y	true task-specific label of an element x (<i>e.g.</i> , its object class or label map, a relevant decision, <i>etc.</i>)
y	Y-position in 3D space
\mathcal{Y}	task-specific label space
\tilde{y}	predicted task-specific label of an element x , returned by a function given x
z	Z-position in 3D space, <i>i.e.</i> , depth (usually in millimeters)

Introduction

” *The errors which arise from the absence of facts are far more numerous and more durable than those which result from unsound reasoning respecting true data.*

— **Charles Babbage**

(Mathematician and 1st Computer Scientist, in “On the Economy of Machinery and Manufactures”, 1832)

In recent years, computer vision has grown into a key domain for innovation, with an increasing number of applications reshaping businesses and lifestyles. However, this development is still impeded by the recurring lack of relevant images to teach the machine-learning algorithms their recognition tasks on. As a result, a large number of methods rely on synthetic images, rendered using computer graphics tools. However, these solutions often have their performance compromised by the visual discrepancies between the synthetic images they were trained over and the target real pictures they are applied to. The present thesis addresses this issue from several angles, resulting in a variety of contributions with direct industrial impacts.

This prefatory chapter further details the underlying incentives for this research work in Section 1.1. Section 1.2 states in plain terms the central problem addressed in this thesis. The chapter ends with an overview of the thesis in Section 1.3, highlighting the main contributions and outlining the dissertation.

1.1 Motivation

The thesis presented in this document has been completed in collaboration with Siemens Corporate Technology. There, numerous projects are aiming at integrating computer vision solutions into industrial processes, in spite of harsh constraints in terms of data availability. From this concrete framework with direct repercussions, multiple theoretical considerations were derived, further motivating this research work.

1.1.1 Big Data, Deep Learning, High Expectations

The idea of artificial beings, able to reason and dedicated to assisting their creators, goes back to the most ancient myths (*e.g.*, Talos, the protective bronze giant in Greek mythology, or the Golem, the servile clay *robot* in Jewish folklore). It is indeed humankind's long dream to rise above its condition and free itself from labor – an extravagant dream that has inspired a constant flow of philosophical considerations and technical advances.

It seems that humankind has never been closer to achieving its dream, observing the frantic development of artificial intelligence (AI) the past decades. Though we are still far from delegating the functioning of our society to robots, to collectively share and enjoy the fruits of their labor, the variety of tasks automated through computer science is growing by the day. Two concomitant exponential trends are to be credited for enabling this fast-paced development of intelligent systems.

First, the quantity of data which is constantly generated and stored has been booming since the development of the first computers, thanks to the conception of better devices to digitize and store content, and the development of the Internet to share it. But more importantly, this exponential increase in the quantity of data created and shared is linked to the development of sensing technologies. Sensors are everywhere nowadays. Our *smartphones* come with a full array of them which we use daily to communicate (camera, mic, *etc.*); devices are installed, worn, or implanted to monitor our public infrastructures, homes, health, and more; produced goods are spawning data during their whole life cycles (manufacturing records, delivery tracking, *etc.*); and so on. It did not take long for scientists and investors to measure the potential of this barely exploited amount of information (*big data*), labeling it as the *new oil*. However, like oil, raw data need first to undergo a refinement process in order to power new applications – a challenging process given the complexity and scale of the task at hand.

Thankfully, the second trend that has been shaking up the technological world the past decades is the exponential increase in the computational capabilities of mankind's machines (famously coined as *Moore's law* [175]). Data processing algorithms which yesterday could only run on the best computers are now deployable on mobile devices, and methods requiring the processing of large amounts of data have seen their runtime decrease year after year, overall enabling new use-cases and catalyzing research (as illustrated in the comparative Figure 1.1).

This golden age of data science has been—and is still—benefiting one domain of AI in particular: machine learning, and more precisely deep learning. With big data as fuel and powerful computers as motor, deep learning methods—inspired by how the information flows in our brains—are addressing increasingly complex tasks. In computer vision

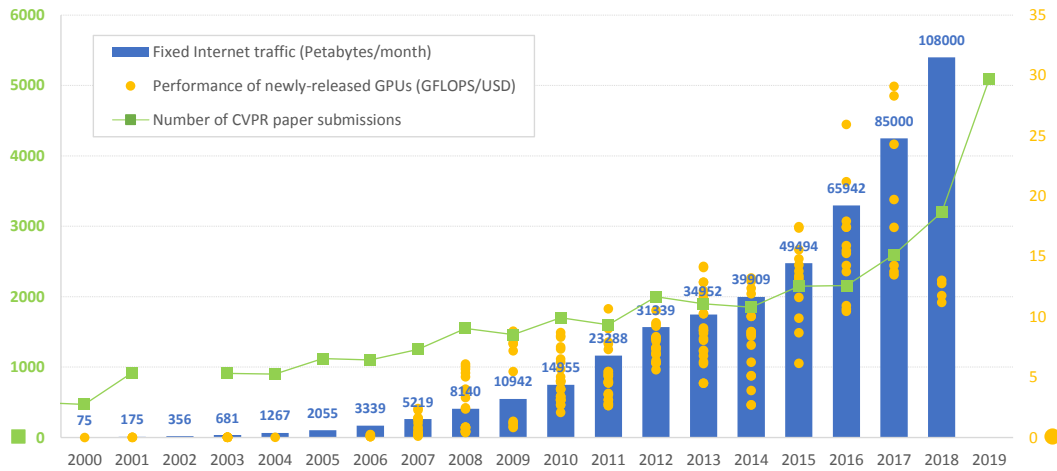


Fig. 1.1: Visualization of the exponential trends in machine learning and computer vision, as a multi-scale plotting of the Internet traffic [50] (correlated to data creation and exchange), the per-dollar power of latest computational units [167], and the number of paper submissions at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [109], over the years.

especially, the domain aiming at the automation of vision-based tasks (*e.g.*, automated scene understanding from images), deep learning has led to impressive progress and its solutions are already deployed in numerous applicative domains.

Indeed, as the generation of digital data (images and videos included) becomes ubiquitous and systemic, so does the application of machine learning, solving a growing number of complex tasks.

In this context, it is no surprise that the main leaders in AI nowadays are information technology (IT) companies such as Google and Facebook, *i.e.*, corporations that have both an abundance of data and the computational infrastructure to process it. More and more companies are similarly trying to turn the data that they have been collecting over the years into knowledge, and this knowledge into new business opportunities. Thankfully, lots of efforts are also made to turn data not only into financial capital but also into smarter systems that could benefit individuals and our societies as a whole. Leveraging novel cameras and sensors, algorithms are being developed to help medical experts provide accurate diagnoses, to assist disabled people in their daily life, to provide insight into social and environmental phenomena, *etc.*

1.1.2 Computer Vision for a Smarter Industry

The industrial sector too has plenty of raw data waiting to be mined. From design schematics to quality measurements, from video streams to label tracking, manufacturing companies

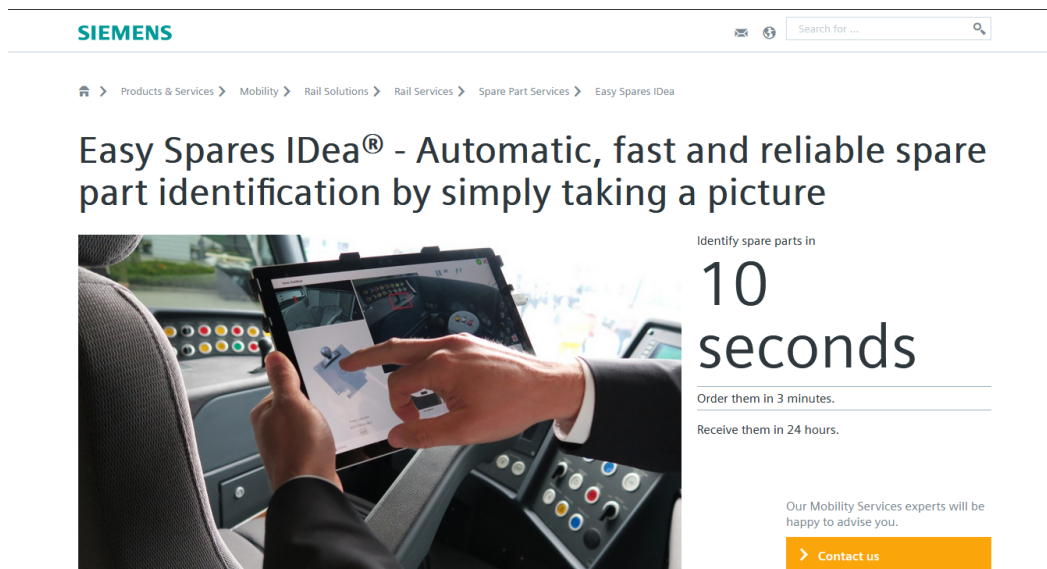


Fig. 1.2: Screenshot of the product webpage of Easy Spares IDEa® [241], a Siemens industrial app powered by computer vision which leverages some of the results presented in this dissertation (all rights over the website are reserved by Siemens AG).

are also accumulating information on the life cycle of their goods, which could be further leveraged by modern machine learning solutions.

With other recent technical advances in Internet of things (IoT), cloud computing, additive manufacturing, *etc.*, computer vision systems based on deep learning are the pillars of the “smart factory” revolution, *e.g.*, promoted as the “Industrie 4.0” by the German government through a variety of large scale projects. Data-powered computer vision could be applied to robot automation, simulation, augmented reality (AR) interfacing, *etc.*

For example, “Mobile 3D-Erfassung und 3D-Druck für industrielle Anwendungen (M3D)”, one of the Industrie 4.0 projects conducted by Siemens AG, leverages the developments in additive manufacturing and in computer vision algorithms based on three-dimensional (3D) models, to improve various aspects of the spare parts supply chain. Another similar project resulted in the commercialization of *Easy Spares IDEa*®. Combined to Siemens online platform for spare part orders, this mobile application (app) is meant to assist operators in charge of the maintenance of Siemens Mobility products (*e.g.*, trains, streetcars). From a single picture of a defect part that an operator captured with their mobile device, the app can directly recognize the part (from the Siemens catalog containing tens of thousands of entries), provide information on the object and its stock availability, let the operator order a new part in a click, *etc.*

Leveraging pre-existing and underused data (*e.g.*, Siemens’ rich virtual catalog of manufactured parts) and novel devices (*e.g.*, smartphones with new visual sensors), computer vision can, therefore, automate complex procedures and enable more horizontal

interactions between industrial actors. Conducted in the context of these aforementioned projects, the present dissertation contains several contributions directly applied to these goals.

1.2 Problem Statement

The integration of computer vision systems into industrial processes is, however, not without obstacles. The following section presents how the scarcity or irrelevance of pre-available data samples is a major problem for the development of robust recognition system, motivating the research presented in this dissertation.

1.2.1 Data Scarcity in Modern Computer Vision

Data is the life and blood of deep learning applications, as they need large amounts of relevant data—usually with precise annotations—to be optimized for their target task (*e.g.*, such a method would need to iterate over a multitude of images depicting a specific object, in order to later recognize it in new pictures). Despite being in the *big data* era, datasets large enough to properly optimize these methods are still tedious to gather and even more to annotate (as human inputs are usually required).

Moreover, oftentimes, relevant images themselves may not be available before the deployment of the recognition systems. For instance, when building automation models for the industry, it would be too time and money consuming to ask human operators to capture an array of images for every manufactured object and their components—assuming they are available for photography (target objects may not have been manufactured yet or may be in remote plants).

In the case of the aforementioned Siemens app for industrial part recognition, capturing and annotating a large-enough sequence of images for each of the tens of thousands of parts constituting the target rail cars would have been a preposterous task. Furthermore, this undertaking would have had to be repeated for every new train supported by the app. Relying on real captured images would be both cost-ineffective and unscalable.

Data scarcity is, consequently, a common problem in computer vision, and much effort has been expended in trying to *train* robust models (*i.e.*, to properly parameterize them w.r.t. their target task) despite the lack of training images or rigorous annotations [27, 53, 77, 192, 254, 272].

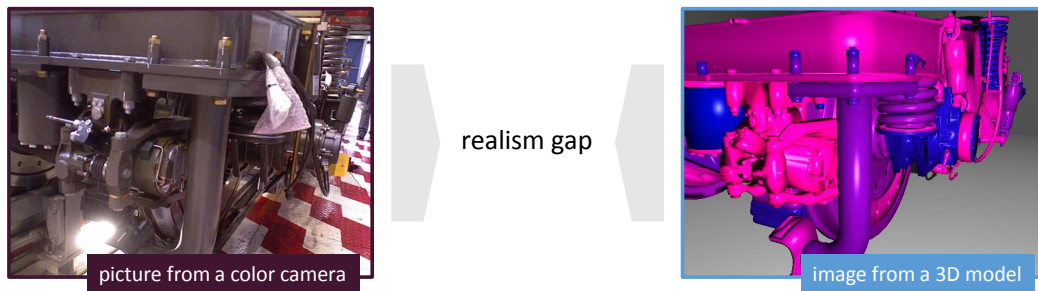


Fig. 1.3: Illustration of the *realism gap* for industrial applications.

1.2.2 CAD-based Recognition and Realism Gap

A common proxy to capturing real images for the training computer vision methods is the generation of synthetic images from 3D models [37, 147, 162, 216, 233, 250, 286]. Leveraging the latest tools in computer graphics, it is indeed straightforward to render a huge amount of relevant pictures from 3D models. Such a solution is especially interesting for industrial use-cases, as 3D computer-aided design (CAD) models of target objects are commonly pre-available.

However, for a variety of reasons further detailed along this dissertation (low photo-realistic quality of CAD models, lack of information w.r.t. overall target scenes, difficulty to simulate the noise induced by actual cameras, *etc.*), the salient discrepancies between training synthetic images and target real pictures—known as *realism gap*—still heavily impairs the application of synthetically-trained machine-learning algorithms to real-life situations. The knowledge that these methods acquired over synthetic data does not always apply to real images, due to the realism gap. For instance, the synthetic images may contain objects rendered with slightly different colors, the images may be less cluttered or noisy than real ones, the lighting conditions may differ, *etc.* Having been trained only w.r.t. this synthetic domain, it appears normal that algorithms would fail to fulfill their task on real data if they cannot rely anymore on the only visual cues that they know.

Solving the problem of data scarcity by somehow bridging the realism gap has consequently become a pivotal issue in computer vision, and is the central motivation behind this thesis.

Several works have already been proposed to partly close this gap, either through the generation of more relevant synthetic images [27, 88, 139, 254] or through the transfer or adaptation of the knowledge acquired by algorithms to the target use-cases [77, 192, 272, 300]. But most of these solutions are based on the assumption that either a small set of real images or highly-realistic 3D models of the target objects (or both) are available at training time. In other words, with such information at hand, most state-of-the-art methods only consider use-cases where the realism gap is already narrowed.

Motivated by industrial applications where only colorless CAD models of target objects and a bare knowledge of the target visual conditions are available, the present thesis, therefore, addresses the realism gap in a more generic and systemic manner.

1.3 Thesis Overview

Articulated around the core themes of data scarcity and realism gap in computer vision, the present dissertation introduces various contributions, each tackling a different sub-problem. This section contains a brief overview of the thesis, listing the main contributions and presenting the outline of the remaining chapters.

1.3.1 Contributions

Considering the development of robust systems for visual recognition from CAD data, the two main contributions of the present thesis are addressing the realism gap each from a different rim. A pipeline is proposed to simulate specific visual sensors commonly used in the industry (depth sensors) in order to render synthetic images closer to the real ones; whereas another solution is presented to map unseen real images into controlled synthetic domains for easier recognition. Additionally, this thesis proposes a novel method to build a global representation of an object or scene from a small set of partial images (*e.g.*, when 3D modeling has not been done yet or cannot be) and to generate new images from it. Overall, this dissertation contains the following high-level contributions:

Simulation Pipeline for the Generation of Realistic Depth Scans.

Since their first commercialization almost a decade ago, low-cost commodity depth (2.5D) sensors such as the *Microsoft Kinect* and *Structure IO* have been used in a variety of recognition systems [70, 164, 223, 242, 247]. Measuring the distance to objects in their field of view, these devices are especially suitable to CAD-based recognition, *i.e.*, when the shape of the target objects is known but not necessarily their exact visual appearance (color, texture, *etc.*).

When real 2.5D data is unavailable or too scarce to optimize recognition methods, it is straightforward to render synthetic 2.5D images from 3D models with computer graphics tools. However, real 2.5D scans are suffering from various types of noise and clutter. The realism gap is, therefore, also impeding depth-based recognition methods trained on synthetic data.

Based on a thorough study of depth sensors and the types of noise affecting them, an end-to-end pipeline is proposed, which simulates the mechanisms of the actual devices in order

to generate images with the same visual quality. Quantitative evaluations on a variety of depth-based recognition tasks demonstrate that training algorithms on images generated by the proposed simulation pipeline improves their end accuracy when applied to the target real images, compared to the same algorithms trained over other state-of-the-art simulators.

Generative Method to Map Unseen Real Data into Synthetic Domains.

Addressing the realism gap from the opposite angle, a second solution is proposed to train an artificial neural network (ANN) to denoise and declutter unseen real images, to map them closer to the synthetic image domain. The processed images can then be passed to any recognition methods that have been trained purely on synthetic data. Therefore applied to images similar to those they were trained on, the recognition methods perform much more accurately, as demonstrated once again over a series of quantitative experiments.

To learn its mapping from unknown real image domains into the synthetic one when only texture-less 3D models are provided (*i.e.*, with no real image available at training time), the solution leverages an advanced neural architecture (with multi-task decoders and self-attentive distillation) and domain randomization [225, 265] (*i.e.*, learning against a noise source corrupting the synthetic data).

Mnemonic System for Incremental Scene Registration and Synthesis.

For some applications, data scarcity is such that only a dozen of images capturing a target object or scene are available, with no 3D model or other representation to generate additional samples. Similarly, in other instances, one may want an autonomous agent (*e.g.*, a robot or drone) to learn from its observations as it explores new environments.

Considering the latter case, Henriques and Vedaldi [98] proposed a neural method to incrementally build a global representation of new scenes from partial observations, localizing the images one to another based on their content and registering them in a spatial mnemonic structure. In this thesis, their solution is incorporated into a larger pipeline, which can interpolate from the memorized features and extrapolate from previous knowledge to hallucinate the content of unobserved regions at each time step, and can be used to synthesize novel views (*e.g.*, from unexplored viewpoints). This novel pipeline ensures the consistency of the generated images from one to another, *e.g.*, enabling the generation of larger datasets for downstream tasks.

1.3.2 Outline

The remainder of this dissertation is structured as follows:

Chapter 2 – Background on Visual Recognition from Scarce Training Data.

The following chapter provides the readers with the necessary theoretical background, contextualizing the thesis. Starting with a formal presentation of modern computer vision and deep learning, it details how data scarcity and realism gap impede the training of robust recognition methods, referring to seminal works on domain adaptation.

Chapter 3 – Realistic Depth Sensor Simulation.

Focusing first on depth-based recognition in industrial settings, this chapter presents a study of depth sensors and introduces the simulation pipeline aiming at generating realistic images from CAD models to facilitate the training of recognition methods. Qualitative and quantitative results are shared to highlight the correctness and versatility of this solution, followed by a discussion on its usage and limitations.

Chapter 4 – Incremental Novel View Synthesis through Scene Learning.

Despite its effectiveness, the aforementioned simulation pipeline—like many other rendering solutions—rely on the availability and quality of object representations (*e.g.*, 3D models). Considering the registration of new representations from partial visual observations and the direct synthesis of novel views, this chapter formalizes an end-to-end derivable read/write mnemonic system for incremental scene synthesis. Though it could presumptively benefit from more powerful generative models and denser memory, this pioneering study provides satisfying results, shared and discussed in the chapter.

Chapter 5 – Reversed Domain Adaptation Scheme for CAD-based Learning.

Addressing again data scarcity for CAD-based recognition, this chapter proposes a novel domain adaptation scheme, training an ANN to denoise, declutter, and map unseen real images into synthetic domains, in order to facilitate downstream recognition tasks. In this chapter, the method is applied to the projection of real color images into geometrical representations which can easily be produced from 3D models. A series of experiments illustrate how this pre-processing greatly improves the accuracy of recognition methods which had been trained on synthetic data.

Chapter 6 – Discussion and Future Work.

The final chapter summarizes the thesis, discusses its overall contribution to the state-of-the-art, and provides some insight regarding possible future work.

Background on Visual Recognition from Scarce Training Data

” *Learning is not attained by chance, it must be sought for with ardor and diligence.*

— **Abigail Adams**

(First Lady of the United States, 1780.)

Teaching algorithms to expertly process visual information is a complex task, made much harder when the relevant data to extract knowledge from is scarce. In this chapter, a formal background is provided to substantiate this intuitive statement and to put in perspective the contributions thenceforth.

In Section 2.1, a presentation of the field of computer vision is provided, highlighting its contemporary challenges and formalizing its models. Section 2.2 expands on artificial neural networks, their development into deep learning, and their prevalence and limitations for visual tasks. Section 2.3 concludes this chapter by formalizing the importance of relevant and rich data for the training of machine learning models, and discussing proxy data sources specific to computer vision such as synthetically generated training images.

2.1 Conceptualization of Computer Vision

In order to contextualize the contemporary challenges of computer vision that this thesis is tackling, this research field is briefly introduced, along with the related formalism reused throughout this document.

2.1.1 Introduction to Computer Vision

Computer vision has become so ubiquitous that its definition can drastically vary from one expert to another. This introductory section paints a global picture of computer vision, highlighting its domains of application and challenges.

Definition and Motivations

Computer vision covers a large panel of topics, sitting at the crossroad of several research and development fields, such as computer science (algorithms, data processing, graphics), physics (optics, sensors), mathematics (calculus, information theory), biology (visual stimuli, neural processing), *etc.* Being an interdisciplinary field, computer vision emanates from a variety of theoretical and practical motivations. For instance, while some scientists are focusing on simulating our biological visual system through mathematical models, other experts are developing bottom-up autonomous systems relying on visual stimuli to interact with their environment [105]. However, down to its core, computer vision can be summarized as the *automatic extraction and understanding of higher-level information from digital visual contents.*

We humans rarely stop to contemplate the wonders our own brain achieves when it comes to processing visual information. But our ability to decipher the stimuli coming from our eyes' photoreceptors, to recognize objects seen only once before, to tell apart faces using the smallest details, to instantly build a mental geometrical representation of a new room we are walking in, *etc.* is little short of incredible. For computers, however, images are just digital *blobs of pixels*, matrices of integer values with no further meaning [205].

Therefore, the main goals of computer vision are not only to develop tools for digital systems to automatically extract meaningful features from these pixel matrices, but also to provide them with the computational and contextual means to *understand* the semantic information relayed by these features. Indeed, plenty of *image processing* methods are already available nowadays, to extract basic image features such as edges, gradients, contours, *etc.* [35, 66, 94, 200]. However, these features themselves are not enough to draw intelligent conclusions. Background knowledge should be acquired and applied to link some feature arrangements to specific semantic information.

Taking a concrete example, a low-level understanding is needed to associate some circular contour and monochrome features to the presence of a red circle in a picture. Then knowledge of real-world objects is required to understand that this red circle and other neighboring features represent a traffic light. Finally, the ability to project the position and orientation of this object from the image coordinate system to the three-dimensional (3D) world system, *e.g.*, relative to the car the picture was taken from, as well as an understanding of traffic rules, are required for an autonomous driving system to decide and take action upon this visual input. This bottom-up process from perception (“*a red circle of pixels is observed*”) to decision (“*it represents a red traffic light, positioned in front of the car, meaning the driving agent should carefully stop*”) defines modern computer vision.

In recent years, thanks to a driven research community fueled by the biggest IT companies and the ever-increasing availability of data and visual sensors, layers of abstraction and understanding have been stacked onto this bottom-up approach to visual intelligence. More and more, scientists are proposing ambitious autonomous systems able to supersede humans for routine or complex visual tasks (*e.g.*, detection of defect parts in large manufactured systems, video-surveillance of high-risk areas, vision-based vehicle navigation, *etc.*) or even to solve tasks beyond human abilities (*e.g.*, automatic annotation of large media streams, medical screening using sensors performing in the non-visible light domain, *etc.*).

Applications and Knowledge Transferability

Applications of computer vision are indeed numerous and varied. This heterogeneity led to an abundance of theoretical and practical developments over the years, but may also be holding back the field as a whole.

Computer Vision in the Wild. Content recognition (*i.e.*, the automatic semantic annotation of visual content) is probably the most ubiquitous task tackled in computer vision, and can be further divided in a panel of cases based on the granularity of the recognition (*e.g.*, from *image-level* recognition for image classification to *pixel-level* annotation for semantic segmentation), on the definition of the label space (*e.g.*, predefined and fixed set of labels for *object* or *category classification*, or dynamic partitioning for *person re-identification*), on the input dimensionality (*e.g.*, single-image face recognition versus video-based action classification or instance tracking), *etc.* Tasks can also have continuous label spaces, *e.g.*, when developing algorithms to regress the position and orientation of target elements relative to the camera in the 3D space, *i.e.*, attempting to reverse the projection from 3D to image space.

Other algorithms are focusing on extracting and matching features from sets of images, to understand their relations and build higher-level models. The most widespread examples are stereo-matching (the process of regressing the distance to the camera for every pair of pixels in stereo-images, as done by two-eyed animals) and simultaneous localization and mapping (SLAM, the simultaneous registration of images from an agent into a common map representation and tracking of the agent inside this representation).

Learning recurrent features and their relations in larger datasets, computer vision models can also be used to infer new relevant content, for tasks like image completion (generating relevant content for missing image patches) or dataset augmentation (generating novel images following the content distribution of a given dataset).

Toward Generalisable Models? However, like machine learning, computer vision is still far from achieving a *visual* artificial general intelligence (AGI), *i.e.*, a generic, all-in-one, model for visual understanding. Experts are still tackling applications separately, applying various solutions to the heterogeneous input data and target predictions. As later discussed in this thesis, current models are not yet able to efficiently transfer their expertise from one task to another (at least without undergoing some *adaptation*).

This lack of transferability is a key challenge of modern computer vision, as it impairs both the generalisability of state-of-the-art solutions (toward building an AGI) and also their proper adaptation to visual tasks that do not meet the requirements in terms of knowledge base (*e.g.*, lacking the pre-available samples and annotations to tune the solutions).

2.1.2 Development of the Field

According to Confucius, one should “study the past if [one] would define the future.” By painting the evolution of computer vision in the decades since its creation, this subsection highlights and puts in perspective some of the modern challenges of this field.¹

Early Approaches to Computer Recognition

From the start, researchers underestimated the complexity of visual recognition. The evolution of our understanding w.r.t. biological cognition and perception not only had important sociological and philosophical consequences on our modern societies, but it also heavily impacted the development of artificial intelligence (AI) and computer vision.

Abstract Cognition versus Animal Functions. Computer vision as a domain started in the sixties, among the AI research community which was dominated by the *symbolic* approach [194]. According to this paradigm, many aspects of intelligence can be achieved through the manipulation of symbols. This assumes that “a sharp line can be drawn between the physical and intellectual abilities” of an intelligent agent (*e.g.*, a human being) [267].

This way of thinking is reminiscent of the Cartesian mind-body dualism, according to which, mental phenomena are “non-physical”, and mind and body are distinct and separable. Building on this assumption, researchers hence focused on solving purely intellectual tasks such as chess and checkers [228, 236]. Proposing ad hoc technical solutions to well-defined but constrained problems, this line of research originated from an assumption that solutions

¹This section reuses some paragraphs and figures which I authored for the first chapter of the book *Hands-On Computer Vision With TensorFlow 2* [205]. Authorization to share this content has been kindly granted by the publisher.

to such problems might be significant to solving AGI (*e.g.*, that dealing with the particular task of playing chess can be transferable to a broader class of problems such as filter design, language translation, *etc.*).

While we owe the symbolists the foundations of the domain and various works still relevant nowadays—*e.g.*, Turing’s Imitation Game [267], his primitive formulation of reinforcement learning [268], or the rule-based systems [228, 236]—one could object to their approach that intelligence should be regarded as the ability to accomplish a wide variety of goals, or that trying to reproduce human intelligence—seen as the pinnacle—is a classic anthropocentric error (human self-importance). As these researchers underestimated the complexity of lower animal functions such as perception, it took years for computer vision to be given the attention matching its challenges and to grow into a field of its own.

Understating Perception. Marvin Minsky was one of the first to outline an approach towards building AI systems based on perception [170]. Minsky argued that with the use of lower functions such as pattern recognition, learning, planning, and induction, it could be possible to build machines capable of solving a variety of higher-level problems. He also pointed out the importance of perception and the concept of an agent interacting with the environment, termed as a “reinforcement” machine. While some of these ideas might be considered a transition towards embodied learning, the approach was still substantially symbolic. Minsky too underestimated the complexity of perception problems such as object recognition and missed the strong connection between perception and action.

In the eighties, it was clear that the symbolic approach was stalling, and the aforementioned objections grew louder. Researchers in the field of robotics switched their focus to solving problems that would allow their machines to perceive, learn, recognize and thus act and interact with the environment—aspects of intelligence that were considered less important by the symbolists. Led by John McCarthy, the Dartmouth proposal [165] is a clear example of how the symbolists misconstrued the problem, disregarding whole aspects of what makes intelligent agents able to interact.

Subsymbolic Revolution. This observed failure of the figureheads of symbolism led other researchers to look into new ways of considering intelligence and developing machines capable of solving concrete challenges, focusing on perception and action. This “subsymbolic” movement aimed to approach intelligence by acknowledging the strong connection between perception and action, hence without requiring a symbolic representation of knowledge.

In 1984, in support of the importance of perception to the development of intelligent systems, Hans Moravec [176] noted that our nervous system, through the process of evolution, has

The diagram illustrates the decomposition of a portrait image. On the left is a color portrait of a man's face. This is followed by an equals sign. To the right of the equals sign is a sequence of terms: a mean image (a blurred version of the original face), followed by a plus sign, a coefficient '0.91', a multiplication sign, a grayscale eigen image (a dark, textured patch), a minus sign, a coefficient '3.07', a multiplication sign, another grayscale eigen image (a different dark, textured patch), a plus sign, a coefficient '3.27', a multiplication sign, a third grayscale eigen image (a lighter, textured patch), a minus sign, a coefficient '2.63', a multiplication sign, a fourth grayscale eigen image (a darker, textured patch), and finally a plus sign followed by an ellipsis '...'. A bracket underneath the entire right-hand side of the equation is labeled 'weighted sum of the database's eigen vectors'. A separate bracket underneath the mean image is labeled 'mean image'.

Fig. 2.1: Usage of Eigenface features [269]. A portrait image is decomposed into the mean image and weighted sum of *eigen* images. These mean and eigen images were computed over a larger face dataset.

developed specifically to tackle perceptual tasks. As he noted, even if modern computers are extremely efficient at arithmetic, they cannot compete with such perceptual abilities. In this sense, programming a computer to solve *purely intellectual tasks* (e.g., playing chess) does not necessarily contribute to the development of systems that are intelligent in a general sense or relative to human intelligence.

Among the major contributions to the reshaping of AI were those made by Rodney A. Brooks [30, 31], who rejected symbolic AI and proposed to focus on embodied intelligence. Brooks dwelt onto the importance of testing developed AI systems in the real world. He was a great defender of a modular approach to intelligence that would slowly link perception to action. In particular, he believed that robotic behavior should not be guided by symbolic representations of the world. Instead, sensory information should guide action selection, following a bottom-up approach—the one mentioned in the introduction, that would become central to computer vision and artificial intelligence.

From Feature Extraction to Decision

Therefore, tackling computer vision from the bottom, researchers focused their efforts on mimicking the primitive mechanisms of animal perception. This shift in the approach to computer vision marked the beginning of *feature-based recognition*.

Crafting Visual Features. Inspiration came from developments in sensory neurobiology, which highlighted the presence of simple, highly specialized, cellular blocks (e.g., in animals' visual cortex) in charge of detecting basic *features* like edges, color changes, *etc.* [106, 144, 161]. Computer scientists developed matricial and gradient-based methods to extract such features in digital images. According to Richard Szeliski in the introduction of his famous book “Computer vision: algorithms and applications” [260], the first computer vision algorithms were based on the extraction of lines and edges, which could be used to gain a basic understanding of the two-dimensional (2D) projected geometry and, e.g., to identify letters and digits in scanned texts [51, 107]. Texture and lighting information was then also taken into account, leading to early object classifiers based on feature correspondences. Over the years, increasingly expressive and robust features were considered.

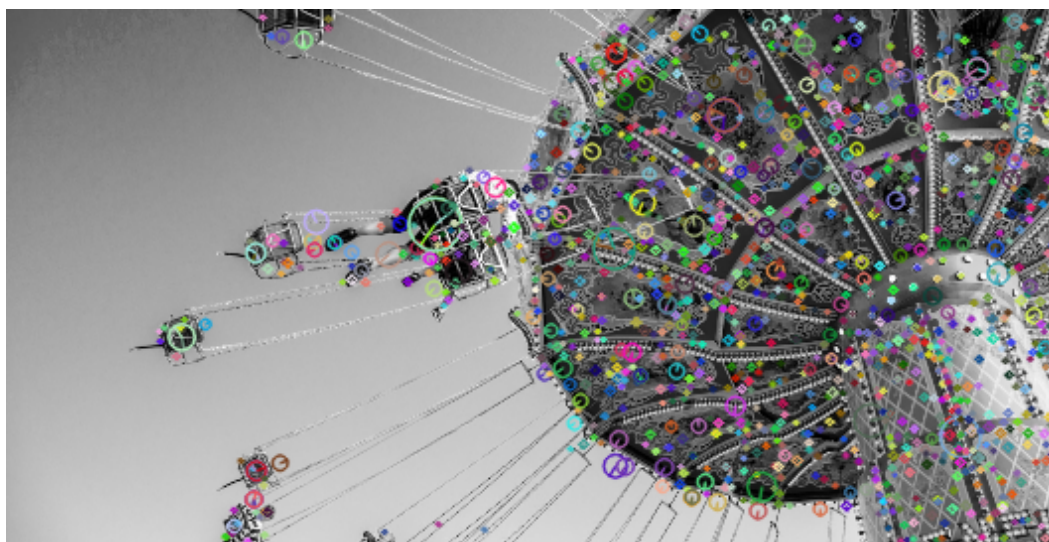


Fig. 2.2: Representation of SIFT key points extracted from a given image (using OpenCV [28]). For each localized key point, the radius of the circle represents the size of its meaningful neighborhood, and the line shows its orientation (*i.e.*, the main orientation of the local gradient).

In the nineties, features based on statistical analyses of image datasets—such as principal component analysis (PCA) [287]—were successfully applied to complex recognition problems such as face classification (*c.f.* the “Eigenface” features proposed by Turk and Pentland [269] and illustrated in Figure 2.1). Later that decade, the popular scale invariant feature transform (SIFT) method was also proposed by Lowe [156], based on pairs of key points extracted from images and their respective local gradient-based features computed to be robust to changes in scale and orientation (*c.f.* Figure 2.2). Other local feature extractors were proposed the following decades, like speeded up robust features (SURF) [13] and local energy-based shape histograms (LESHs) [230].

Researchers had indeed become more and more ingenious at *hand-crafting* visual features, combining local gradients, histograms, *etc.* As the expressiveness and robustness of engineered features kept improving, so did the decision-making methods relying on these features. While direct matching of features extracted from query images to labeled features from a known dataset is fine for concise problems like character recognition, such correspondence-based methods cannot easily scale up to larger problems or deal with intra-category variations (*e.g.*, matching features across images for face recognition may fail if the lighting conditions are different, if the person got a new haircut in between, *etc.*).

Development and Limitations of Decision-Making Models. Therefore, machine learning methods have been adopted and adapted by the computer vision community to reduce the prediction search space or to statistically model potential feature variability among classes. For instance, in the late nineties, support vector machines

(SVMs)—standardized by Cortes and Vapnik [52]—became the default solution to map high-dimensional structures (like images or their feature vectors) to simpler labels (like classes).

Such machine learning methods are typically defined by a set of parameters which can be tuned to approximate the non-linear function mapping the provided inputs to the desired labels. Proper parameters for a specific task are *learned* by the algorithms during a *training phase*. This training usually consists of an iterative supervised optimization process done over relevant data samples and their ground-truth labels. Other significant machine learning methods applied to computer vision are Bayesian models [190, 259], random forests [146], bags of words [298], and neural networks [140, 220].

With the overwhelming success of convolutional neural networks (CNNs) after 2012 [130], researchers in computer vision went from hand-crafting features to teaching computational models to extract optimal features w.r.t. their target tasks. While CNNs brought a new era of vision-based applications, the unequivocal reliance on these models has been criticized by many, as detailed in the next sections of this chapter. Nevertheless, the cognitive layer CNNs added to the bottom-up visual intelligence is most valuable if not an end by itself. Researchers are nowadays exploring additional solutions built upon the current stack, especially with the goal to finally increase the *horizontal* coverage of current intelligent systems (*i.e.*, their generalisability and ability to connect heterogeneous abstract concepts).

2.1.3 Formalization of Computer Vision Models

Adopting notations commonly used in machine learning and computer vision [15, 16, 42, 53, 54, 117, 132, 192, 193, 215, 277] and borrowed from statistical learning theory [273], let $\mathcal{X} \subset \mathbb{R}^d$ be a d -dimensional input space and $X \in \mathcal{X}$ be an observable variable with an arbitrary but fixed marginal probability distribution $P(X)$ (*e.g.*, $\{x_0, x_1, \dots, x_n\}$ sampled *i.i.d.* according to $P(X)$ are input images, or their vectors of extracted features, that a model may have to process). Similarly, let \mathcal{Y} be the label space and $Y \in \mathcal{Y}$ be an observable variable with a distribution $P(Y)$. \mathcal{Y} can be discrete (*e.g.*, categories of pictured objects, next actions an autonomous agent should take w.r.t. the latest visual inputs) or continuous (*e.g.*, the states of the pictured objects like their 3D poses).

Discriminative Models

In computer vision and machine learning, most solutions have a *discriminative* role [18]. They can be defined as parameterized predictive functions (or *hypotheses*) belonging to the hypothesis class $\mathcal{H}_D = \{h_\theta : \mathcal{X} \rightarrow \mathcal{Y} \mid \theta \in \Theta\}$ with Θ ensemble of all possible parameter sets for this family of hypotheses. Configured by a set of parameters θ and given x sampled

i.i.d. according to $P(X)$, a function $h_\theta(x) = h(x; \theta) = \tilde{y}$ has for purpose to return $\tilde{y} = y$, *i.e.*, to estimate the true task-specific label y out of all other candidates belonging to the target space \mathcal{Y} .

Therefore, a discriminative task \mathcal{T}_D can be expressed as $\mathcal{T}_D = \{\mathcal{Y}, f_D\}$, *i.e.*, defined by its label space \mathcal{Y} and its objective labeling function f_D which has to be approximated. Pushing the probabilistic approach further, discriminative methods $h \in \mathcal{H}_D$ tackling \mathcal{T}_D can be considered as modeling the posterior probability distribution $P(Y|X)$ (this applies to solutions for logistic or *softmax* regression, *i.e.*, methods whose predictions are normalized applying the logistic/sigmoid or softmax function).

The optimal hypothesis h^* is the function parameterized by θ^* such that:

$$\theta^* = \arg \min_{\theta \in \Theta} \mathcal{R}(h_\theta) = \arg \min_{\theta \in \Theta} \mathbf{E}_{x \sim P(X)} [\mathcal{L}(f_D(x), h(x; \theta))] , \quad (2.1)$$

i.e., so that the hypothesis minimizes the *expected risk* \mathcal{R} (also named *expected loss*, *generalization risk*, or *out-of-sample risk*) expressed by a loss function \mathcal{L} that quantifies the difference between each predicted \tilde{y} and ground-truth y according to the task context (*e.g.*, a loss function could more heavily punish false negatives than false positives for safety-related tasks).

In practice, since the distribution $P(X)$ is unknown, the expected risk cannot be directly computed and can only be approximated over an empirical dataset of m pairs $\{(x, y)\}_{i=1}^m$ with x_i input elements gathered *i.i.d.* from $P(X)$ and $y = f_D(x) \in \mathcal{Y}$ their respective true labels. Given this available dataset, hypotheses should be minimizing the *empirical risk* $\hat{\mathcal{R}}$ defined as:

$$\hat{\mathcal{R}}(h_\theta) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(f_D(x_i), h(x_i; \theta)) . \quad (2.2)$$

Generative Models

Another type of computer vision methods is instead applied to *generative* tasks $\mathcal{T}_G = \{\mathcal{X}, f_G\}$ [18], *i.e.*, trying to approximate f_G and to infer the true distribution $P(X)$. In other words, whereas discriminative models learn to label input samples based on specific features (*e.g.*, to classify cat pictures per breed), generative models can be used to sample new *plausible* elements from \mathcal{X} according to $P(X)$ (*e.g.*, to generate realistic images from the domain “cat pictures”). In some cases, generative methods can also be trained to model $P(X|Y)$, *i.e.*, to approximate the unknown function $f_G : \mathcal{Y} \rightarrow \mathcal{X}$ which samples valid inputs x w.r.t. provided labels y (*e.g.*, realistic cat images x corresponding to a specific breed y).

According to a majority of experts, such generative models hold the key to the next stage of machine learning. To be able to generate a large and varied amount of novel elements despite their mnemonic and computational limitations, these solutions have to distillate the dataset, to uncover its structure and key features, *etc.* They have to *understand* the data.

While the following section will detail the progress made in that direction with the development of CNNs, computer vision as a whole still has a long way to go. Nevertheless, modern discriminative and generative models are already covering advanced real-life applications, leading to the most interesting results when applied jointly, as the remaining of this thesis will highlight.

2.2 Prevalence and Limits of Deep Learning

The past decade has seen the rise of *deep learning*. Outshining most of the traditional computer vision methods, deep neural networks have been adopted by the vast majority of the community. This section provides the necessary background to understand their mechanisms and their success, while discussing their limitations.

2.2.1 Development of Artificial Neural Networks

Interestingly, before the current decade, not many people would have bet on the supremacy of artificial neural networks (ANNs) observed nowadays in machine learning and related domains. Indeed, since their invention, it has been a bumpy road to success as presented in this subsection, along with details on their functioning.

Birth of the Perceptron

In the fifties, Frank Rosenblatt proposed the *perceptron*, a linear classifier and the underlying block of first ANNs.

Biological inspiration. ANNs are loosely inspired by the biological mechanisms supporting animal cognition (*e.g.*, our own thoughts). Our brain is a complex network of *neurons*, passing information to each other, processing sensory inputs (as electrical and chemical signals) into thoughts and actions.

Each neuron receives its electrical inputs from its *dendrites*, some cell fibers which propagate the electrical signal from the *synapses* (the junctions with preceding neurons) to the *soma* (the neuron's main body). If the accumulated electrical stimulation exceeds a specific

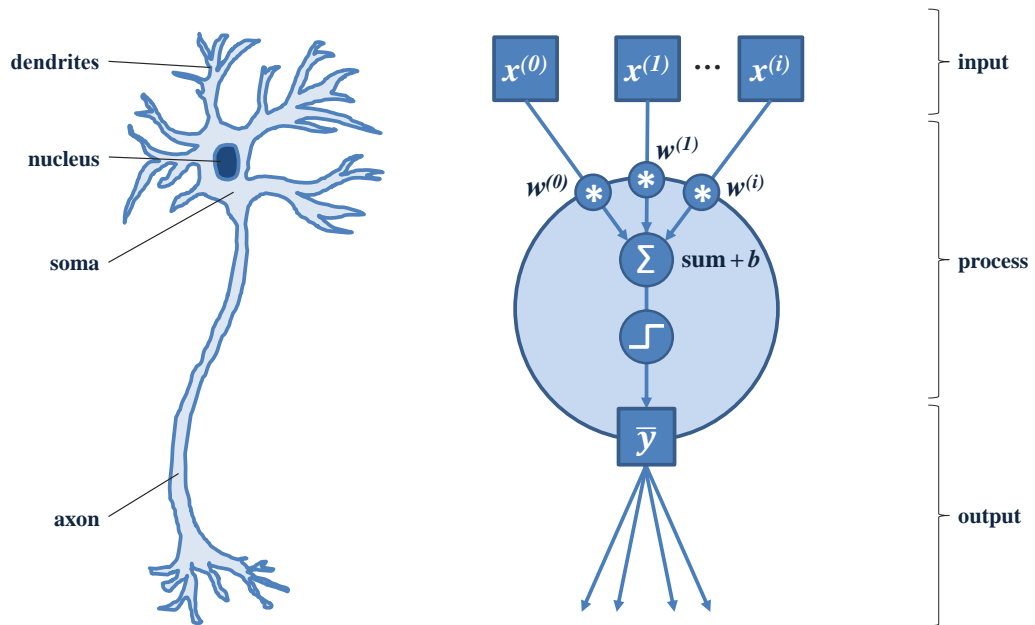


Fig. 2.3: Biological neuron and its artificial counterpart. On the left, a simplified biological neuron is depicted,; and on the right, a perceptron is represented.

threshold, the cell is activated and the electrical impulse is propagated further to the next cells through the neuron’s *axon* (its “output cable” ending with several synapses linking to other neurons). Each neuron can thus be seen as a relatively simple signal processing unit (a filter).

Mathematical model. Like its biological counterpart (represented in Figure 2.3), the artificial neuron accepts multiple input values, *i.e.*, an input vector x . The model sums its values together and finally applies an *activation* function to obtain the output signal which can be passed to the next neurons in the network (the latter can be seen as a directed graph).

The input summation performed by the perceptrons is done in a weighted way. Each value $x^{(i)} \in x$ is scaled by a specific weight $w^{(i)} \in \omega$. Oftentimes, a *bias* value is added to this weighted sum, before passing the result to the activation function of the perceptron. Historically, as the first perceptrons were used as binary classifiers, the centered *step* function ($f_{ac}(z) = 0$ if $z < 0$ else 1) was applied, its binary output corresponding to the model’s prediction \tilde{y} [220]. The whole process can be summarized as $\tilde{y} = f_{ac}(x \cdot \omega + b)$, with $\{\omega, b\} = \theta$ the parameters of the perceptrons to be trained in order to achieve proper recognition (a simple iterative linear algorithm was presented to optimize a perceptron over available samples [220]).

Once again mimicking our brain, perceptrons can be grouped into *layers* linearly combining them. A *dense* layer fully-connecting n artificial neurons can be defined by its trainable

weight matrix $W = (\omega_0, \omega_1, \dots, \omega_n)$ and bias vector $b = (b_0, b_1, \dots, b_n)$; and its prediction $\tilde{y} \in \mathbb{Z}_2^n$ (i.e., \tilde{y} a n -dimensional binary vector) can be obtained as $\tilde{y} = f_{ac}(x \cdot W + b)$.

Winter of Connectionism. Though the perceptrons and other *connectionist* models earned a certain fame in the sixties, it all ended in 1969 when Marvin Minsky and Seymour Papert published a book highlighting the perceived limitations of these models, famously demonstrating that the perceptron could not learn a function as simple as *XOR* (i.e., exclusive *OR*) [171]—as a perceptron can only fit linear functions (*XOR* is not).

This proto-AI winter affected the research into ANNs for years, until it was demonstrated that, unlike single-layer perceptrons, *fully-connected* ANNs composed of multiple layers stacked together could actually model nonlinear functions (i.e., performing as a directed graph, passing the output vector of a layer as input to the next until the final prediction).

Optimizing by Backpropagation

To efficiently optimize multi-layer neural networks (NNs) for their tasks, more advanced techniques had to be developed. Invented in the sixties [33, 122] and refined in the seventies [60, 151] by multiple research teams, the *backpropagation* method became the preferred solution to train neural networks [209, 224]. This *training* procedure works by computing the network's error and back-propagating it through the layers of perceptrons to update their parameters, using *derivatives*.

Gradient Descent. As mentioned in Subsection 2.1.3, a recognition model is evaluated by an empirical risk expressed by the mean value of a task-specific loss function \mathcal{L} quantifying the discrepancies between the true labels $y_i = f(x_i)$ and predictions $\tilde{y}_i = h(x_i; \theta)$ over the available dataset $S = \{(x_i, y_i)\}_{i=1}^m$, with f the target labeling function that the network h_θ is tasked to approximate (c.f. Equation 2.2).

To tune the parameters θ so to minimize the loss in this supervised setting², one needs to further evaluate the role that each parameter is playing in the resulting predictions and error of the NN. Therefore, a solution to tune a parameter (e.g., $W \in \theta$) is to compute the derivative of the loss w.r.t. this parameter (e.g., $\frac{d\mathcal{L}}{dW}$). Once this derivative computed for each parameter of an NN after a round of predictions, those parameters can be updated as follows: $\theta \leftarrow \theta - \epsilon \frac{d\mathcal{L}}{d\theta}$ (with ϵ a learning rate affecting the update amplitude of each training iteration).

²A supervised training can be seen as the procedure of teaching to a method a mapping between two modalities, with a supervisor entity providing feedback for every prediction the method makes in order to optimize its parameters θ accordingly.

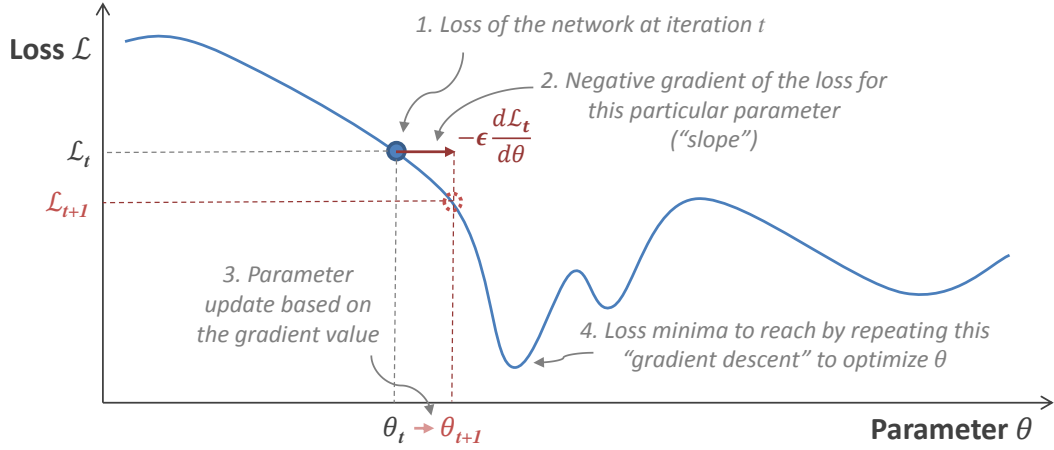


Fig. 2.4: Simple representation of a gradient descent, to optimize the parameter θ of a function h according to a loss $\mathcal{L}(y, h(x; \theta))$ for $(x, y) \in S$.

This iterative optimization can be pictured as walking step by step down the “slope” of the loss function w.r.t. each parameter, hence its name *gradient descent* (illustrated in Figure 2.4).

Backpropagation. In order to estimate the derivatives of the loss function w.r.t. the parameters of each layer composing a neural network, the backpropagation technique makes use of the *chain rule*. The chain rule offers an elegant formula to express the derivative of composed functions: $(f^{(2)} \circ f^{(1)})' = (f^{(2)'}) \cdot f^{(1)'}$. Considering each layer as a function $h^{(i)}(\cdot; \theta^{(i)})$ for $i \in [0, k]$ (k number of stacked layers) with, therefore, $\tilde{y}^{(i)} = h^{(i)} \circ h^{(i-1)} \circ \dots \circ h^{(0)}(x)$ (parameters $\theta^{(i)}$ are omitted for clarity) and $\tilde{y} = \tilde{y}^{(k)}$, their derivatives can be recursively computed as follows:

$$\begin{aligned} \frac{d\mathcal{L}}{d\theta^{(i)}} &= \frac{d\mathcal{L}}{d\tilde{y}^{(k)}} \frac{d\tilde{y}^{(k)}}{d\theta^{(i)}} = \frac{d\mathcal{L}}{d\tilde{y}^{(k)}} \frac{d\tilde{y}^{(k)}}{d\tilde{y}^{(k-1)}} \frac{d\tilde{y}^{(k-1)}}{d\theta^{(i)}} = \frac{d\mathcal{L}}{d\tilde{y}} \frac{d\tilde{y}^{(k)}}{d\tilde{y}^{(k-1)}} \cdots \frac{d\tilde{y}^{i+1}}{d\tilde{y}^{(i)}} \frac{d\tilde{y}^{(i)}}{d\theta^{(i)}} \\ &= \frac{d\mathcal{L}}{d\tilde{y}} \prod_{j=1}^{k-i} \left(\frac{d\tilde{y}^{(k-j+1)}}{d\tilde{y}^{(k-j)}} \right) \frac{d\tilde{y}^{(i)}}{d\theta^{(i)}}. \end{aligned} \quad (2.3)$$

Considering NNs composed of fully-connected layers $h^{(i)}(\tilde{y}^{(i-1)}; \theta^{(i)}) = f_{ac}^{(i)}(z^{(i)})$ with $z^{(i)} = \tilde{y}^{(i-1)} \cdot W^{(i)} + b^{(i)}$, each final term of the above equation can be conveniently computed as:

$$\begin{aligned} \frac{d\tilde{y}^{(i)}}{d\tilde{y}^{(i-1)}} &= \frac{d(f_{ac}^{(i)}(z^{(i)}))}{d\tilde{y}^{(i-1)}} = \frac{d(f_{ac}^{(i)}(z^{(i)}))}{z^{(i)}} \frac{d(z^{(i)})}{d\tilde{y}^{(i-1)}} = f_{ac}^{(i)'}(z^{(i)}) \frac{d(\tilde{y}^{(i-1)} \cdot W^{(i)} + b^{(i)})}{d\tilde{y}^{(i-1)}} \\ &= (W^{(i)})^\top f_{ac}^{(i)'}(z^{(i)}), \end{aligned} \quad (2.4)$$

$$\begin{aligned} \frac{d\tilde{y}^{(i)}}{d\theta^{(i)}} &= \left\{ \frac{d\tilde{y}^{(i)}}{dW^{(i)}}, \frac{d\tilde{y}^{(i)}}{db^{(i)}} \right\} = \left\{ \frac{d\tilde{y}^{(i)}}{dW^{(i)}}, \frac{d\tilde{y}^{(i)}}{db^{(i)}} \right\} \\ &= \left\{ (\tilde{y}^{(i)})^\top f_{ac}^{(i)'}(z^{(i)}), f_{ac}^{(i)'}(z^{(i)}) \right\}, \end{aligned} \quad (2.5)$$

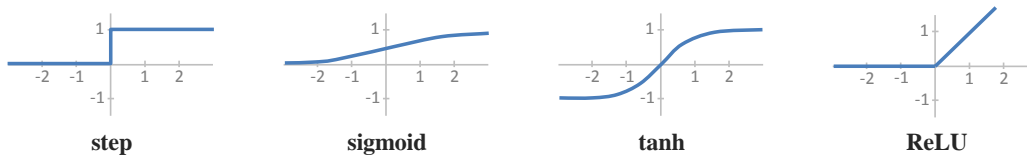


Fig. 2.5: Representation of common activation functions used in NNs.

with $f_{ac}^{(i)'}(z^{(i)})$ the derivative of the activation function $f_{ac}^{(i)}$ w.r.t. its input $z^{(i)}$. This justifies the common usage of easily and continuously derivable functions such as (c.f. Figure 2.5):

- sigmoid $s_L(z) = \frac{1}{1+e^{-z}}$ with $s_L'(z) = s_L(z) - s_L(z)^2$;
- tanh $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ with $\tanh'(z) = 1 - \tanh(z)^2$;
- rectified linear unit $\text{ReLU}(z) = \max(0, z)$ with $\text{ReLU}'(z) = 1$ if $z > 0$ or $\text{ReLU}'(z) = 0$ if $z < 0$;
- *etc.*

Similarly, the backpropagation technique requires the usage of a loss that is derivable w.r.t. to its input \tilde{y} , *i.e.*, in order to compute the term $\frac{d\mathcal{L}}{d\tilde{y}}$. Therefore, typical losses are based on the Manhattan or Euclidean distance between y and \tilde{y} ($\mathcal{L}_p(y, \tilde{y}) = \|y - \tilde{y}\|_p$ with $p = 1$ or 2), or based on their cross-entropy, *etc.*

Training Sequence. The supervised training of a neural network over an available labeled dataset S is an iterative process. Each iteration, the loss is either computed and directly *back-propagated* after a single prediction $h(x; \theta)$ with (x, y) usually randomly picked from S hence the name stochastic gradient descent (SGD); or the loss is first accumulated over a *mini-batch* of n predictions (with $n \leq |S|$) before being back-propagated at once, *e.g.*, as its mean value. In both cases, the sequence of forward and backward propagations through the models to be trained is repeated until convergence, according to Equation 2.2 (a sequence of iterations covering the whole training set S is called an *epoch*).

The integration of the backpropagation technique made it theoretically possible to train *deep* NN models (*i.e.*, NNs with larger numbers of layers stacked vertically)³, but the hardware capability in the eighties was heavily constraining any larger-scale experimentation.

³The term “deep learning” was first coined by Dechter [56] in 1986 and was applied to NNs a decade after [4, 100]. The expression entered popular domain only after 2012 and the breakthrough of “*really-deep*” NNs.

Convolutional neural networks

Inherently, fully-connected neural networks are poorly adapted to tensorial inputs like images, which led to the development of a different family of NNs: the convolutional neural networks (CNNs).

Limitations of Traditional Neural Networks. First of all, due to their dense connectivity, the number $|\theta|$ of parameters a fully-connected NN should tune is proportional to the dimensionality of its input and output vectors. When considering multi-layered networks to process images (*i.e.*, $\mathcal{X} \subset \mathbb{R}^{c \times h \times w}$ with $h \times w$ its spatial dimensions and c its number of channels), the number of parameters can easily explode.

Furthermore, as the neurons in fully-connected layers have indifferently access to all input elements, any spatial relation in the tensors is lost. For instance, considering images again, the networks have no notion of proximity/distance between the pixels, nor notion of depth/channels.

Constraining the Spatial Connectivity of Neurons. Developed in parallel in the eighties [73, 74] and refined over the following decades [142, 243], CNNs offer efficient solutions to these shortcomings. While they work the same way as traditional NNs (feed-forward, backpropagation, *etc.*), changes were brought to their architecture.

Unlike fully-connected networks, neurons in CNNs have limited spatial connectivity, *i.e.*, they only have access to values in the neighboring region of their previous layer. The region (of $c \times k_h \times k_w$ pixels for images) that each neuron is connected to is called its *receptive field*. This limited connectivity not only drastically reduces the number of parameters to train (for each of these neurons, $|\theta| = \dim(W) + \dim(b) = c \times k_h \times k_w + 1$, a number of parameters which is not proportional to the input size anymore), but it also preserves the localization of image features.

Convolutional Layers. CNNs are named after the *convolutional* layers at the core of their architecture. In these layers, the number of parameters is further reduced by sharing the same weights and bias among all neurons linked to the same output channel \bar{c} . These neurons with shared parameters and limited spatial connectivity ($W^{\bar{c}} \in \mathbb{R}^{c \times k_h \times k_w}$, $b^{\bar{c}} \in \mathbb{R}$) can be modeled as a single neuron—called *filter* or *kernel*—sliding over the whole input tensor with a vertical and horizontal *stride* (s_h, s_w) .

At each position (h_i, w_j) that it can take over the input tensor⁴ (considering the stride), this neuron still behaves like traditional ones, linearly combining the values in its receptive field at this position and applying an activation function to the result:

$$\tilde{y}_{h_i, w_j}^{\bar{c}} = f_{ac} \left(b^{\bar{c}} + \sum_{l=0}^{c-1} \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} W_{l, m, n}^{\bar{c}} \cdot x_{l, h_i+m, w_j+n} \right). \quad (2.6)$$

After sliding a kernel over the whole input tensor (optionally padded with p_h rows and p_w columns on each side, e.g., with constant values), $\tilde{y}^{\bar{c}} \in \mathbb{R}^{h_{out} \times w_{out}}$ is obtained, with:

$$h_{out} = \frac{h - k_h + 2p_h}{s_h} + 1, \quad w_{out} = \frac{w - k_w + 2p_w}{s_w} + 1. \quad (2.7)$$

Given a layer composed of c_{out} kernels, its response maps are stacked into $\{\tilde{y}^{\bar{c}}\}_{\bar{c}=1}^{c_{out}} = \tilde{y} \in \mathbb{R}^{c_{out} \times h_{out} \times w_{out}}$, which can then be passed to other layers. The fact that Equation 2.6 can be efficiently performed as a *convolution*⁵ gave the name to the corresponding layers.

Architectures and Properties of CNNs. Convolutional layers are powerful tools for data processing. As mentioned earlier, they can be applied to large input images without suffering from an explosive number of parameters. It is, therefore, possible to stack a larger number of such layers, compensating for their limited spatial connectivity. Indeed, the effective receptive field $\text{ERF}^{(i)}$ of a layer $f^{(i)}$ —i.e., the region in the input image x which affects the activation of its neurons—increases the deeper the layer is, as expressed by the following recursive formula (applicable to the horizontal and vertical dimensions):

$$\text{ERF}^{(i)} = \text{ERF}^{(i-1)} + (k^{(i)} - 1) \prod_{j=1}^{(i-1)} s^{(j)}, \quad (2.8)$$

with $k^{(i)}$ the layer's kernel size ($k_h^{(i)}$ or $k_w^{(i)}$, though mostly square kernels are used in practice) and $s^{(j)}$ the stride of a previous layer $h^{(j)}$ (similarly, commonly $s_h^{(j)} = s_w^{(j)}$). To further increase the effective receptive field of their final layers, CNNs are often also

⁴From now on, to simplify notations, only image-like tensors are considered, i.e., from $\mathcal{X} \subset \mathbb{R}^{c \times h \times w}$.

⁵Actually, the proper mathematical term for this operation is “cross-correlation”, though “convolution” is commonly used in the machine learning community. For all valid positions (h_i, w_j) of ω over x , the cross-correlation of a matrix $x \in \mathbb{R}^{h \times w}$ with a kernel $W \in \mathbb{R}^{k_h \times k_w}$ is:

$$(W * x)_{h_i, w_j} = \sum_{l=0}^{k_h-1} \sum_{m=0}^{k_w-1} W_{l, m} \cdot x_{h_i+l, w_j+m},$$

whereas the mathematical convolution of these elements would be:

$$(W \star x)_{h_i, w_j} = \sum_{l=0}^{k_h-1} \sum_{m=0}^{k_w-1} w_{l, m} \cdot x_{h_i-l, w_j-m}.$$

Note that these operations are, however, quite similar in such a setup, and convolution results can be obtained from the cross-correlation operation by simply *flipping* the filters before. Unless specified differently, to follow the current convention in the field of machine learning, the term “convolution” and the symbol “ \star ” will be used when referring to the operation performed by CNNs.

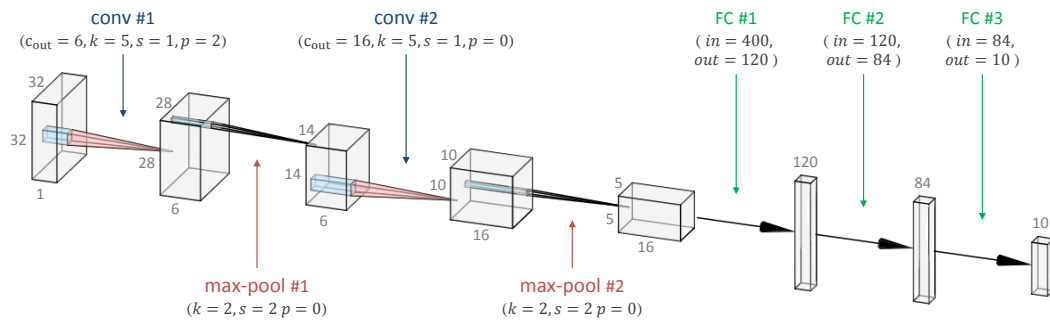


Fig. 2.6: LeNet-5 architecture for hand-written digit recognition [141, 142]. “conv” are convolutional layers, “max-pool” are maxpooling ones, and “FC” are fully-connected ones. Observe how the spatial dimensionality decreases with each layer, leading to the prediction – a vector of 10 probabilities, corresponding to the belief the network has that the input image corresponds to the 10 considered digit classes (figure rendered with the NN-SVG tool by Alexander Lenail [143]).

composed of *pooling* layers. Those layers have the specificity of not having any trainable parameters (their operation is fixed), making them lightweight. Applied over the whole input tensor with strides (s_h, s_w) , *average-pooling* neurons simply return the average value in their $k_h^{(i)} \times k_w^{(i)}$ receptive field (computed over each channel separately), whereas *max-pooling* neurons return the maximal value. Note that the previously introduced fully-connected layers are also commonly used in CNNs, usually as final layers leading to the models’ predictions by linearly interpolating and activating over the features extracted by the convolutional and pooling layers.

Indeed, convolutional layers also have interesting properties related to image processing and feature extraction. With training, each kernel learns to react to a specific local feature wherever it appears in the input tensor (convolutional results are invariant to translation in the image coordinate space). The response map of a kernel over an input tensor can be described as a *feature map*, *i.e.*, an output tensor expressing the positions where the kernel responded to its target feature. The deeper a layer in a CNN, the more abstract the features its neurons will react to are. Typically, the first convolutional layers in CNNs activate for specific lines, color gradients, *etc.*, whereas deeper layers are fed with these feature maps and thus react to combinations of features, like shapes or textures, then facial features, specific objects, *etc.*

CNN Winter and Spring. Despite their numerous advantages, CNNs were still computationally too heavy for the eighties’ hardware capabilities, and researchers favored more lightweight methods such as SVMs. Even though a handful of research teams kept refining and optimizing CNNs and their implementations [100, 189, 276] (*e.g.*, with the famous *LeNet-5* architecture proposed by LeCun et al. [142] for hand-written digit recognition and illustrated in Figure 2.6), the application of these models to real-life problems stalled for more than a decade.

It was a major breakthrough in 2012 which finally gave CNNs their actual prominence, drastically changing their status from underdogs to unchallenged solutions to visual tasks. Indeed, that year, the third edition of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)—an image classification benchmark over the large ImageNet dataset [58] composed of ~14 million images from 1,000 classes—was won by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton and their eight-layer CNN later named *AlexNet* [57, 130]. This victory was significant not only because it embodied the first successful application of CNN to such a large-scale task, but also because *AlexNet* won with a staggeringly low classification error rate of 16%, outshining previous methods stuck at 26%.⁶

All the following editions of ILSVRC and other famous computer vision challenges were won by CNN-based methods, refined and derived to tackle more and more complex tasks. So began the ongoing *deep learning* era of computer vision.

2.2.2 Rationale for the Success of Deep Learning

If neural networks finally took over computer vision after decades of being shunned, it was the result of concomitant developments and change of mindsets in computer and data science.

Concomitant Development of Computer Science

Indeed, besides the tenacity of some researchers who kept refining CNNs the past decades, the main reasons for the resurgence of deep learning are twofold, linked to the expansion of digitalization and computer hardware (as superficially mentioned in the introduction of this thesis, *c.f.* Figure 1.1).

Information Age. With the exponential development of the Internet and digital content, data scientists had soon access to the largest data sources in the history of mankind, the so-called *big data*. By simply indexing the content shared online first by experts, soon followed by an increasingly large proportion of mankind as well as automated devices, data scientists started building the first large-scale media datasets, such as the aforementioned ImageNet [58].

This novel access to such databases was a blessing for *data-hungry* machine learning algorithms like NNs, *i.e.*, algorithms requiring large amounts of data (and often the

⁶The main metric used for ILSVRC is the *top-5* accuracy, *i.e.*, algorithms can return up to 5 predicted classes, and the overall prediction is deemed erroneous if the ground-truth image class does not appear in this set.

corresponding annotations) to be trained. It also created a new, urgent need for such algorithms. As the *cold* data accumulated by companies increased exponentially, so did the demand to convert this data into strategic knowledge.

Hardware Accelerators. Thankfully, concomitantly to the digitalization of society, hardware capability also kept increasing exponentially (*c.f.* the aforementioned Moore’s law [175]). As personal computers multiplied and became faster, a new processing unit was created to meet popular needs for effective 3D computations, *e.g.*, for graphics interfaces and video-games: the graphics processing units (GPUs). GPUs are build to efficiently perform and parallelize matrix operations, *e.g.*, for image manipulation. Though invented in the eighties, they became affordable only with the new millennia.

In 2007, Nvidia, one of the main GPU manufacturers, released the first version of CUDA [229], a programming language for developers to build and run programs on compatible GPUs. OpenCL [178], a similar language, appeared soon after. It did not take long for researchers to harness the GPU acceleration for their algorithms, dividing the training and inference time of CNNs by a significant number (from ~20 to 100 time faster) [130, 189].

With this increasing demand for applied machine learning models and finally the technical mean to meet it, the time was ripe for NN advocates to demonstrate their potential.

Democratization of Model Crafting

Though this explains why CNNs finally became viable solutions, it does not justify their success over previous methods, *e.g.*, combining the extraction of statistical features and the use of regression models like SVMs. In the following paragraphs, reasons for the current CNN hegemony are discussed.

Hierarchical Approach and Modularity. First, an interesting parallel can be drawn between the aforementioned bottom-up approach to computer vision—which has become its main paradigm—and the layered architecture of CNNs themselves. Indeed, these models are composed of layers, each an independent function, which hierarchically extract increasingly complex features. They react to multiple levels of representation, from basic visual features to semantically rich concepts. This pyramidal processing is thus an attractive extension of the robotists’ approach to perception.

Furthermore, the layers (convolutional, pooling, dense, *etc.*), as basic components of CNNs, can be combined and derived into a large variety of architectures, adapted for specific use-cases (*e.g.*, one could edit or replace the final layers of *LeNet-5* shown in Figure 2.6

to tackle instead the recognition of handwritten roman letters). The modular properties of CNNs, along with the development of efficient frameworks to simplify their implementation (*Caffe* [113], *TensorFlow* [1], *PyTorch* [198], *etc.*), led many experts to experiment with CNNs and apply them to new problems, building up their popularity.

Automatized Feature Learning and Transfer. Compared to traditional methods, CNNs also streamlined the recognition process and the approach to new image domains. Traditional feature-based recognition systems are composed of three steps. First, a *keypoint* detector is applied to the data to identify regions of interest. For each region, a *local feature* is computed as a lower-dimensional and robust representation. Finally, local features are concatenated into a *descriptor* vector which is handed to a classification or regression algorithm for recognition [180]. Besides their complexity, such systems require some expertise to be transferred to different applications.

Indeed, as mentioned in Subsection 2.1.2 and explained in a survey by Nanni et al. [180], experts used to predominantly rely on manually designed features (“*hand-crafted*”). Facing new tasks often implied having to manually configure or merge different feature models, which itself implied having prior knowledge on the target domains. Feature engineering was and is still a craft and, as such, it is often tedious to adapt it.

In comparison, neural networks can be considered as end-to-end adaptive solutions. With their trainable kernels convolved over images and feature maps, CNNs learn inductively to identify and extract features suitable for the task at hand. Furthermore, like other machine-learning algorithms, their expertise acquired during training can be transferred to other models, *i.e.*, by sharing the totality or a subset of their tuned parameters.

For instance, a common and often successful practice when setting up a model $h^t(\cdot; \theta^t)$ for a new *target* task $\mathcal{T}^t = \{\mathcal{Y}^t, f^t\}$ consists of reusing the feature-extracting layers of another model $h^s(\cdot; \theta^s)$ successfully trained to tackle a well-covered *source* task $\mathcal{T}^s = \{\mathcal{Y}^s, f^s\}$, and adding on top new layers leading to the desired output format. Mathematically, this means considering $h^s = h_{pred}^s \circ h_{extr}^s$, with h_{extr}^s feature-extracting layers of h^s and h_{pred}^s its final layers processing the features into predictions in \mathcal{Y}^s ; and this means defining h^t as $h^t = h_{pred}^t \circ h_{extr}^s$ with h_{pred}^t new layers to process features into predictions in \mathcal{Y}^t ($\theta_{extr}^s \cap \theta_{pred}^t = \emptyset$ and $\theta_{extr}^s \cup \theta_{pred}^t = \theta^t$). If the source and target tasks \mathcal{T}^s and \mathcal{T}^t are similar (*e.g.*, if their input distributions are almost identical), only θ_{pred}^t need to be optimized for the new task, greatly reducing the complexity of the training process (*i.e.*, *freezing* θ_{extr}^s). Even if the tasks differ significantly, transfer learning can benefit the new model. The transferred parameters θ_{extr}^s can be optimized (*i.e.*, *fine-tuned*) along with θ_{pred}^t , often reaching better optima than training with θ^t randomly initialized [76, 299].

Their ability to cover complex recognition tasks from end to end, building upon previous models in a modular fashion, has been key to the success of CNNs. Research based on these models, as well as their integration into industrial systems, keep gaining impetus, with direct applications in a large panel of domains (autonomous driving, industrial automation, medical screening, augmented reality, *etc.*)

2.2.3 Acknowledgment of CNN Limitations

The dominion of CNN-based methods over computer vision is such nowadays that more and more researchers are voicing their concerns, alarmed that deep learning may be seen as an end rather than a mean. CNNs indeed have intrinsic limitations which need to be acknowledged.

“Black Box” Argument

One of the long-standing arguments against ANNs is their “black-box” nature [19, 221], impairing their integration into sensitive decision-making systems.

Lack of Interpretability. Indeed, a common criticism of machine learning models is their *lack of interpretability*. Because these models perform over high dimensionality and rely on overwhelming numbers of parameters, the logical connection between their predictions and the observed features leading to them is usually obscured and hard to trace back. This lack of visibility/interpretability of the end-to-end procedure led to the term “black box” [166, 221].

For some critical applications like autonomous driving or medical screening, the inability to provide human-readable explanations to the resulting predictions has been hindering the integration of CNN-based systems, for safety/judicial reasons. While more and more works are focusing on “opening the black box” (*e.g.*, combining CNNs with natural-language question-answering systems [285], studying and highlighting which features methods are *attentive* to [90, 238], *etc.*), we are still far from interpretable AI [91].

Sensitivity to Perturbations. This lack of interpretability is also correlated to the inability of deep learning methods to represent causal relationships or to directly perform on *symbols* (as opposed to *vectors*), *e.g.*, to interact with structured knowledge databases [93, 159].

As typical CNNs are unable to integrate abstract knowledge, they have to purely rely on the empirical observations provided during their training phase and can fail to generalize to new out-of-distribution samples (*e.g.*, to generalize to uncommon poses of target elements,

to different light settings or sensor quality, *etc.*) [6, 159]. Multiple research projects have demonstrated the sensitivity of CNNs to *adversarial attacks* (*i.e.*, to slight corruptions of visual inputs invisible to humans but leading to the complete failure of advanced artificial models) [5, 214].

Deep learning solutions are thus subordinate to their pseudo-empirical knowledge and, therefore, conditioned by any bias the training data may have. While the rise of very-large—and thus diverse-enough—datasets circumvented this problem and allowed CNNs to shine, this inherent flaw is still present and resurfaces for applications with limited access to training data.

Dependency on Data Quantity and Quality

Another common argument raised against deep learning models is, therefore, their dependency on large amounts of training samples, and the data collection and pre-processing issues arising from it.

Data-Hungry Solutions Neural networks not only require large and diverse amounts of training images to reach robust optima, most models also need the corresponding ground-truth labels, being taught in a supervised manner.

Annotating datasets is typically a tedious and costly process. For complex tasks, it can take several minutes to annotate a single image (*e.g.*, to create pixel-precise labels for semantic segmentation), and some annotations may have to be validated/corrected by experts (*e.g.*, when labeling medical images) [205].

While transfer learning and derived methods can relax the need for large application-specific datasets by capitalizing on previous tasks, data still represents the life and blood of deep-learning solutions. As we will formalize in Section 2.3, scarcity of training data still deeply impair the performance of CNNs in many real-life scenarios.

Upstream Feature and Model Crafting For many, deep learning thus only moved the task of feature crafting upstream. Instead of engineering features relevant for the task at hand, scientists are now focusing much effort on pre-processing the available training data before passing it to their models (*balancing* labels, *augmenting* samples, extrapolating new ones, *etc.*). Another part of their effort is dedicated to *network engineering*, *i.e.*, the development and refinement of CNN architectures able to best capture the distinctive and informative features of the dataset.

Developed too early for their time, deep neural networks now completely revolutionized computer vision thanks to the concomitant rise of hardware accelerators and *big* data. However, as CNNs are more and more refined by academicians and adopted by industries, their intrinsic constraints have come under increasing scrutiny. Many companies with the ambition to leverage their power to automate complex recognition tasks come to realize that the data that they intended to feed the models is either not comprehensive enough or require expert processing to yield robust systems.

2.3 Data Scarcity and its Ramifications

The field of machine learning has already an extensive literature evaluating the data requirements of learning algorithms, as well as the effects of distribution divergence between training and target domains. After expanding on these theories, the following section also provides some considerations specific to data quantity and relevance in computer vision.

2.3.1 Learnability and Data Dependencies

For any intelligent system, learning how to perform a specific task requires relevant data samples, in large enough quantity, to distill effective knowledge. If the provided examples are not relevant enough to the task, or if they do not fully cover all its aspects, one cannot expect the acquired knowledge to *generalize* properly. In this subsection, key machine learning concepts are summarized to formalize the needs for rich and relevant data to train algorithms.

Sample Complexity of Machine Learning Models

In machine learning, the *sample complexity* represents an estimation of the minimum number of training samples a model would need to effectively learn the objective function. Such an estimation is paramount in order to choose adequate models, taking into consideration the trade-off between the algorithms' complexity and the amount of available training data.

Risk Convergence and Dataset Size. Subsection 2.1.3 formalized how machine learning models are selected/trained with the goal to minimize the expected risk \mathcal{R} for their task (*c.f.* Equation 2.1) even though, in practice, they can only be evaluated over the empirical risk $\hat{\mathcal{R}}$ (*c.f.* Equation 2.2). Therefore, the key purpose of machine learning is to obtain hypotheses which minimize this empirical risk over the training data $S = \{(x_i, y_i)\}_{i=1}^m$ (where x_i sampled *i.i.d.* from $P(X)$ and $y_i = f(x_i)$ with f true labeling function) and

which generalize well when applied to new elements from the same distribution, *i.e.*, also minimizing the expected risk ($\hat{\mathcal{R}}$ is thus also named *in-sample risk*, whereas \mathcal{R} can be named *out-of-sample risk* or *generalization risk*).

Intuitively, we can expect that the larger the training set (*i.e.*, the larger the number m of samples it has), the more likely the empirical risk of a given hypothesis $h \in \mathcal{H}$ will be close to the expected one, because the more likely the available samples will be representative of the true distribution $P(X)$. This intuition can be formalized through the weak law of large numbers:

Theorem 2.1 (Weak Law of Large Numbers). *Let $\{a_i\}_{i=1}^m$ be a set of i.i.d. samples of a random variable A following a distribution $P(A)$. Then for all $\epsilon > 0$:*

$$\lim_{m \rightarrow \infty} P\left(\left|E_{a \sim P(A)}[a] - \frac{1}{m} \sum_{i=1}^m a_i\right| \geq \epsilon\right) = 0.$$

Referring to Equations 2.1 and 2.2, it can thus be written that for a fixed hypothesis $h_\theta \in \mathcal{H}$:

$$\lim_{m \rightarrow \infty} P\left(\left|\mathcal{R}(h_\theta) - \hat{\mathcal{R}}(h_\theta)\right| \geq \epsilon\right) = 0. \quad (2.9)$$

Furthermore, according to the Chebyshev's inequality [232], for a fixed hypothesis $h_\theta \in \mathcal{H}$, a given dataset $S = \{(x_i, y_i)\}_{i=1}^m$ with x_i sampled *i.i.d.* from $P(X)$ and $y_i = f(x_i)$, and for all $\epsilon > 0$:

$$P\left(\left|\mathcal{R}(h_\theta) - \hat{\mathcal{R}}(h_\theta)\right| \geq \epsilon\right) \leq \frac{\text{var}(\{x_i\}_{i=1}^m)}{\epsilon^2} = \frac{\text{var}(X)}{m\epsilon^2} \quad \text{with var the variance.} \quad (2.10)$$

This equation confirms the earlier intuition, showing that the probability that $\hat{\mathcal{R}}$ calculated over S diverges from true \mathcal{R} by $\epsilon > 0$ at most decreases as the sample size augments. It also demonstrates the impact of the data variance. The larger the variance a considered population has, the larger the sampled dataset should be to probabilistically ensure that a specific hypothesis would perform nearly as well (or as bad) out-of-sample as it does in-sample [2, 3].

Concentration inequalities can further be applied to bind the divergence between a set of randomly sampled variables and their true distribution. One of the most famous is Hoeffding's inequality [103].

Theorem 2.2 (Hoeffding's Inequality [103]). *Let $\{a_i\}_{i=1}^m$ be a set of i.i.d. samples of a random variable A following a distribution $P(A)$, and let $\forall i, a_{\min} \leq a_i \leq a_{\max}$. Then for all $\epsilon > 0$:*

$$P\left(\left|E_{a \sim P(A)}[a] - \frac{1}{m} \sum_{i=1}^m a_i\right| \geq \epsilon\right) \leq 2e^{\frac{-2m\epsilon^2}{(a_{\max} - a_{\min})^2}}.$$

Assuming that the loss \mathcal{L} returns normalized values between 0 and 1 ($a_{min} = 0, a_{max} = 1$), replacing the expected and empirical risks in Hoeffding's inequality provides a stricter upper-bound to the probability that the distance between $\hat{\mathcal{R}}$ and \mathcal{R} exceeds ϵ : the probability decays exponentially as the number of samples in the available dataset increases [3, 227].

Learnability Theory. While these inequalities provide interesting upper-bounds to the generalization ability of a given $h_\theta \in \mathcal{H}$ (i.e., to the probability of $\mathcal{R}(h_\theta)$ being close to $\hat{\mathcal{R}}(h_\theta)$), they only apply to this fixed h_θ and do not indicate how to select/train a hypothesis which would minimize $\hat{\mathcal{R}}$ and \mathcal{R} (i.e., so that $\hat{\mathcal{R}} \approx \mathcal{R} \approx 0$).

In training settings, h_θ can be considered as a random variable depending on the sampled training set S of size m . The goal is, therefore, to obtain a proper *uniform convergence bound*, i.e., to prove that for the task, the probability (noted δ) is small that $\exists h \in \mathcal{H}$ such that $\mathcal{R}(h)$ differs significantly from $\hat{\mathcal{R}}(h)$ (i.e., by more than a value $\epsilon > 0$) [2, 23, 227]:

$$\mathbb{P}\left(\sup_{h \in \mathcal{H}} |\mathcal{R}(h) - \hat{\mathcal{R}}(h)| \geq \epsilon\right) \leq \delta, \quad (2.11)$$

where sup represents the supremum. For finite hypothesis spaces (i.e., $|\mathcal{H}|$ cardinality of \mathcal{H} countable), the union bound inequality can be applied. By then injecting results of Theorem 2.2 (supposing the values of \mathcal{L} normalized), the left term of this equation can be rewritten as:

$$\begin{aligned} \mathbb{P}\left(\sup_{h \in \mathcal{H}} |\mathcal{R}(h) - \hat{\mathcal{R}}(h)| \geq \epsilon\right) &= \mathbb{P}\left(\bigcup_{h \in \mathcal{H}} |\mathcal{R}(h) - \hat{\mathcal{R}}(h)| \geq \epsilon\right) \\ &\leq \sum_{h \in \mathcal{H}} \mathbb{P}\left(|\mathcal{R}(h) - \hat{\mathcal{R}}(h)| \geq \epsilon\right) \\ &\leq 2|\mathcal{H}|e^{-2m\epsilon^2}. \end{aligned} \quad (2.12)$$

Considering the notation in Equation 2.11 and solving Equation 2.12 for ϵ , the following theorem defines the learnability of discriminative functions for \mathcal{H} finite, with an upper-bound function of the size of the hypothesis space and of the training dataset:

Theorem 2.3 (Generalization Bound for \mathcal{H} finite [2, 23, 227]). *Let $\mathcal{H} = \{h_\theta : \mathcal{X} \rightarrow \mathcal{Y} \mid \theta \in \Theta\}$ be a finite hypothesis set, and let $S = \{(x_i, y_i)\}_{i=1}^m$ be the available training data with x_i sampled i.i.d. according to $\mathbb{P}(X)$ and $y_i = f(x_i)$ where f is the objective labeling function. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for all hypotheses $h_\theta \in \mathcal{H}$:*

$$\mathcal{R}(h_\theta) \leq \hat{\mathcal{R}}(h_\theta) + \sqrt{\frac{1}{2m} \left(\ln |\mathcal{H}| + \ln \frac{2}{\delta} \right)}.$$

While this inequality elegantly combines expected and empirical risks, size/complexity of the model space, and size of the training set, it becomes useless as $|\mathcal{H}|$ increases. For instance, for CNNs with millions of real-valued parameters, the number of different models is near infinite, and so becomes the right-hand term of the above inequality.

Generalization Bounds for Near-Infinite Hypothesis Spaces. When considering infinite or near-infinite hypothesis spaces, an inequality stricter than the union bound is thus needed in order to get rid of the problematic $|\mathcal{H}|$ term. A key intuition is to take into account the possible similarities among numerous hypotheses in \mathcal{H} and, therefore, the overlaps among the probabilistic events that $|\mathcal{R}(h) - \hat{\mathcal{R}}(h)| \geq \epsilon$ (while the union bound supposes no overlapping).

Following those lines, the seminal work of Vapnik and Chervonenkis [45, 274] and the later study by Blumer *et al.* [23] offer a tighter measure of the effective complexity/size of hypothesis spaces. Considering the tasks of binary classification⁷—*i.e.*, for $\mathcal{H} = \{h_\theta : \mathcal{X} \rightarrow \mathcal{Y} \mid \theta \in \Theta\}$ with $\mathcal{Y} = \{0, 1\}$ —they demonstrated that the tasks are learnable if and only if the Vapnik-Chervonenkis (VC) dimension of \mathcal{H} —noted $vc(\mathcal{H})$ —is finite. This dimension $vc(\mathcal{H})$ is defined as the largest value of $m = |S|$ such that any configuration of labels on this sampled set is consistent with the results of at least one $h \in \mathcal{H}$.

As explained in a theoretical survey by Galanti *et al.* [76], this analysis relies on the *growth* function defined for the considered hypothesis space \mathcal{H} , which expresses the maximum number of different labels that \mathcal{H} can return over m samples [45, 274]:

$$\Pi_{\mathcal{H}}(m) = \max_{S_x \sim \mathcal{P}(X)^m} \left| \{h(x_1), \dots, h(x_m) : h \in \mathcal{H}\} \right| \quad \text{with } S_x = \{x_1, \dots, x_m\}. \quad (2.13)$$

Lemma 2.1 (Perles–Sauer–Shelah’s Lemma [231, 237]). *If the VC dimension of a hypothesis space \mathcal{H} is finite, then its growth function $\Pi_{\mathcal{H}}(m)$ is polynomial in m and for $m > vc(\mathcal{H})$:*

$$\Pi_{\mathcal{H}}(m) \leq \left(\frac{em}{vc(\mathcal{H})} \right)^{vc(\mathcal{H})}.$$

Putting everything together, the aforementioned theorem of Vapnik and Chervonenkis can be expressed as follows.

Theorem 2.4 (Vapnik and Chervonenkis’ Theorem [23, 45, 274]). *Let $\mathcal{H} = \{h_\theta : \mathcal{X} \rightarrow \{0, 1\} \mid \theta \in \Theta\}$ be a finite hypothesis set, let $S = \{(x_i, y_i)\}_{i=1}^m$ be the available training data with x_i sampled i.i.d. according to $\mathcal{P}(X)$ and y_i the corresponding true labels, and let*

⁷Other recent studies have been proposing different measures of the complexity/size of hypothesis spaces for more complex tasks such as multi-class classification [17, 182, 295] and regression [210]. However, whichever complexity measure is chosen, the resulting inequalities binding the difference between empirical and expected risks keeps a form similar to the one presented here with the Vapnik–Chervonenkis measure [227].

the empirical and expected risks $\hat{\mathcal{R}}$ and \mathcal{R} be computed from a loss \mathcal{L} with values in $[0, 1]$. Then:

$$\mathbb{P}\left(\sup_{h_\theta \in \mathcal{H}} \left| \mathcal{R}(h_\theta) - \hat{\mathcal{R}}(h_\theta) \right| \geq \epsilon\right) \leq 4\Pi_{\mathcal{H}}(2m)e^{-\frac{1}{8}m\epsilon^2}.$$

Therefore, if $\Pi_{\mathcal{H}}$ is polynomial, the right term of the inequality converges toward 0 as m increases (as the rest of the term decays exponentially with m), meaning that the expected and empirical risks uniformly converge toward each other [2, 76]. Theorem 2.4 can be rewritten similarly to Theorem 2.3, *i.e.*, for any $\delta > 0$ and for all $h_\theta \in \mathcal{H}$, with probability at least $1 - \delta$:

$$\begin{aligned} \mathcal{R}(h_\theta) &\leq \hat{\mathcal{R}}(h_\theta) + 2\sqrt{\frac{2}{m}\left(\ln \Pi_{\mathcal{H}}(2m) + \ln \frac{4}{\delta}\right)} \\ &\leq \hat{\mathcal{R}}(h_\theta) + 2\sqrt{\frac{2}{m}\left[\text{vc}(\mathcal{H})\left(\ln \frac{2m}{\text{vc}(\mathcal{H})} + 1\right) + \ln \frac{4}{\delta}\right]}. \end{aligned}$$

Theorem 2.5 (VC Theorem – Dataset Size [11, 23, 45, 274]). *Considering the same settings as Theorem 2.4, for any $\epsilon > 0$ and $\delta > 0$, if S contains m samples such that*

$$m = O\left(\frac{1}{\epsilon^2}\left(\text{vc}(\mathcal{H}) + \ln \frac{1}{\delta}\right)\right),$$

then for all $h_\theta \in \mathcal{H}$, with probability at least $1 - \delta$:

$$\left| \mathcal{R}(h_\theta) - \hat{\mathcal{R}}(h_\theta) \right| \leq \epsilon.$$

Overall, the more complex/large \mathcal{H} , the higher its VC dimension $\text{vc}(\mathcal{H})$ is (indicating how likely its hypotheses can generalize) but also the larger the training dataset S should be to ensure generalization [2].

CNN Expressiveness and Data Hungriness. Considering NNs, their VC-dimension ranges from $O(|\Theta| \ln |\Theta|)$ to $O(d^2|\Theta|^2)$, with $|\Theta|$ their number of parameters and d their depth, depending on the considered architectures (type of activation functions, use of *dropout* [249] and other regularization methods, *etc.*) [12, 95].

Though ample discussions are still trying to explain why deep learning models, with their millions of parameters, generalize better than what the inequality of Theorem 2.4 is expressing [303], the aforementioned theorems still provide an important formalism to the *hunger for data* characterizing deep learning. Recent empirical studies like the one performed by Sun *et al.* [256] confirm the “unreasonable effectiveness of data in deep learning era”, presenting a logarithmic increase in performance of CNNs on visual tasks w.r.t. the size of the training dataset.

The formal analysis drawn from statistical machine learning and these empirical works justify the need to balance the current race towards increasingly complex CNNs with the sampling and gathering of increasingly larger and more varied datasets, in order to unlock the full potential of deep learning.

Domain Gaps and their Impact

However, these previous conclusions rely on a significant assumption: it is considered that the samples available for training are drawn from the same distribution $P(X)$ as the samples the models will be applied to afterward.

In practice, this assumption usually does not hold. In the following paragraphs, typical shifts between distributions are introduced, and a new generalization bound accounting for the divergence is presented from the literature.

From now on, the marginal probability distribution which the input samples in the *source* training dataset $S = \{(x_i, y_i)\}_{i=1}^{m_S}$ are drawn *i.i.d.* from is named $P_S^X = P_S(X)$, and the probability distribution of the input data from the *target* domain is $P_T^X = P_T(X)$. In some cases, we will also assume access to an unlabeled and usually scarce dataset from the target domain at training time: $T = \{x_i\}_{i=1}^{m_T}$, with x_i sampled *i.i.d.* according to P_T^X .

Domain Shifts. A variety of reasons can be behind the discrepancies between the distribution of available training samples and the distribution of target data. In computer vision like other fields, these domain *shifts* can be caused by a biased experimental sampling protocol, leading to a training dataset non-representative of the target domain.

For example and as illustrated in Figure 2.7, training images can be captured by cameras with a different image quality than the target sensors, or captured under different lighting and clutter conditions [53], *e.g.*, by capturing *out-of-assembly* images of manufactured object (*i.e.*, each object separately on a clean background) though recognition should be then done in more cluttered conditions. If, these distribution discrepancies are observed (*i.e.*, $P_S(X, Y) \neq P_T(X, Y)$), but it is assumed that the true labeling function f is the same for the training and testing data (*i.e.*, $f(x) = y$ for x sampled *i.i.d.* from P_S^X or P_T^X ; *i.e.*, $P_S(Y|X) = P_T(Y|X)$) then the models are facing a *covariance drift* [53, 255].

Class imbalance or *prior shift* can also plague training datasets, *i.e.*, when the true label distributions $P_S(Y)$ and $P_T(Y)$ are different but not the related conditional distributions $P_S(Y|X)$ and $P_T(Y|X)$ (*e.g.*, when labels are under or over-represented in the available samples) [53, 255].

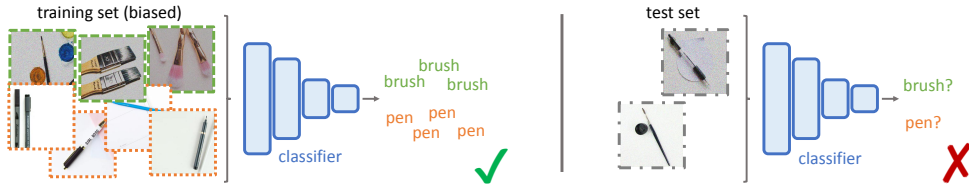


Fig. 2.7: Example of a recognition model trained on a biased dataset. Suppose a company wants to develop a CNN to discriminate pictures of *pens* and pictures of *brushes*. However, the training images for each class were gathered by two entities which did not agree on a precise acquisition protocol beforehand (*e.g.*, opting for a same camera model or setting similar lighting conditions). As a result, the sampled *brush* images are distinctly darker and noisier than the *pen* ones. Since NNs are trained to use any visual cues for their tasks, the CNN taught on this dataset may end up relying on these obvious yet biased visual discrepancies to classify the objects, instead of purely focusing on the concept representations (*e.g.*, the shape of the objects, their texture, *etc.*). In production, this model would fare poorly, unable to rely on these biases anymore.

As explained by Kouw [129], *concept shift* happens when the posteriors are different in the source domain and in the target one, even though the data distributions are the same.

Note that in practice, machine-learning models have to face a combination of these various domain shifts with differences in both the marginal and conditional distributions [53, 129]. It is also important to note that, often, these domain drifts can be deliberate, exemplified by transfer learning. One may want to apply a recognition system trained for our *source* domain to another, *e.g.*, when training data cannot be collected for that target domain.

Effects on Generalization. Whichever the drift(s), models trained in one domain usually perform poorly if applied to a different one. As mentioned in Subsection 2.2.3, this can be intuitively deduced for CNNs. Trained on a specific dataset, they learn to extract and rely on specific visual features (*e.g.*, discriminative patterns and shades in training images). If later test images do not exhibit the same cues, the models won't be able to infer properly.

Therefore, when evaluating the generalization capacity of models, it is crucial not only to factor in the *size* and *variance* of training datasets, but also their *relevance* to the target tasks. Significant effort has been expanded toward accounting for domains shifts in the generalization bounds [15, 16, 81, 125, 129, 132, 215, 255, 277], as well as toward developing models more robust to these shifts [27, 42, 53, 77, 78, 192, 265, 272].

In domain adaptation, the goal is indeed to minimize the expected risk \mathcal{R}_T w.r.t. the *target* domain while only being able to measure the empirical risk $\hat{\mathcal{R}}_S$ over the training *source* dataset S :

$$\mathcal{R}_T(h_\theta) = \mathbf{E}_{x \sim P_T^X} [\mathcal{L}(f(x), h_\theta(x))] , \quad (2.14)$$

$$\hat{\mathcal{R}}_S(h_\theta) = \frac{1}{m_S} \sum_{i=1}^{m_S} \mathcal{L}(y_i, h_\theta(x_i)) \quad \text{over } S = \{(x_i, y_i)\}_{i=1}^{m_S} . \quad (2.15)$$

Similarly, for convenience, the expected risk w.r.t. the *source* domain is defined as $\mathcal{R}_S = \mathbf{E}_{x \sim P_S^X} [\mathcal{L}(f(x), h_\theta(x))]$. In this setting, still considering \mathcal{Y} binary for simplicity (though once again, the study is extendable to more complex tasks [78]), Ben-David *et al.* provided the following definitions and theorem to bind the generalization capacity of hypotheses for domain adaptation.

Definition 2.1 (Ideal Joint Hypothesis [15, 16]). The *ideal joint hypothesis* $h_\theta^* \in \mathcal{H}$ is the hypothesis minimizing the combined expected risks over the source and target domains, *i.e.*:

$$h_\theta^* = \operatorname{argmin}_{h_\theta \in \mathcal{H}} \mathcal{R}_S(h_\theta) + \mathcal{R}_T(h_\theta) .$$

Ben-David *et al.* note the combined loss of h_θ^* as $\lambda_{\mathcal{H}} = \mathcal{R}_S(h_\theta^*) + \mathcal{R}_T(h_\theta^*)$.

Definition 2.2 ($\mathcal{H}\Delta\mathcal{H}$ -Divergence [16]). Given the hypothesis space $\mathcal{H} = \{h_\theta : \mathcal{X} \rightarrow \{0, 1\} \mid \theta \in \Theta\}$ and two marginal distributions P_S^X and P_T^X over \mathcal{X} , the $\mathcal{H}\Delta\mathcal{H}$ -divergence of these distributions is defined as:

$$d_{\mathcal{H}\Delta\mathcal{H}}(P_S^X, P_T^X) = 2 \sup_{(h, h') \in \mathcal{H}^2} |\mathcal{R}_{P_T^X}(h, h') - \mathcal{R}_{P_S^X}(h, h')| ,$$

with $\mathcal{R}_{P_S^X}(h, h') = \mathbf{E}_{x \sim P_S^X} [\mathcal{L}(h(x), h'(x))]$ the *expected source disagreement* of two hypotheses h and h' , and $\mathcal{R}_{P_T^X}(h, h')$ similarly defined as their *expected disagreement* over the *target* domain. $d_{\mathcal{H}\Delta\mathcal{H}}(P_S^X, P_T^X)$ represents the capacity of hypotheses in \mathcal{H} to discriminate elements from the two marginal distributions.

Lemma 2.2 ($\mathcal{H}\Delta\mathcal{H}$ -Divergence Bound [16]). Let $\mathcal{H} = \{h_\theta : \mathcal{X} \rightarrow \{0, 1\} \mid \theta \in \Theta\}$ be a hypothesis set of finite VC dimension $vc(\mathcal{H})$ and let $S \sim (P_S^X)^m$ (available source labels are not represented here) and $T \sim (P_T^X)^m$ be two input datasets of m i.i.d. samples. The symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ is defined as $\mathcal{H}\Delta\mathcal{H} = \{g : \mathcal{X} \rightarrow \{0, 1\} \mid g(x) = h(x) \oplus h'(x), \text{ for } (h, h') \in \mathcal{H}^2\}$ (with \oplus the XOR operator. It represents the sets of disagreements between any couple of hypotheses and has thus at most a VC dimension $vc(\mathcal{H}\Delta\mathcal{H}) \leq 2vc(\mathcal{H})$. Considering the results of Theorem 2.4, then for any $\delta > 0$ and $\forall h_\theta \in \mathcal{H}$, with probability at least $1 - \delta$:

$$d_{\mathcal{H}\Delta\mathcal{H}}(P_S^X, P_T^X) \leq \hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S, T) + 2 \sqrt{\frac{2}{m} \left[2vc(\mathcal{H}) \left(\ln \frac{m}{vc(\mathcal{H})} + 1 \right) + \ln \frac{4}{\delta} \right]} ,$$

with $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S, T)$ the empirical $\mathcal{H}\Delta\mathcal{H}$ -divergence between the two domains w.r.t. \mathcal{H} .

Theorem 2.6 (Generalization Bound for Domain Adaptation [15, 16, 125]). Let $\mathcal{H} = \{h_\theta : \mathcal{X} \rightarrow \{0, 1\} \mid \theta \in \Theta\}$ be a hypothesis set of finite VC dimension $vc(\mathcal{H})$, let $S = \{(x_i, y_i)\}_{i=1}^{m_S}$ be the available training data from the source domain with x_i sampled

i.i.d. according to P_S^X and y_i the corresponding true labels, and let $S' \sim (P_S^X)^m$ and $T \sim (P_T^X)^m$ be two unlabeled datasets of m i.i.d. samples (e.g., S' subset of input elements in S when $m \leq m_S$). Let the risks be computed from a loss function \mathcal{L} with values in $[0, 1]$. Then for any $\delta > 0$ and $\forall h_\theta \in \mathcal{H}$, with probability at least $1 - \delta$:

$$\begin{aligned} \mathcal{R}_T(h_\theta) &\leq \mathcal{R}_S(h_\theta) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(P_S^X, P_T^X) + \lambda_{\mathcal{H}} \\ &\leq \mathcal{R}_S(h_\theta) + \frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S', T) + \sqrt{\frac{2}{m} \left[2 \text{vc}(\mathcal{H}) \left(\ln \frac{m}{\text{vc}(\mathcal{H})} + 1 \right) + \ln \frac{4}{\delta} \right]} + \lambda_{\mathcal{H}}. \end{aligned}$$

This inequality can further be expanded by replacing $\mathcal{R}_S(h_\theta)$ with its upper-bound relative to $\hat{\mathcal{R}}_S(h_\theta)$ according to Theorem 2.4.

Interpretation of the Generalization Bound for Domain Adaptation. Theorem 2.6 elegantly binds the expected risk of a model over the target domain when trained over a dataset drawn from a source domain and formalizes some intuitions. Analyzing each term of its inequality, a model $h_\theta \in \mathcal{H}$ can generalize to the target domain only if:

- The model minimizes the expected source risk $\mathcal{R}_S(h_\theta)$, *i.e.*, if it minimizes the empirical source risk $\hat{\mathcal{R}}_S(h_\theta)$ over a training dataset S large and varied enough;
- The divergence between the two domains is small enough, at least with regard to \mathcal{H} and the inability of its hypotheses to discriminate the two distributions (*c.f.* $\frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(P_S^X, P_T^X)$);
- There exist hypotheses in \mathcal{H} which can perform well on both domains, *i.e.*, the tasks on the source and target domains are similar and can be solved by hypotheses in \mathcal{H} accurately enough (*c.f.* $\lambda_{\mathcal{H}}$).

Assuming that the hypothesis space is adequately chosen for the task (a common hypothesis in domain adaptation), the last term is considered small. Therefore, Theorem 2.6 expresses the trade-off between the accuracy of a model, the size/complexity of its hypothesis family \mathcal{H} , the inability of models in \mathcal{H} to distinguish source and target samples, and the richness and relevance of the training dataset [15, 16, 78, 81].

In this study, the hypothesis families are considered as fixed/provided (*e.g.*, common CNN architectures are used out-of-the-box). This leaves the term w.r.t. the domain divergence to minimize, by exploring solutions to bridge the gap between training/source and test/target data in terms of absolute discrepancies or with regard to the models' ability to tell them apart.

2.3.2 Dealing with Data Scarcity in Industrial Computer Vision

As hinted in the introductory Chapter 1, despite the increasing availability of large image datasets, a variety of specific computer vision applications still suffer from data scarcity. Indeed, as deep learning solutions are being adopted by more and more companies, their target applications are becoming more increasingly specific (*e.g.*, recognition of manufactured components, medical image analysis, *etc.*) and thus diverging too much from the domains covered by pre-available large image databases. This justifies the growing need for data sampling and domain adaptation solutions.

This subsection expands further on this problematic, elaborating on the heterogeneity of visual data and its consequences on machine learning models, as well as detailing the usage of synthetic data to compensate for the lack of relevant samples.

Visual Supports of Recognition and their Availability

Visual data have high dimensionality and a myriad of factors can affect their content. An infinite number of images can be derived from a single concept (*e.g.*, a specific object or individual). Advanced algorithms, themselves characterized by high sample complexity, are thus required to capture this variability inherent to image or video-based recognition.

The following paragraphs further develop on how the high variability of target visual domains, combined with the data requirements for the convergence of designated models, sent scientists on a quest to collect large yet relevant training datasets.

Heterogeneous of Visual Data and Sensors. With the proliferation of *smartphones*, the expansion of closed-circuit television (CCTV) networks, the development of specialized devices for industries, *etc.*, cameras have become completely ubiquitous. With these sensors easily accessible to the public and to professionals, a wide range of behaviors and applications are being revolutionized, such as social interactions, artistic expression, visual monitoring, augmented reality (AR), *etc.*

However, not all cameras have the same properties (field of view, aperture, sensor size and precision, color balance, sensibility to specific wavelengths in the visible spectrum or beyond, *etc.*) or the same target applications (commodity camera versus specialized sensors, *e.g.*, for medical or industrial imaging). Besides sensor properties, a broad variety of factors can affect the quality and content of images captured by a color camera: the layout of the captured scene (projection of the subject in the image, clutter, occlusion, background), the



Fig. 2.8: Pairs of color and depth images, from T-LESS [102] and LineMOD [99] datasets.

lighting conditions (projected shadows, under/over-exposition, lens flare, *etc.*), the *texture* of captured elements (color variability due to specular and diffuse properties, mutability, *e.g.*, from wear-and-tear, *etc.*), and so on.

As color images of the same *concepts* can vary tremendously, some researchers have been investigating alternative sensing devices for visual recognition, *i.e.*, devices more robust to some of these changes. In the industry especially, many experts have been adopting *depth sensors* [63, 138, 245] for instance, *i.e.*, sensors measuring the distance to elements in their field of view (*c.f.* Figure 2.8)⁸. Providing range images more robust to some aforementioned visual variations (*e.g.*, to color and lighting changes), these sensors had known a concomitant development leading to their popularization. As further detailed in Chapter 3, low-cost commodity depth sensors—such as the *Microsoft Kinect V1* and *V2*—are being nowadays applied to a wide panel of applications [70, 135, 158, 184, 223, 239].

Still, a majority of companies made the strategic choice to keep color images as visual input modality of their recognition systems, given the ubiquity of color cameras compared to other more specialized devices. For instance, in contrast to commodity cameras, only a handful of smartphones or tablets on the market are integrating depth sensors (*e.g.*, the now-defunct *Google's Tango* brand, or *Apple's* new *iPhone* devices incorporating the *TrueDepth* sensor), and these depth sensors usually have limited capability.

Nevertheless, taking into account other specialized sensors (*e.g.*, medical scanners), the list of devices which can be linked to computer vision systems is long. Domain adaptation methods are, therefore, crucial in our domain, in order to train models despite common discrepancies between source and target domains, both in terms of image content and sensing quality.

From Generic to Specialized Datasets. To tackle this issue of data availability and relevance, part of the community has been in parallel focusing on gathering increasingly large generic datasets (*e.g.*, *ImageNet* [58] and, more recently, *Microsoft COCO* [150] or *Google Open-Images* [134]), collecting images from heterogeneous domains (*e.g.*, from different sources, cameras, format, *etc.*). As mentioned in Subsection 2.2.2, *big data* has provided the metaphorical fuel leading to the booming of deep learning, and the *bigger* and

⁸Most depth sensors return image structures or files where the value of each pixel represents the measured distance to the sensed surface (*e.g.*, in millimeters or according to a device-specific scale). For visualization, these values are commonly scaled to display the depth scans as grayscale images (*i.e.*, the brighter, the further). This representation is used all along this thesis.

more varied the available data becomes, the more recognition models can robustly learn [15, 16, 256]. Even if these large datasets can quite diverge from the target task domains, their availability often reduces the quantity of task-relevant data required to train models, *e.g.*, by benefiting from transfer learning [76, 299].

However, as mentioned both in Chapter 1 and in the introduction of this subsection, an increasing number of applications are targeting *niche* domains, *i.e.*, highly-specific image domains in terms of content (*e.g.*, industrial components, body scans, *etc.*) and/or visual modality (*e.g.*, range imaging, thermography, *etc.*). While transfer learning could be considered, the target domains of these specific applications are often too alien compared to the common large-scale datasets. Transferring pre-trained weights for CNN models could benefit their first, more generic layers (*c.f.* Section 2.2.2), but the domain gaps are overall too big for the entire models to generalize to the target distribution.

Furthermore, for such use-cases, it is often challenging, if not impossible, to gather samples to properly train or fine-tune the recognition models (*e.g.*, when the target objects are not produced yet or are only available at some remote manufacturing location). To deal with this scarcity or non-existence of relevant data, some efforts have been expanded toward gathering and sharing datasets for some of these specialized use-cases (*e.g.*, the *IKEA* [148] or *T-LESS* [102] datasets for manufactured elements, the *MIMIC-CXR* [84, 114] or *fastMRI* [302] datasets for medical imaging, *etc.*). However, by definition, such datasets do not generalize well to other specific applications.

Still, an alternative solution to obtain relevant-enough images—when capturing proper ones is not possible—has been exploited by researchers for years; a solution which furthermore suits most industrial applications: the generation of synthetic images.

Synthetic Datasets and Realism Gap

In these final paragraphs of the background chapter, the generation and usage of synthetic data in computer vision are detailed, and the current limitations of using such images as training surrogate are developed.

Creation of Synthetic Datasets. In computer vision, *synthetic data* are computer-generated images, as opposed to *real pictures* captured by sensing devices. While some synthetic images are purely rendered from procedural or physics engines modeling some realistic phenomena (*e.g.*, synthetic images of meteorological or astronomical events, like the scientifically-rendered black hole in the Hollywood movie *Interstellar* [111]), most images are based on *3D models*.

A 3D model is a computer file containing a representation of the geometry of an object or scene (*e.g.*, storing the 3D positions of its key surface points), and sometimes of some of its physical or visual properties (*e.g.*, the texture of its surfaces). Computer graphics algorithms, nowadays combined into powerful systems named *3D engines*, can be used to render images from these 3D models; by defining a simulated camera, projecting the 3D coordinates of the scene into the image plane of this camera, ignoring occluded elements, interpolating the color of each pixel based on the texture and lighting properties, *etc.*

Thanks to the lucrative entertainment industry, computer graphics have come a long way the past decades, and rendering engines can nowadays generate highly realistic images from 3D models (*e.g.*, for video games, 3D animated movies, special effects, *etc.*). Scientists were quick to grasp the potential for computer vision.

Databases of 3D models started appearing [38, 244, 248, 293], and scripts were developed and shared to render huge datasets of pseudo-realistic images from these models (changing the viewpoints, intrinsic properties of the virtual cameras, lighting conditions, scene setups, *etc.*). Having full control over the scene content and capture process, all kinds of annotations can also be generated along the synthetic images to train recognition systems (*e.g.*, keeping track of the position of the objects in the scene for the training of object-detection algorithms, using the 3D engine to render binary masks for semantic segmentation, *etc.*). Additionally, 3D models and game engines can also be used to create interactive simulation environments, which can be used to teach complex skills to any sort of intelligent agents. For example, in the aerospace domain, pilots and engineers are commonly trained to perform key tasks in virtual environments, as cheap and safe surrogates to real scenarios [108]. An increasing number of such simulation platforms are being dedicated to the training of autonomous agents like self-driving cars, robots, *etc.* [29, 264, 290].

Synthetic Data for Industrial Recognition Systems. As a result, the literature exploring the usage of synthetic data to train recognition system started abounding the past decade [37, 70, 80, 147, 162, 216, 233, 250, 286], and more and more companies started considering this as a solution to their problem of data scarcity. Indeed, especially in the industry, 3D models of target objects are often pre-available, *e.g.*, developed for the manufacturing process like 3D computer-aided design (CAD) models.

However, most industrial CAD models are not conceived to be visually realistic, but rather to be used as blueprints for the manufacturing process, to perform virtual quality tests, *etc.* (*c.f.* Figure 1.3). To develop highly-realistic 3D models like those used in the entertainment industry, hours should be spent gathering visual information and modeling it (*i.e.*, analyzing and compiling the textures of the real objects, refining the virtual geometry, *etc.*). Such a task requires access to the original objects and is extremely costly – even more so than directly building a dataset of real images of these objects.

Furthermore, even in the few instances that real-looking CAD models are available, rendered images may diverge from real ones for a variety of reasons: the appearance of real objects may be evolving during their service life (*e.g.*, because of deterioration, dirt, *etc.*), the rest of the environment may be different or unknown (leading to discrepancies in terms of image clutter and background), the rendering engine cannot comprehensively simulate the visual result of target sensing devices, *etc.*

As a result, it is often impossible to generate realistic synthetic data to train industrial recognition algorithms. Like other domain discrepancies, this *realism gap* between computer-generated training data and the real target images harms the performance of the models (*c.f.* Subsection 2.3.1).

Overcoming the Realism Gap. Currently, a lot of effort is being devoted to tackling the realism gap for computer vision. Solutions can be divided into two categories depending on if they address the problem from a data-centrist or a model-centrist angle.

In the former case, some methods have been proposed to more easily capture visual properties of target objects, to build more precise 3D data models [14, 82]. Other research works have been focusing on improving the quality or variety of rendering algorithms, *e.g.*, to simulate specific sensing devices [7, 88].

Other experts have instead adopted concepts from domain adaptation, and adapted those to bridging the realism gap for the training of visual recognition methods, *i.e.*, so that the knowledge these algorithms acquired from synthetic environments can be better transferred to real situations [25, 27, 78, 265].

In this thesis, the realism gap is approached according to these two paradigms. In that regard, the following chapters further expand on the respective state-of-the-art landscapes and propose novel solutions dedicated to the training of robust industrial systems.

Realistic Depth Sensor Simulation

” *More data beats clever algorithms, but better data beats more data.*

— **Peter Norvig**

(Google’s Director of Research, 2009.)

Recent progress in computer vision has been dominated by deep neural networks trained over large amounts of labeled data. As mentioned in previous chapters, collecting such datasets is however a tedious, often impossible task; hence a surge in approaches relying solely on synthetic data as surrogates for their training. However, for range imaging especially, discrepancies with real depth scans still noticeably affect the end performance, as research focusing on simulating these sensors is still in its early stage.

In this chapter, an end-to-end framework, named *DepthSynth* [207], is proposed to comprehensively simulate the mechanisms of depth sensors and to model vital factors, in order to generate realistic depth data from three-dimensional (3D) models (*c.f.* Figure 3.1). As demonstrated through qualitative and quantitative evaluations, this pipeline covers a wider range of sensors and achieves more realistic results compared to previous methods. Optimized to generate synthetic data in near-real-time (*i.e.*, enabling trainings over datasets of larger size m , *c.f.* Theorems 2.4 and 2.6) and closer to the target real domain (*i.e.*, reducing the divergence $d_{\mathcal{H}\Delta\mathcal{H}}$, *c.f.* Theorem 2.6), the proposed solution thus enhances the performance of recognition systems trained on its data, as confirmed empirically.

The motivation behind this first work is detailed in Section 3.1. A survey of pertinent studies is then provided to the reader in Section 3.2, and the proposed framework is then presented in Section 3.3, detailing each step. Section 3.4 elaborates on the experimental protocol; first comparing the sensing errors induced by the pipeline to experimental data and theoretical models, then demonstrating its usefulness by applying it to the training of recognition algorithms for various visual tasks. Finally, Section 3.5 concludes the chapter with some insightful discussions.

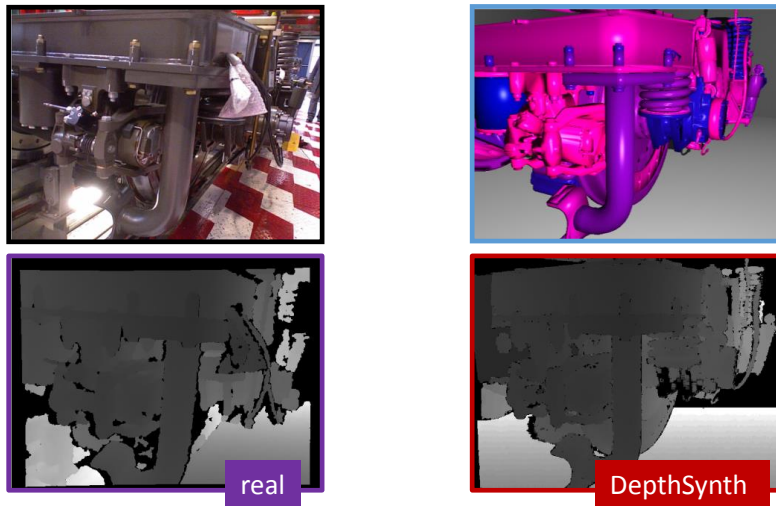


Fig. 3.1: Real color and depth pictures, opposed to the corresponding CAD model and simulated depth image.

3.1 Motivation

This thesis has been articulated around several industrial projects investigating the usage of depth sensors to tackle specific visual tasks requiring accurate predictions. These industrial use-cases shared similar constraints which motivated the development of a simulation tool for depth sensors. The following section expands on these constraints and motivations.

3.1.1 Rise of Depth-based Computer Vision

As explained in this subsection, the adoption of depth sensors for industrial recognition projects has been a global trend in past years, hence a growing demand for theoretical evaluation and technical solutions related to these devices.

Advent of Low-Cost Depth Sensors

Range-imaging devices have been around for decades, relying on a variety of techniques such as stereo-vision, interferometry, triangulation, photogrammetry, *etc.* [133, 138]. More recent solutions are based on:

- *Structured-light, i.e.*, the process of projecting a known coded pattern onto a scene and capturing the visual result with a camera, to measure the distortion of the pattern and infer the depth accordingly (taking into account the distance between pattern projector and camera for triangulation);

- *Time-of-flight (ToF)*, *i.e.*, the radar technique consisting of measuring the time that the beams of light projected by the sensor take to reach a surface in the scene and to be reflected back to the device; in order to deduce the distance.

However, as explained by Landau in his thesis [138], any of these sensing methods require advanced electronic and optic components, and the price of depth sensors used to be in the four digits until the end of the 2000s.

When the cost of some of these components finally dropped, commodity sensors like the *Microsoft Kinect V1* and *Intel RealSense* quickly appeared on the market. Originally released in 2010 as a gaming accessory for motion sensing, the *Kinect* was soon adopted by researchers for a large panel of technical use-cases related to 3D computer vision.

Depth Data for Robust 3D Computer Vision

Understanding the 3D layout and semantic content of real-world scenes captured in two-dimensional (2D) images has been a classic computer vision problem for decades [43, 83, 307]. Therefore, the release of low-cost depth sensors—specifically structured light cameras—resulted in a significant paradigm shift. What in the past revolved around the interpretation of raw pixels in 2D projections has now become the analysis of real-valued depth (2.5D) data.

This has drastically increased the scope of practical applications ranging from recovering 3D geometry of complex surfaces [158, 184] to real-time recognition of human actions [163, 239], inspiring research in automatic object detection [70, 223, 242, 247], classification [136, 164, 223, 244, 246, 247] and pose estimation [70, 148, 234].

3.1.2 Call for Realistic Depth Simulation Tools

Though depth data has enabled novel applications, it suffers—like other visual modalities—from representational complexity. The following paragraphs expand on the specificities of data scarcity and realism gap w.r.t. depth scans, justifying the need for advanced simulation tooling.

Adoption of Synthetic Data in Computer Vision

As detailed in Chapter 2, while access to target real data is paramount to the training of robust machine-learning systems, it is common for real-life use-cases not to meet this

condition. Therefore, a large number of the aforementioned recent studies decompose their tasks into matching acquired depth images of real-world objects to synthetic ones rendered from a database of pre-existing 3D models [37, 70, 148, 223, 244, 246, 247]. With no theoretical upper bound to the number of synthetic images one can generate to either train complex models for classification [164, 234, 253, 291] or fill large databases for retrieval tasks [80, 281], research continues to gain impetus in this direction.

Despite the simplicity of the above flavor of approaches, their performance is often constrained by the *realism gap* (i.e., the discrepancy between real and synthetic data, mentioned in Section 2.3) or by the lack of *variability* (limited configurability) of their rendering process. As a workaround, some approaches fine-tune their systems on a small set of real scans [286]; but in many cases, real data is too scarce to bridge the discrepancy gap. Other methods try instead to post-process the real images to clear some of their noise, making them more similar to synthetic data but losing details in the process [291] which can be crucial for tasks such as pose estimation or fine-grained classification.

Realism Gap for Depth Data

Structured-light and ToF depth sensors are inherently more robust to visual variations caused by the lighting conditions of target scenes and the material properties of the objects in, so one would assume that rendering realistic synthetic depth data should be more straightforward than rendering color ones. But simulation tools for “red-green-blue” (RGB) image rendering have been polished for decades by researchers in computer graphics and by entertainment companies, which have had little interest in the simulation of depth sensors.

Jump-started after the recent dissemination of low-cost devices, research toward the generation of realistic depth scans is, therefore, still at an infant stage compared to other visual modalities. A key motivation of the proposed method is thus to reduce the discrepancies between the simplistic synthetic data and the noisy scans from commodity sensors, in order to facilitate the development of depth-based computer vision.

3.2 Related Work

With the popular advocacy of 2.5D/3D sensor for vision applications nowadays, depth information is the support of active research within computer vision. This section emphasizes on recent approaches which employ synthetic scans, and presents previous methods to generate such 2.5D data from 3D computer-aided design (CAD) models.

3.2.1 Recognition Algorithms and Synthetic Data

This first subsection delivers a brief overview of recent methods taking advantage of 3D models and synthetic data to learn visual features. Among these solutions, additional details are provided for the deep-learning method used in this chapter to demonstrate the effects of simulated data on its training.

CNN-based Recognition from Scarce Data

As detailed in Chapter 2, crafting features to efficiently detect objects, discriminate them, evaluate their poses, *etc.* has long been a tedious task for computer vision researchers. With the rise of machine learning algorithms, these existing models have been complemented [83, 234, 250], before being almost fully replaced by statistically-learned representations. Multiple recent approaches based on deep convolutional neural networks (CNNs) unequivocally outshone previous methods [164, 253, 254, 291], taking advantage of growing image datasets (such as *ImageNet* [58]) for their extensive training.

As a matter of fact, and as mentioned previously, collecting and accurately labeling large amounts of real data is an extremely tedious task, especially when 3D poses are considered for ground truth. In order to tackle this limitation, and concomitantly with the emergence of 3D model databases, renewed efforts [216, 254] were put into the synthetic extension of image or depth scan datasets, by applying various deformations and noise to the original pictures or by rendering images from missing viewpoints. These augmented datasets were then used to train more flexible estimators.

Among other deep learning-based methods for class and pose retrieval recently proposed [164, 253, 291], Wu *et al.* *3D Shapenets* [291] and Su *et al.* *Render-for-CNN* [254] methods are two great examples of a second trend: using the *ModelNet* [291] and *ShapeNet* [38] 3D model datasets they put together, they let their networks learn features from this purely synthetic data, achieving consistent results in object registration, next-best-view prediction, or pose estimation.

Triplet CNN as Exemplary Recognition Method

Diving further into the problem of depth-based instance classification and pose estimation (ICPE) chosen to illustrate the present work, Wohlhart and Lepetit [286] proposed a scalable feature learning approach addressing a two-degree-of-freedom pose estimation problem. Though they used real images to train their main network modality, their study showed promising results using synthetic data without structured noise simulation. This method

has been extended by Zakharov et al. [300], a student (now colleague and friend) whose Master's thesis I co-supervised.

According to this method, a CNN is trained to extract relevant features from an image $x \in \mathcal{X}$ and encode them into a low-dimensional feature vector $y_{inter} \in \mathcal{Y}_{inter}$ with $\mathcal{Y} \subset \mathbb{R}^d$ and d low dimensionality (e.g., mapping the images to a descriptor space where object instances and their poses are well separated). The feature vectors y_{inter} representing known images can then be stored along with their labels $y \in \mathcal{Y}$ (e.g., the class and pose of the main object in each picture), so that given a new image, the CNN can be used to extract its corresponding vector and compare it with the stored ones for recognition.

Therefore, to learn discriminative features, Wohlhart *et al.* method applies the following loss to the training of the convolutional network noted $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}_{inter}$ with θ its learnable parameters:

$$\mathcal{L}(\theta) = \mathcal{L}_{tri}(\theta) + \mathcal{L}_{pair}(\theta) + \lambda \|\theta\|_2^2, \quad (3.1)$$

where \mathcal{L}_{tri} is a *triplet* loss function, \mathcal{L}_{pair} a *pairwise* one, and λ an optional regularization factor (the regularization term is to prevent over-fitting). This loss is thus computed over batches composed of triplets and pairs of samples.

A triplet is defined as (x_b, x_+, x_-) , with x_b an input image, from the training dataset S , used as binding anchor representing a specific class (c_b) and pose (e.g., defined by a quaternion q_b), x_+ a *positive* image similar to x_b in terms of label (similar class $c_+ = c_b$ and/or pose $q_+ \approx q_b$), and x_- a *negative* sample with dissimilar content (different class c_- and/or pose q_-). The triplet loss function is, therefore, designed to force the network to extract features similar for x_b and x_+ , and dissimilar for x_b and x_- :

$$\mathcal{L}_{tri}(\theta) = \sum_{(x_b, x_+, x_-) \in S^3} \max \left(0, 1 - \frac{\|h_\theta(x_b) - h_\theta(x_-)\|_2^2}{\|h_\theta(x_b) - h_\theta(x_+)\|_2^2 + \epsilon} \right)$$

$$\text{where } \epsilon = \begin{cases} 2 \arccos(|q_b \cdot q_+|) & \text{if } c_b = c_+, \\ n & \text{else, for } n > \pi, \end{cases} \quad (3.2)$$

with ϵ is the margin setting the minimum ratio for the distance between similar and dissimilar pairs of samples. This task-related dynamic margin (here defined for the training of a model for instance classification and pose estimation) was introduced by Zakharov et al. [300].

On the other hand, pairs (x_b, x'_b) are composed of an anchor sample x_b and a similar one x'_b with the label but, for instance, with a perturbation in terms of image noise or lighting (if all training images are synthetic) or from a different image domain (e.g., if the training dataset also contains real images). The pairwise loss is thus defined to enforce proximity between

descriptors of samples with different visual cues but similar labels (this can, therefore, be considered as an unsupervised domain adaptation scheme):

$$\mathcal{L}_{pair}(\theta) = \sum_{(x_b, x'_b) \in S^2} \|h_\theta(x_b) - h_\theta(x'_b)\|_2^2. \quad (3.3)$$

For recognition applications, once trained, the method h_θ is used to compute the feature vectors of a subset of S . These vectors are then used as keys to index the labels of samples, to form a feature-descriptor database S_{db} . Recognition is done on test data by using the trained network to compute the descriptor of each provided image and then by applying a nearest-neighbor search algorithm to find its closest descriptor in S_{db} .

In this chapter, this *triplet* CNN method is extended to recognizing 3D pose with six degrees of freedom (6-DOF) (as done by Zakharov et al. [300]) and is fed only with realistic synthetic images from *DepthSynth*. As demonstrated in the following sections, this leads to significantly higher flexibility and scalability of the system, as well as a more seamless application to real-world use-cases.

3.2.2 Generation of Synthetic Images

Though mostly applicable to color images, a variety of research projects have been exploring solutions to generate synthetic images from 3D models, in quantity and quality satisfying to the training of machine-learning algorithms.

Toward Relevant Synthetic Images for Machine Learning

Early research along the direction of rendering synthetic ranging images involves the work of Marr and Nishihara [160] and Brooks et al. [32], wherein search based on 3D representations are introduced. More recently, Rozantsev *et al.* presented a thorough method for generating synthetic images from 3D models [223]. Instead of focusing on making them look similar to real data for an empirical eye, they worked on a similarity metric based on the features extracted during the machine training. However, their model is tightly bound to properties impairing regular cameras (*e.g.*, lighting and motion blur), which cannot be applied to depth sensors.

Su et al. [254] worked concurrently on a similar pipeline, optimizing a synthetic red-green-blue (RGB) image renderer for the training of CNNs. While working on finding the best compromise between quality and scalability, they notice the ability CNNs have to *cheat* at learning from too simplistic images (*e.g.*, by using the constant lighting to deduce the

models poses, or by relying too much on contours for pictures rendered without background, *etc.*). Their pipeline has thus been divided into three steps: the rendering from 3D models, using random lighting parameters; the alpha composition with background images sampled from the *SUN* dataset [294]; and randomized cropping. By outperforming state-of-the-art pose estimation methods with their own one trained on synthetic images, they demonstrated the benefits that such pipelines can bring to computer vision. Inspired by this work applied to color images, *DepthSynth* is composed of similar steps to generate its images

Simulation Tools for Depth Sensors

As mentioned among the reasons behind this work, only a handful of simulation tools for depth sensors have been developed so far. One of the most recent and realistic, the solution proposed by Landau *et al.* [138, 139] reproduces the *Microsoft Kinect*'s behavior by simulating the infrared capture and stereo-matching process. While inspired by their latter step, the solution detailed in this chapter is based on a less empirical, more exhaustive and generic model for the simulated projection and capture of pattern(s).

Similar simulation processes were also developed to reproduce the results of ToF sensors [121, 203]. If this chapter mostly focuses on single-shot and multi-shot structured-light sensors, *DepthSynth*'s genericity allows it to simulate ToF sensors as well, by using a subset of its operations (discarding the baseline distance within the device, defining a simpler projector with phase shift, *etc.*). Such a subset is then comparable to the method developed by Keller and Kolb [121].

For the sake of completeness, tools such as *BlenSor* [88] or *pcl::simulation* [7, 67] should also be mentioned. However, such simulators were implemented to help testing vision applications and rely on simplistic modeling of the sensors, *e.g.*, ignoring reflectance effects or using fractal noise to approximate sensing errors.

3.3 Methodology: Simulation of 2.5D Sensors

The following section details the proposed simulation method for the generation of realistic depth images, based on a study of the actual devices.

3.3.1 Study of Structured-Light Depth Sensors

To realistically generate synthetic depth data, one should first understand the mechanisms behind their generation, and how some of these mechanisms can cause the various kinds of noise observed in scans from real structured-light sensors.

Sensing Mechanisms

Structured-light depth sensors work as follows:

1. A light emitter projects a predefined pattern onto the scene;
2. A camera sensitive to the emitter's wavelength(s) captures an image of the scene with the pattern projected onto;
3. Original and captured pattern images are processed by a stereo-matching algorithm which infers the depth based on the discrepancies between the two, for each pixel;
4. Optionally, some post-processing operations are performed to compensate for sensing errors.

This process is illustrated in Figure 3.2 and further detailed in the following paragraphs.

Pattern Projection and Capture. The first two steps consisting of pattern projection and capture are key to the quality of the estimated depth. First, for the triangulation process to even work, the pattern emitter and the receptor should be separated by a known *baseline* distance d_{base} . It is, therefore, common for depth devices to have the projector and camera mounted together in a common structure to fix this distance. Furthermore, to simplify the tasks of depth estimation algorithms in step 3, it is also common to have their image planes aligned and to have the baseline parallel to the horizontal axis of the common image plane (as represented in Figure 3.2).

Projected patterns vary from one depth sensor to another. Some devices rely on stripped patterns, others on dot patterns, or on continuous ones, *etc.* Patterns can also be multicolored, or be emitted in the IR domain (as *Microsoft Kinect* and *Occipital Structure* do for instance, with the advantage of making the procedure invisible to the human eye and robust to some lighting conditions). Patterns are typically predefined, *i.e.*, pre-coded so that their structure can be robustly captured (*e.g.*, by being heavily contrasted) and can facilitate the depth estimation process (step 3) which require to find correspondences between the

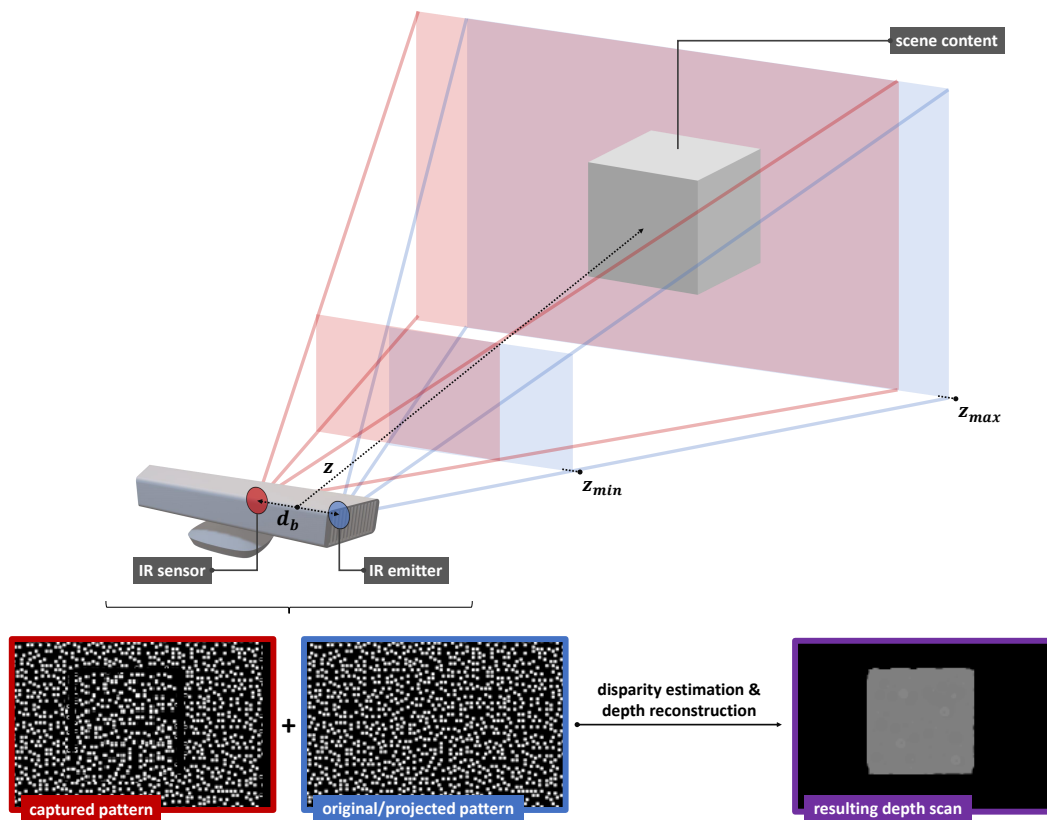


Fig. 3.2: Illustration of a structured-light sensor using an infrared (IR) dot pattern (credits to Pierre Yves P. for the *Microsoft Kinect 3D model*¹ used to render this figure).

original pattern image and the captured reflection (*e.g.*, by opting for projected content with non-repetitive patterns).

Note that some devices, called *multi-shot* depth sensors, rely on a sequence of projections and captures to generate a single depth image (*i.e.*, repeating steps 1 and 2 multiple times, possibly with different patterns, before combining all the captured information to reconstruct a depth image in step 3).

Stereo-Matching and Depth Reconstruction. Relying on the principles of stereo vision, the captured reflection of the pattern is then matched with the reference image, in order to extract the depth information from their *disparity* map. Both images are here used as the stereo stimuli, with these two virtual eyes (the projector and the camera) being separated by d_{base} .

The disparity map represents the distortion in the 2D image plane that a pattern underwent by being projected onto a scene, with each disparity value expressing the displacement of a pattern element (when the emitter and receiver are aligned as previously mentioned, the

¹<https://3dwarehouse.sketchup.com/model/32ab2192d875d85e58aeac7d536d442b/Kinect-sensor>

displacement is then only horizontal/epipolar). Various computer vision methods are used to compute disparity maps from image pairs, *i.e.*, to find correspondences from one image to another in order to measure the displacements (usually performing image block by image block, also known as *block-matching*) [101, 118, 128].

For each pixel, the depth value z is then a direct function of the horizontal disparity d_{disp} :

$$z = f_{px} \cdot \frac{d_{base}}{d_{disp}} \quad (3.4)$$

where f_{px} is the focal length of the receiver (in pixels). Because disparity maps can be noisy or missing data (as explained in the following subsection), some devices include additional methods to smooth the resulting depth maps or interpolate missing values (*e.g.*, using hole-filling/region-growing methods), *c.f.* step 4.

Noise Sources

As I joined my first Siemens industrial project relying on structured-light depth sensors, colleagues had already started analyzing these devices and compiling a list of the different kinds of noise impairing them, along with their sources and characteristics. Extended further, this study highlights how each step of the sensing process introduces its own artifacts [207].

During the initial step of projection and capture of the pattern(s), noise can be induced by the lighting and material properties of the surfaces (*e.g.*, too low or strong reflection of the pattern can prevent its capture), by the composition of the scene (*e.g.*, the density per unit area of the patterns drops quadratically with increasing distance causing axial noise, non-uniformity at edges causing lateral noise, and objects obstructing the path of the emitter, of the camera or both causing shadow noise), or by the sensor structure itself (*e.g.*, its low spatial resolution or the warping of the pattern by the lenses cause structural noise). Note that ToF sensors also share many of the listed noise types. One can observe that several types of noise are related to lighting and surface material properties (axial and lateral noise, capture errors due to too low/high specularity of surfaces, structural noise), impacting the projection and capture of the light beams.

Further errors and approximations are then introduced during the block-matching and hole-filling operations—such as structural noise caused by the disparity-to-depth transform, band noise caused by windowing effect during block correlation, or growing step size as depth increases during quantization.

Tab. 3.1: Comparison of *BlenSor* [88], Landau’s pipeline [138, 139] and *DepthSynth* w.r.t. sensor noise types.

Type of Noise	BlenSor [88]	Landau’s [138, 139]	DepthSynth
Axial and Lateral Noise	Yes	Yes	Yes
Specular Surface	Yes	No	Yes
Non-specular Surface	No	No	Yes
Structural Noise	No	Partial	Yes
Lens Distortion and Effects	No	No	Yes
Quantization Step Noise	No	Yes	Yes
Motion and Rolling Shutter	No	No	Yes
Shadow	No	Partial	Yes

This aforementioned list of noise types deduced from the sensing mechanisms is later used to qualitatively assess the quality of the proposed pipeline, as shown in Table 3.1 or Figures 3.6-3.7.

3.3.2 End-to-End Simulation of Depth Sensors

The proposed end-to-end pipeline for low-latency generation of realistic depth images from 3D CAD data covers various types of 3D/2.5D sensors including single-shot/multi-shot structured light sensors, as well as ToF sensors (relatively simpler than structured-light ones to simulate, using a sub-set of the pipeline’s components *e.g.*, *i.i.d.* per-pixel noise based on distance and object surface material, *etc.*). From this point, emphasis is however put on single-shot sensors (*e.g.*, *Microsoft Kinect*, *Occipital Structure* and *Xtion Pro Live*), given their popularity among the research community.

This proposed pipeline can be defined as a sequence of procedures directly inspired by the underlying mechanisms of the target sensors (*c.f.* Subsection 3.3.1; *i.e.*, from pattern projection and capture, followed by pre-processing and depth reconstruction using the acquired image and original pattern, to post-processing (as illustrated in Figure 3.3).

Pattern Projection and Capture

In the first part of the presented pipeline, a 3D rendering platform is extended to reproduce the realistic pattern projection and capture mechanism. Thanks to an extensive set of parameters, this platform is able to behave like a wide panel of depth sensors. Indeed, any kind of pattern can first be provided as an image asset for the projection, in order to adapt to the single-shot—or multi-shot—depth sensing device one wants to simulate. Moreover, the intrinsic and extrinsic parameters of the camera and projector are configurable.

An optional procedure, suggested by Siemens colleagues Wu *et al.*, covers both the full calibration of real structured light sensors and the reconstruction of their projected pattern

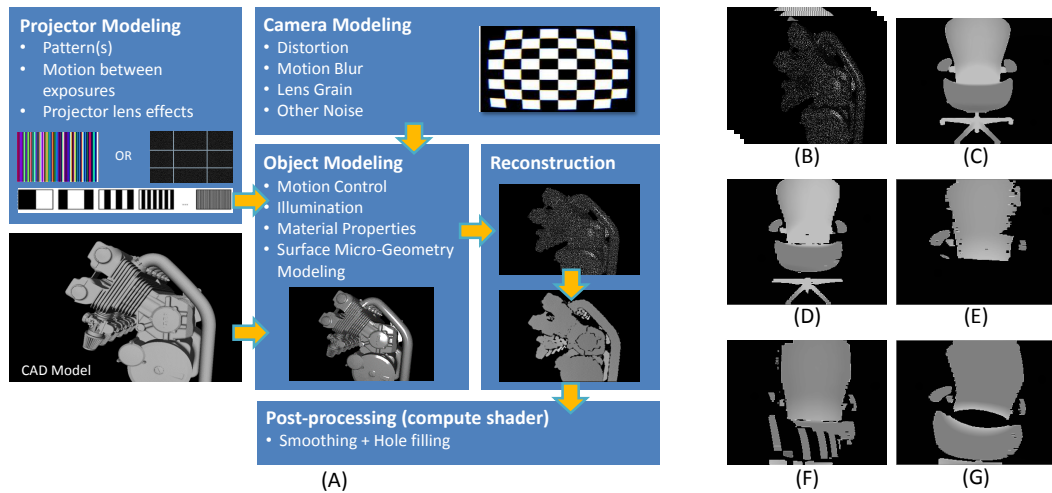


Fig. 3.3: DepthSynth pipeline and results for the simulation of multi-shot depth sensors. (A) Pipeline representation. (B) Rendering of projected patterns under realistic lighting and surface materials. (C) Ideal depth data. (D) DepthSynth generated data without motion or ambient light; (E) with strong ambient light; (F) with motion between exposures (5 cm/s constant speed); (G) with rolling shutter effect (10 cm/s constant speed).

with the help of an extra camera. Once the original pattern obtained, the simulation pipeline automatically generates a binary version of it, followed by other different ones of different scales later used as references in the block matching procedure according to the image resolution of the camera.

Once obtained, these parameters can be handed to the 3D platform to initialize the simulation. The 3D model of the target object or scene must then be provided, along with its material(s). Even though not all 3D models come with realistic textures, the quality of the synthetic results highly depends on such characteristics, especially on their specularity definition.

Indeed, given also a list of viewpoints, the computer graphics platform performs each pattern capture and projection, simulating realistic illumination sources and shadows, taking into account surface and material characteristics. In addition to the object model, the 3D scene is thus populated with:

- A virtual spotlight projector, using the desired high-resolution pattern as light cookie (*i.e.*, as a mask placed over the virtual light source);
- A camera model, set up with the intrinsic and extrinsic parameters of the real sensor, separated from the projector by the provided baseline distance in the horizontal plane of the simulated device;
- Optional light sources, to simulate the effect of environmental illuminations;

- Other optional 3D models (*e.g.*, ground, occluding objects, *etc.*), *e.g.*, to add realistic background or clutter.

By defining a virtual light projector using the provided pattern(s) and a virtual camera with the proper optical characteristics, the exact light projection/capture performed by the actual devices is reproduced, obtaining out of the 3D engine a captured image with the chosen resolution, similar to the intermediate output of depth sensors (*e.g.*, the IR captured patterns from Microsoft Kinect or Occipital Structure devices).

Pre-processing of Pattern Captures

This intermediate result, captured in real-time by the virtual camera, is then pre-processed (fed into a *compute shader* layer), in order to get closer to the original quality, impinged by imaging sensor noise. In this module, noise effects are added, including radial and tangential lens distortion, lens scratch and grain, motion blur, and independent and identically distributed random noise.

Stereo-matching

As explained in Subsection 3.3.1 (step 3 of the sensing procedure), the rendered picture is then matched with its reference pattern to obtain the disparity map.

Here, the disparity map is computed by applying a block-matching process using small sum of absolute differences (SAD) windows to find the correspondences [128], sliding the window along the epipolar line. As a function of the displacement (u, v) (or only of $d_{disp,i}$ if the image planes of the emitter and camera are aligned). the SAD value for the pixel location (h_i, w_j) in the captured image is expressed as:

$$f_{h_i, w_j}^{SAD}(u, v) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} |x_{h_i+m, w_j+n}^c - x_{h_i+u+m, w_j+v+n}^o|, \quad (3.5)$$

where k is the window size, x^c the image captured by the camera, and x^o the original pattern image. The matched location on the pattern image can then be obtained as follows:

$$(u_m, v_m) = \underset{(u,v)}{\operatorname{argmin}} f_{h_i, w_j}^{SAD}(u, v). \quad (3.6)$$

The disparity value d_{disp} can be computed by:

$$d_{disp}(h_i, w_j) = \begin{cases} u_m - h_i & \text{for horizontal stereo,} \\ v_m - w_j & \text{for vertical stereo.} \end{cases} \quad (3.7)$$

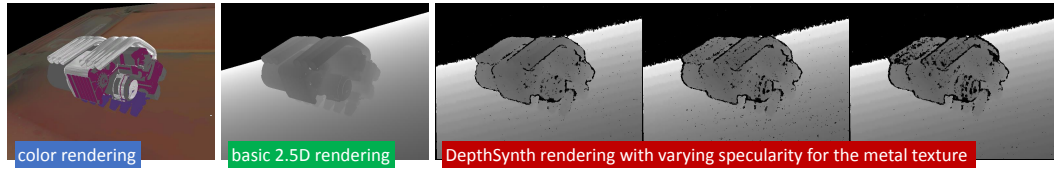


Fig. 3.4: Effects of material specularity on the simulation. The specularity of the metallic material (e.g., for the tubes of the motor) has been increased between each of the three simulated depth scans, impacting their perceived quality as expected from real sensors.

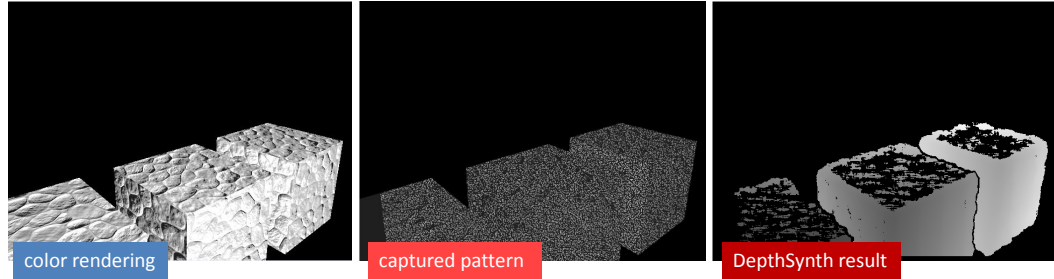


Fig. 3.5: Effects of surface conditions on the simulation, applying a textured normal map to the target cube models.

Based on pixel offsets, each disparity value is, therefore, an integer. To reduce quantization, refinement is performed by interpolating between the closest matching block and its neighbors, achieving a sub-pixel accuracy. Given the direct relation between z and d_{disp} expressed in Equation 3.4, the possible disparity values are directly bound to the sensor’s operational depth range, limiting the search range itself.

Post-processing of Depth Scans

Finally, another *compute shader* layer post-processes the depth maps, smoothing and trimming them according to the sensor’s specifications. If these specifications are not available, one can obtain a reasonable estimation by feeding real images of captured pattern(s) from the sensor into the reconstruction pipeline and derive the parameters from the differences between this reconstructed depth image and the one actually obtained from the sensor, iterating if necessary. Imitating once more the original systems, a hole-filling step is optionally performed to reduce the proportion of missing data.

Figures 3.4 and 3.5 illustrate how *DepthSynth* is able to realistically reproduce the spatial sensitivity of the devices or the impact of surface materials. Similarly, Figure 3.3 (D)-(G) reveals how the data quality of simulated multi-shot structured light sensors is highly sensitive to motion—an observation in accordance with the expectations.

As highlighted in Figures 3.6 and 3.7 through the visual comparisons between *DepthSynth* and other state-of-the-art simulation pipelines, the latter ones are not sensitive to some

realistic effects during capture, or they preserve fine details which are actually smoothed-out up to the window size in block-matching. By closely reproducing the end-to-end process performed by the actual depth devices, *DepthSynth* more exhaustively reproduces the possible sources of noise (*c.f.* Table 3.1).

Background Blending

Most of the depth rendering tools ignore background addition (which can be performed, *e.g.*, by simple alpha compositing), causing significant discrepancy with real data and biasing the learner. Background modeling is hence another key component of *DepthSynth*. Added backgrounds can be:

- From static predefined geometry (*i.e.*, 3D objects added to the scene);
- From predefined geometry with motion (*i.e.*, dynamic 3D objects);
- From large amounts of random primitive 3D shapes (before capture step) or 2D shapes (after capture and reconstruction);
- From a set of real depth scans (*i.e.*, alpha-blending part of these images in a post-processing step).

Optimized for GPU operations (*c.f.* usage of shaders for rendering and post-processing), the whole process can generate ~10 scans (VGA resolution) and their labels (*e.g.*, viewpoints, label map, binary mask, *etc.*) per second on a middle-range computer (Intel E5-1620v2, 16GB RAM, Nvidia Quadro K4200).

3.4 Experiments and Results

To demonstrate the accuracy and practicality of the method detailed in this chapter, Subsection 3.4.2 analyzes the depth error it induces when simulating the *Microsoft Kinect* device, comparing with experimental depth images, other simulation tools, and theoretical models for this device. In Subsection 3.4.3, a state-of-the-art algorithm for classification and pose estimation is chosen to demonstrate how supervised 2.5D recognition methods benefit from using the generated data.

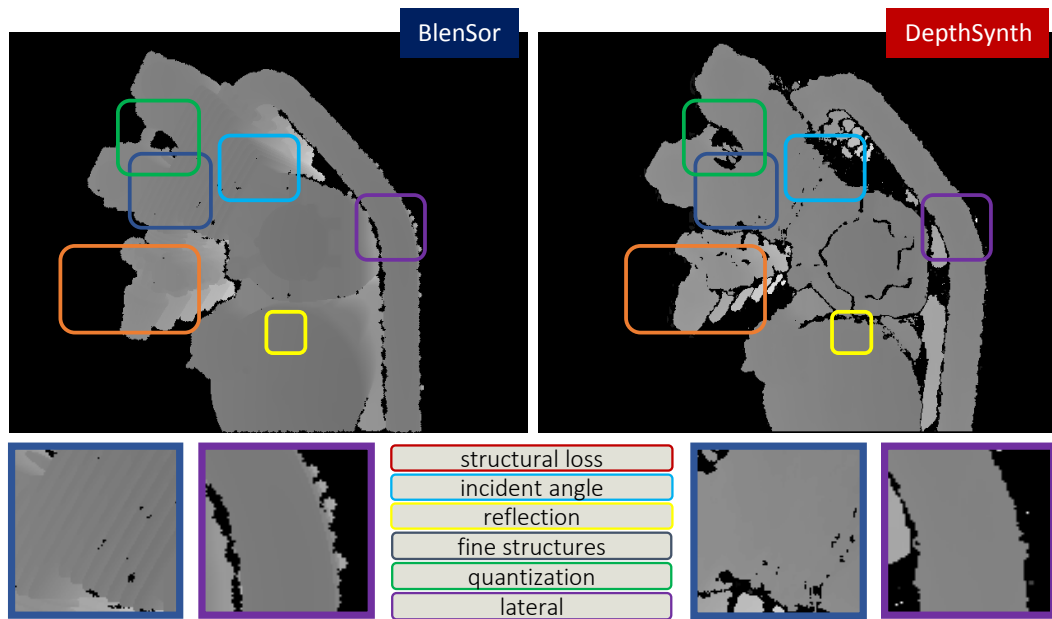


Fig. 3.6: Detailed visual comparison with *BlenSor* [88] highlighting the salient differences, based on the noise study presented in Subsection 3.3.1.

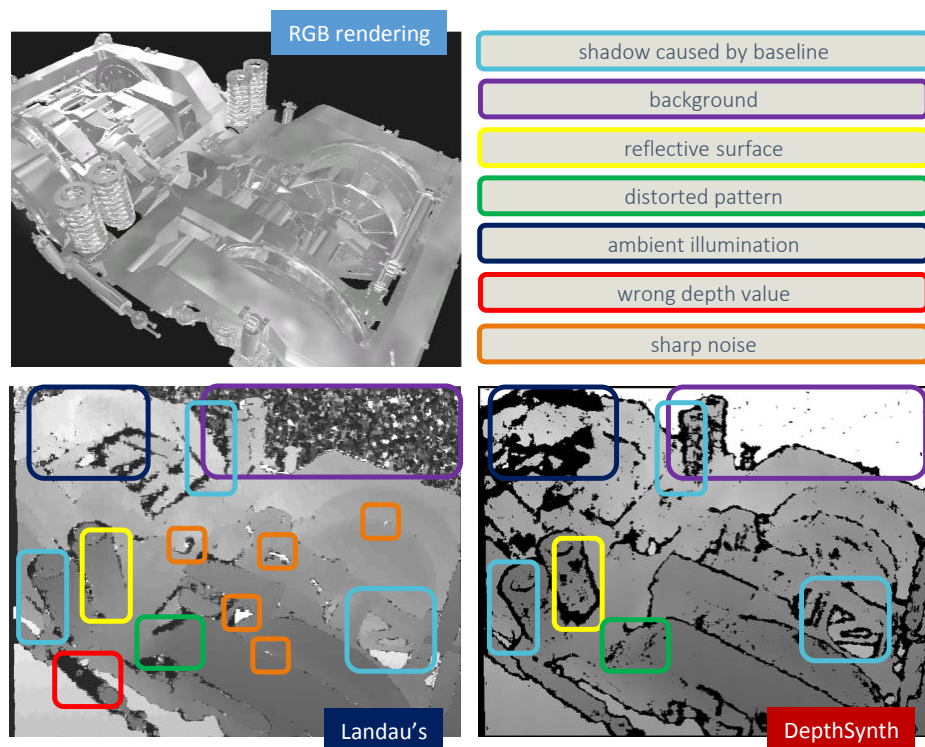


Fig. 3.7: Detailed visual comparison with Landau's solution [138, 139], based on the noise study presented in Subsection 3.3.1.

3.4.1 Implementation Details

Before presenting the experiments, additional implementation details are provided for reproducibility. Though a proof of concept had been first implemented in Matlab, the simulation pipeline used in the following subsections is an optimized version reproduced by colleagues at Siemens Princeton (as *DepthSynth* was immediately needed for an industrial project there).

This version of *DepthSynth* is built on top of Unity 3D Game Engine [64], leveraging its computer graphics capability to project and capture the structured-light patterns in the virtual 3D scenes. The pattern emitter is simulated as a virtual directional light source, which has the pattern image(s) assigned as light cookie.

For the stereo-matching step, instead of trying to reverse-engineer or imitate the algorithms used by the actual depth devices (usually patented), the pipeline that is leveraged in this section simply uses the default stereo-correspondence functions provided by the OpenCV library [28], based on the work of Felzenszwalb and Huttenlocher [69].

3.4.2 Depth Error Evaluation

To validate the correctness of the simulation pipeline, the set of experiments used by Landau *et al.* [138, 139] is reproduced, to compare the depth error induced by *DepthSynth* to experimental values, as well as to the results from Landau *et al.* [138, 139], from *BlenSor* [88], and from 3 theoretical error models for the *Kinect* device—respectively from Menna *et al.* [168], Nguyen *et al.* [185] and Choo *et al.* [46, 138]. All the synthetic and experimental datasets consist of scans of a flat surface placed in front of the 2.5D sensor at various known distances, and at various tilt angles to the focal plane. For each position, several images are captured. The experimental real data was kindly provided by Landau [138, 139].

Figure 3.8-A shows how the distance between the plane and the sensor influences the standard depth error in the resulting scans. The trend in *DepthSynth* data matches well the one observed in experimental scans and in Choo *et al.* model recalibrated by Landau on the same experimental data [138]. As noted by Landau *et al.* [138], these models are based on experimental results which are inherently correlated to the characteristics of their environment and sensor. Therefore, one could expect the error from samples captured in different conditions not to perfectly align with such models (as proved by the discrepancies among them). One can also notice that the quality of *DepthSynth* images degenerates slightly faster as distance increases, compared to real scans; though the proposed method behaves overall more realistically than the others.

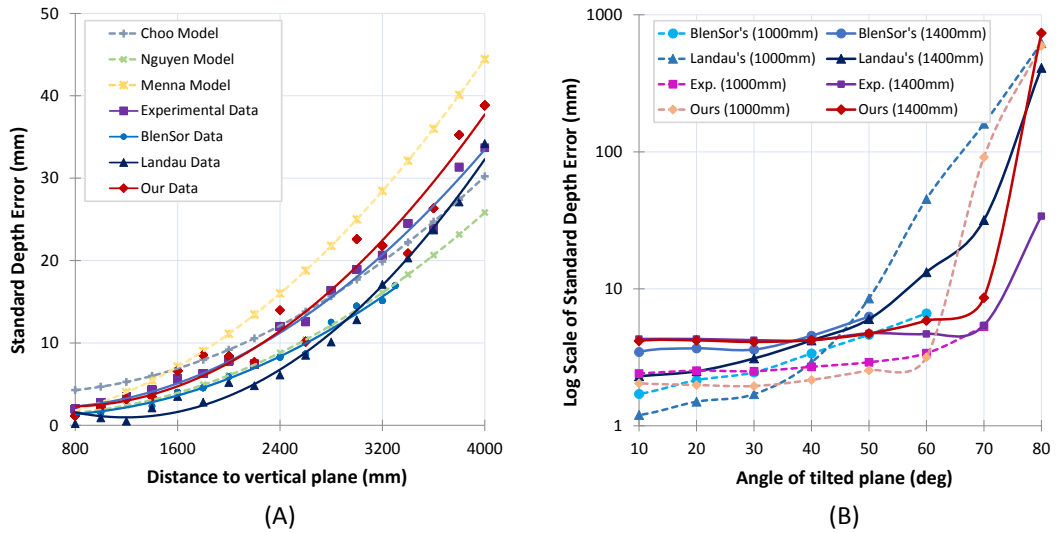


Fig. 3.8: Standard depth error (in mm) computed (A) as a function of the distance (in mm) to a vertical flat wall for various fixed distances; and (B) as a function of its tilt angle (in deg); plotted for the experimental images and the synthetic data from the various solutions and for various fixed distances.

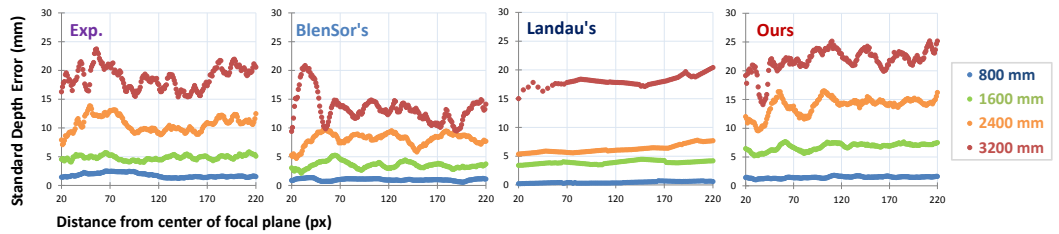


Fig. 3.9: Standard depth error (in mm) as a function of the radial distance (in px) to the focal center; plotted for the experimental images and the synthetic data from the various solutions and for various fixed distances.

Figure 3.8-B represents how the synthetic methods fares when gradually tilting the plane from orthogonal to almost parallel to the optical axis. The errors induced by the proposed pipeline matches closely the experimental results for tilt angles below 70° , with some overestimation for steeper angles. A proper trend is nevertheless observed, unlike for other methods. It should be noted that for larger incident angles, both real scans and *DepthSynth* ones have most of the depth information missing, due to the poor reflection and stretching of the projected pattern(s), heavily impairing the reconstruction.

As a final experiment related to the error modeling, the standard depth error as a function of the radial distance to the focal center is computed. Again, Figure 3.9 highlights the realistic superiority of the proposed pipeline in terms of noise reproduction (despite inducing slightly more noise for larger distances and thus more distorted pattern(s), compared to real devices). *DepthSynth* even satisfyingly reproduces the oscillating evolution of the noise when increasing the distance and reaching the edges of the scans—a well-documented phenomenon caused by “wiggling” and distortion of the pattern(s) [75, 135].

3.4.3 Application to Recognition Tasks

Among the applications which can benefit from the proposed pipeline, the following problem is considered. ICPE is the task of estimating the 6-DOF camera pose and classifying an object instance from single images (2.5D scans here). In the following experiments, it is formulated as an image retrieval problem, supposing no real images can be acquired for the training of the chosen method. However, the 3D models of target objects are considered provided. Therefore, m camera poses are discretized and synthetic 2.5D images are generated, for each pose and each object using *DepthSynth*.

To achieve recognition (*c.f.* recognition solution detailed in Subsection 3.2.1), each available picture is encoded into a discriminative, low-dimensional image representation with its corresponding class and camera pose. This way, a database is built for pose and class retrieval problems. At testing time, given an unseen image, its representation is computed the same way and queried in the database, in order to find the k -nearest neighbor(s) and to return the corresponding class and pose.

In this framework, two components play an important role in ensuring successful indexing. One is the 2.5D image simulation, and the other one is the image representation. A data simulation process is considered valid when it minimizes the quality gap between synthetic and acquired depth data – a minimization achieved here as explained in Section 3.3. On the other hand, a fitting image representation should carry discriminative pose and class information, resulting in successful searches. The degree of discrimination for recognition problems can be defined as the distance between two image representations, which should be small when the objects are similar and their camera poses close to each other, and respectively big when the objects and/or poses are different.

Therefore, to demonstrate the advantages of using *DepthSynth* data irrespective of the selected features, the *triplet* method [286, 300] introduced in Subsection 3.2.1 is selected. For CNN, a simple *LeNet* architecture [142] is applied with custom hyper-parameters. As shown in Figure 2.6, this network is composed of two 5×5 convolution layers, each followed by a ReLU layer and a 2×2 max-pooling layer; with two fully connected layers leading to the output one, also fully connected. This final layer is configured to return the feature vectors, of size $d = 64$ for the following experiments.

Given such a state-of-the-art recognition method (re-implemented in *Caffe* [113]), the accuracy of CNN instances trained on *DepthSynth* data is compared with the accuracy of other CNN instances trained on simple synthetic depth data or on *BlenSor* data. The remaining of this subsection further presents the experiments and discusses their results.

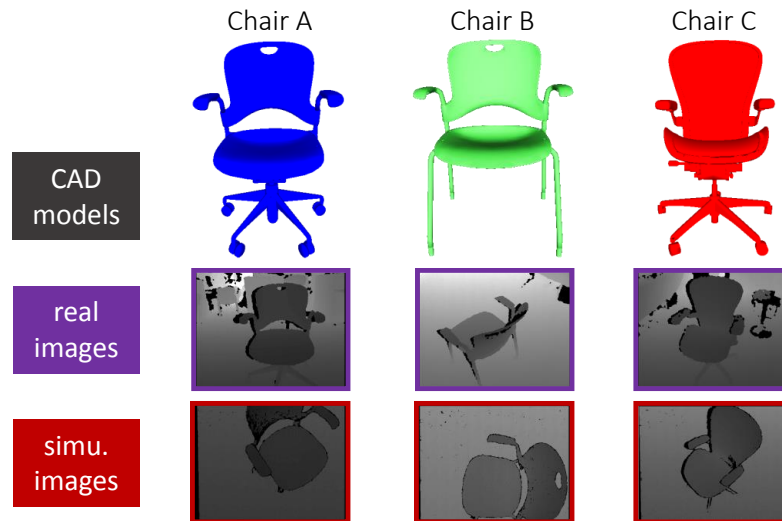


Fig. 3.10: CAD models, sample real images, and *DepthSynth* images used in the experiments (note the strong similarities among these chairs).



Fig. 3.11: Real data acquisition and processing. From left to right: sample real depth data ; sample RGB data with markers ; recovered trajectory (poses) for each samples (for *Chair C*).

Data Preparation

As target objects for the experiment, three similar-looking office chairs are selected, with their CAD models obtained from the manufacturers' websites (Figure 3.10)². In order to capture the test dataset of real depth images with ground-truth pose annotations, Siemens colleagues performed the following procedure.

Augmented reality (AR) markers were placed on the floor around each chair, an *Occipital Structure* sensor was mounted on a tablet, and its infrared camera was calibrated according to the RGB camera of the tablet. Using this tablet, an operator captured sequences of 1,024 RGB and depth frames per object, walking around the chairs (trajectory shown in Figure 3.11).

²At the time this experiment was set, only a few benchmark datasets containing clean 3D models along with labeled depth scans were available (gathering such datasets is tedious, *c.f.* the actual motivation behind this work). The rare datasets meeting these requirements (*e.g.*, LineMOD [99], Rubbers APC [217]) contain toy-like models, too small or too noisily reconstructed for the evaluation of image-rendering solutions; hence the decision to build instead a custom dataset of large and challenging objects.

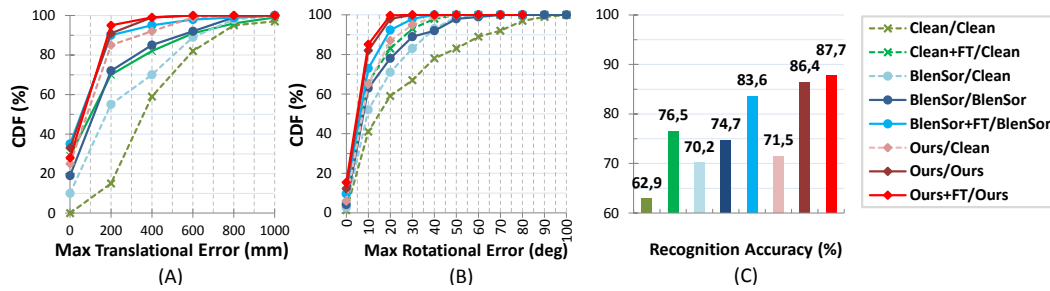


Fig. 3.12: Cumulative distribution functions (CDFs) on errors (A) in translation and (B) in rotation for pose estimation on the *Chair C* dataset. (C) Classification results over the *3-Chairs* dataset, using the method trained over different datasets (“FT” = “fine-tuning”).

In a comprehensive and redundant annotation procedure using robust direct linear transform, 2D-3D correspondences on chairs regions were manually generated based on visual landmarks, choosing a representative set of approximately 60 frames. These estimated camera poses and the detected 2D locations of markers were used to generate triangulated 3D markers locations in the CAD coordinate system. Given the objects’ movable parts, the actual chairs deviate from their model. The deviation was iteratively reduced for the final ground-truth sequence by verifying the reprojections and the consistency of the triangulated markers positions relative to the elements of the chairs.

The IR and RGB camera calibration parameters were then used to align the depth scans into a common 3D coordinate system. In a final fine-tuning step, the poses and 3D models were fed into the simulation pipeline, to generate corresponding noiseless depth maps, used by an iterative closest point (ICP) method [20] to be aligned to the real images; optimizing the ground-truth for the final real test dataset.

DepthSynth images (noisy and noiseless ones) used for the training steps are generated according to the intrinsic parameters of the *Occipital Structure* sensor, sampling viewpoints on a half icosahedron centered on the target CAD models with small pose perturbations (see results in Figure 3.10). In total, 30,000 different synthetic depth images are rendered from the three CAD models, adding a horizontal flat plane as ground floor to the 3D scene. *BlenSor* dataset is generated according to the same procedure and parameters.

Evaluation on Pose Estimation

As a first experiment, the aforementioned approach is limited to pose estimation only, training and testing it over the data of *Chair C*. For the CNN training, the synthetic depth scans are used to form 100,000 triplets and pairs. The learned representation is then applied for the indexation of all the 30,000 images, using the FLANN library [177]. At test time, the representations of the real depth images are extracted and indexed. For each, the nearest

neighbor's pose is then rendered and aligned to the input scan to refine the final 3D pose estimation (using ICP).

To demonstrate how the quality of the synthetic training data impacts the recognition performance, the three different datasets—*noiseless/clean* depth images, *BlenSor* ones, and *DepthSynth* ones—are used either for both representation-learning and database indexing, or only for learning with the *clean* dataset then used for indexing. Furthermore, additional CNN instances are fine-tuned on a subset of 200 real scans, forming 3,000 training samples (triplets and pairs). Estimated 3D poses are compared to the ground-truth values, and the CDFs on errors in rotation and translation are shown in Figure 3.12(A-B).

This figure reveals how the CNN instance trained over *DepthSynth* data gives consistently better results on both translation and rotation estimations; furthermore not gaining much in accuracy after fine-tuning with real data.

Evaluation on Classification

Finally, the classification task for the three similar-looking chairs is considered. Using the same synthetic training datasets extended to all three objects, the accuracy of the recognition methods is computed over a testing dataset of 1,024 real depth images per object, taking as final estimation the class of the nearest neighbor in the descriptor database for each extracted image representation.

Despite the strong similarities among the objects, the recognition method performs reasonably well, as shown in Figure 3.12(C). Again, it can be seen that it gives consistently better results when trained over *DepthSynth* data; and that unlike with other training datasets, the accuracy does not significantly increase with the addition of real data for fine-tuning, validating the inherent realism of the rendered images.

3.5 Discussion

To conclude the presentation of this first study, this section summarizes the contributions, before providing some insights for possible future work.

3.5.1 Contributions

Developed for an industrial project aiming at assisting human operators in recognizing defect parts for train maintenance, this first solution to data scarcity developed during my

thesis has had concrete impacts on the performance of the system. Addressing the realism gap problem for 2.5D data, its key contributions are the following:

Generic Simulation Pipeline for Depth Sensors. Based on a study of depth sensors and the noise impairing them, an end-to-end pipeline is proposed, to synthetically generate depth images from a wide panel of sensors by virtually and comprehensively reproducing the mechanisms of the target devices.

Rendering of Large and Realistic 2.5D Datasets. Presenting realistic properties (*i.e.*, in terms of noise), the resulting images—which can be generated in large numbers as the pipeline leverages graphics processing unit (GPU) operations—can be used to effectively train recognition algorithms when relevant real depth scans are not available (or available in too small numbers). Able to generate large training datasets closer to the real data distribution (*i.e.*, improving the generalization bound and reducing the domain divergence, *c.f.* Subsection 2.3.1), *DepthSynth* facilitates the training of robust 2.5D recognition applications, regardless the ulterior choice of algorithm or feature space.

Versatile Solution Reproducing Sensing Mechanisms. Unlike common domain adaptation methods which are usually statistical in nature, the proposed simulation method is mostly based on deterministic operations (except for some of the post-processing steps, but the intensity of the random noise which they bring is entirely parameterizable). Therefore, *DepthSynth* can be applied as-is to a large variety of applications (*e.g.*, without suffering from dataset bias).

Multi-Stage Validation of 2.5D Simulation Tools. Multiple procedures and their results are shared to validate the proposed solution, comparing to state-of-the-art depth simulation tools. The noise distribution in the resulting images is compared to theoretical models and experimental data, and the effectiveness and flexibility of *DepthSynth* are demonstrated on the training of recognition systems targeting real depth scans.

This concept will hopefully prove itself greatly useful to the community, leveraging the parallel efforts to gather detailed 3D datasets. The generation of realistic depth data and corresponding ground truth can promote a large number of data-driven algorithms, by providing the training and benchmarking resources they need.

3.5.2 Limitations

However, the application of the proposed method to more advanced industrial use-cases highlighted some current limitations, which mostly come from the tight correlation between the quality of the resulting synthetic images and the quality of the 3D models themselves.

Impact of Model Quality on the Simulation. First, while CAD models have well-defined geometries, the same cannot be said to some other models, *e.g.*, obtained through 3D reconstruction from images. Such models may have holes in their *mesh*, wrongly defined surface normals, *etc.*, which affect the rendering process done by the computer graphics engine.

Furthermore, for the simulation to be as realistic as possible, the reflectance of the various materials forming the target objects should be finely defined. While some CAD applications provide catalogs of common material properties which can be applied to 3D models (*e.g.*, model textures simulating *iron, concrete, wood, etc.*), not all models come with such information. However, various methods already exist to capture and assign material reflectance, *e.g.*, as bidirectional reflectance distribution functions (BRDFs) [85, 89, 187].

Finally, when generating synthetic images of target objects, it is common not to know in advance the composition of the environment(s) in which the real objects will be found, or to have 3D models or even images of said environments. In such cases, it is, therefore, challenging to generate relevant visual clutter (background, occlusions, *etc.*).

Leveraging Subset of Real Pictures to Improve Domain Knowledge. To tackle these aforementioned limitations, a possible future work would be to combine *DepthSynth* with post-processing methods such as *PixelDA* [26] or *SimGAN* [240]. Assuming the availability of some real images depicting the target objects, these methods teach generative adversarial networks (GANs) (a type of unsupervised *generative* models further described in the following chapters) to refine synthetic images, *e.g.*, adding statistically-learned camera noise or background to them.

When a small number of real pictures are available, another approach would be to correlate these images in order to understand and virtually reconstruct the target objects/scenes. The resulting models could then be used to directly generate relevant, realistic images. Such an approach is the topic of the next chapter.

Novel View Synthesis through Incremental Scene Learning

” *Man shapes himself through decision that shape his environment.*

— **René Dubos**

(French-American microbiologist and humanist,
20th century)

Another family of scenarios related to data scarcity in computer vision is now tackled, where a subset of images but no relevant models of target objects or scenes are available. More formally, an agent is considered (*e.g.*, a human operator with a camera, a drone exploring a new environment, an autonomous car on the road, *etc.*), which provides sequences of unlocalized images. From these partial observations of an environment, the goal is to understand its distribution in order to generate new, relevant, and consistent images (*e.g.*, images of the observed scene, from new viewpoints).

To that end, the following method is proposed, depicted in Figure 4.1 and detailed in this chapter [206]. Extending a state-of-the-art solution for image localization and registration [98], this method incrementally builds and refines global representations of the target two-dimensional (2D) or three-dimensional (3D) scenes from unlocalized observations. These representations can then be used to render novel images, consistent from one to another. These representations are stored in a global read/write derivable memory with spatial properties, in which, additionally, unobserved regions can be hallucinated in consistency with previous observations, hallucinations, and global priors. The efficacy of the proposed mnemonic and generative pipeline, trainable end-to-end, is demonstrated on various 2D as well as 3D use-cases.

Following a traditional structure, this chapter starts by detailing the motivation in Section 4.1, followed by a study of related work in Section 4.2. Section 4.3 introduces the proposed solution, which is then qualitatively and quantitatively evaluated in Section 4.4. Section 4.5 concludes the chapter, summarizing the contributions and providing some additional insights.

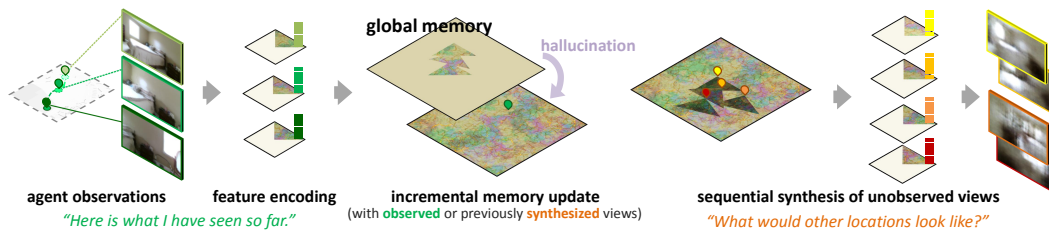


Fig. 4.1: Proposed solution for scene understanding and novel view synthesis, given non-localized agents.

4.1 Motivation

Though the work presented in this chapter also originated from assessing the problems linked to data scarcity in computer vision, further-reaching considerations also motivated and influenced its development.

4.1.1 Visual Understanding for Autonomous Agents

To begin with, it is hard nowadays to think of solutions for scene modeling without considering their potential applications to autonomous systems.

Scene Exploration by Autonomous Agents

Powered by deep learning solutions, an increasing number of autonomous systems are integrating vision-based methods to navigate in—or interact with—their surroundings. The development of self-driving cars is a well-known contemporary example, though the range of applications is constantly broadening: rescue robots operating in hazardous environments, drones monitoring infrastructures in remote areas, *etc.* These autonomous agents have in common the need to understand their 3D environment based on their sensory input, in order to perform their downstream tasks such as planning, exploration, and target navigation [40].

Moreover, like actual creatures, these agents should be able to perform and anticipate under uncertainty. The information gathered at each instant from sensory stimuli only provides a partial picture of the surroundings. Therefore, the ability to *patch* together these partial observations as they come to build a global representation, and the ability to complete it from experience, are primordial to understand new environments and plan actions accordingly.

We, humans, perform this process all the time. For instance, as we walk through the streets of a new city, we are able to build a mental representation of its layout. Based on this representation and our knowledge of other agglomerations, we can also predict, with more

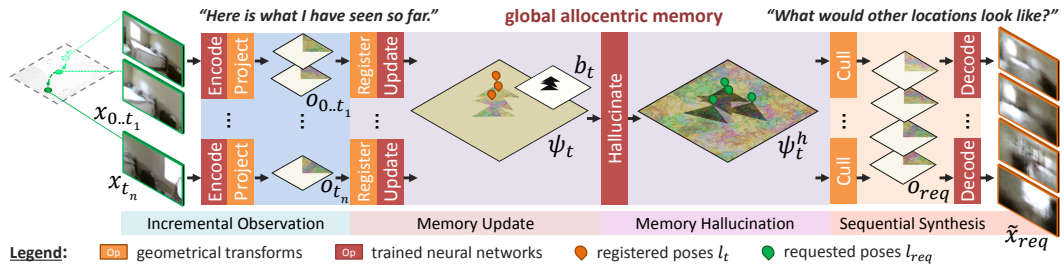


Fig. 4.2: Detailed pipeline for incremental view synthesis, considering non-localized agents exploring new scenes. Observations x_t are sequentially encoded and registered in a global feature map ψ_t with spatial properties, used to extrapolate unobserved content and generate consistent novel views \tilde{x}_{req} from requested viewpoints.

or less confidence, how parts of the city which have not been explored yet may look like, and we can act upon these predictions (*e.g.*, heading to larger boulevards when looking for a tramway station).

To perform these mental tasks, we rely on multiple high-level neural processes: spatial localization and memorization, mnemonic querying and mental visualization, knowledge transfer, *etc.* While the current state-of-the-art in machine learning cannot compare to our neurological aptitudes, autonomous agents able to emulate such mental and visual tasks—even to some extent—would greatly gain in adaptability.

Capturing and Modeling New Scenes

The ability to incrementally build global representations from visual observations would benefit not only the agents in their current tasks, but also systems which could capitalize on the acquired information.

Picking once again the case of self-driving vehicles as an example, more and more companies are trying to leverage the continuous data flow captured by these vehicles. As they are typically connected to global positioning system (GPS), the data captured by their sensors can directly be localized and registered in global topographic representations, *e.g.*, to update or refine urban maps. These databases of higher definition can then be applied to the development of more precise and robust navigational algorithms, for example. Note, however, that this registration process becomes much more complex when provided data samples are not localized (*e.g.*, images captured by a human operator walking in a new scene).

Overall, the capacity to build appropriate representations from sets of partial observations could benefit the training of recognition methods, for scenes or objects which have not been modeled yet.

4.1.2 Novel Image Synthesis

Besides these far-reaching considerations, the proposed method was mainly developed to tackle the problem of data scarcity when training robust solutions for visual recognition.

Shortcomings of Simulation

In Chapter 3, we saw how simulation tools can be powerful solutions to the generation of large amount of realistic images. As demonstrated, synthetic datasets can then be successfully applied, *e.g.*, to the training of machine learning methods.

However, this previous chapter also highlighted the limitations of simulation-based solutions w.r.t. the problem of data scarcity. High-quality simulated data implies the acquisition of models, themselves of high-quality. In many cases, it would be as costly (in terms of hardware requirements and human efforts) to build such realistic models as to simply gather and annotate images (if not more). There is, therefore, a need for any method that could further automate the modeling of new objects or scenes.

Interpolation and Extrapolation of Novel Images

The most practical motivation behind the following work is thus the capacity to model the content of a subset of images, in order to generate new relevant images.

Applications could go from the creation of larger datasets from small sequences of target images (“given this handful of observations of a scene, generate more images of the scene from any viewpoint”) to the guiding of autonomous agents, as mentioned before (“given previous observations, what could be found in different parts of the scene?”).

Unlike the usual use-case of generative models (*c.f.* Subsection 2.1.3), the considered images are here not independently and randomly sampled from a target distribution, but are sequences of partial observations provided by an agent whose policy and localization are unknown. The target task is, therefore, to sequentially learn the distribution of the target scene, in order to sample relevant and globally consistent images from it. These images should be interpolated from the representation, or extrapolated from domain-relevant knowledge for unexplored/unobserved regions.

4.2 Related Work

The problem tackled in this chapter relates to image-conditioned localization, mapping, and view synthesis. Relevant work is discussed in the following section, presenting traditional methods, as well as novel neural solutions which the proposed pipeline will be compared with.

4.2.1 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) is a long-standing problem in computer vision. Given an agent exploring an unknown environment and providing regular observations (*e.g.*, RGB images, depth scans, radar measurements, *etc.*, depending on the use-cases), the task consists of localizing the agent at each time step while registering the observations in a global map structure (*e.g.*, a topographic map or RGB point cloud).

Traditional SLAM

Different from structure-from-motion (SfM) solutions [188, 252, 289] that typically operate offline and on unordered images, classic SLAM methods [59, 71, 116, 124, 174, 179, 251] usually rely on real-time continuous cues in videos and on visual consistency among frames. Traditional methods are composed of several carefully engineered deterministic or statistical modules for tracking, mapping, and re-localization (*e.g.*, bundle adjustment modules, Kalman filtering, *etc.*).

While current SLAM methods can perform efficiently for a wide range of applications (from autonomous driving to augmented reality for instance), they are rigid systems. As argued by Zhang et al. [305], authors of the Neural SLAM technique, traditional SLAM methods cannot be tightly integrated into end-to-end systems for autonomous agents, given the current predominance of ANN-based solutions for decision making and other downstream tasks. Unable to interact directly with—and adapt to—non-derivable systems, neural-based methods can only be passed the final output of SLAM algorithms (*e.g.*, as done by Bhatti et al. [21]). Zhang et al. [305] further argue that if the SLAM system and the modules in charge of the downstream tasks (navigation, exploration, novel view synthesis, *etc.*) could be “deeply integrated as a whole”, they would benefit from their mutual training and from feature sharing.

Neural Scene Understanding

This conclusion, recently and simultaneously drawn by various research teams in the domain of reinforcement learning, led to a variety of methods trying to solve SLAM with neural networks.

Some of these novel methods, such as CNN-SLAM [263], have been focusing on replacing specific modules in classical SLAM systems with neural components. More experimental solutions have been instead developed from scratch with derivable components only. These solutions are typically relying on recurrent neural networks (RNNs), used to accumulate features from image sequences, *e.g.*, to predict the camera trajectory [197, 219] or infer novel views [308].

Extending these solutions with a queryable memory, state-of-the-art models are mostly egocentric and action-conditioned [40, 72, 196, 211, 305]. They usually consider an agent exploring the environment, which can provide not only an observation x_t at each time step t , but also its state s_t (*e.g.*, its pose) or action a_t leading to this new observation (*e.g.*, “move forward”, “rotate clockwise”, *etc.*). It is, therefore, assumed that some “oracle” is providing the agent’s state or action at each time step t [196]—a condition not applicable to many real-life scenarios. The aforementioned methods use the agent’s state (obtained directly from the oracle or regressed from the provided action) to index the corresponding observation x_t or its features in a memory structure.

In the spirit of SLAM, Neural SLAM [305] and MapNet [98] are two neural methods relying solely on the visual observations to regress the poses. Both propose a spatial memory system for autonomous agents. Whereas the former deeply interconnects memory operations with other predictions (*e.g.*, motion planning), the latter offers a more generic solution with no assumption on the agents’ range of action or goal. Extending MapNet, the proposed model not only attempts to build a map of the environment, but also makes incremental predictions and hallucinations based on both past experience and current observations.

4.2.2 Incremental Scene Sampling

While the localization and registration steps are a necessity given the considered applications, the end goal is the generation of novel views over a partially observed scene, irrespective of the agent’s nature and the view density.

3D Modeling for Image Rendering

Most SLAM methods register the observations in a global structure which can then be queried to generate new samples. For example, the 3D point clouds returned by some RGB-D solutions can be passed to computer graphics pipelines to render images, *e.g.*, from different viewpoints [47, 124, 179, 251].

Methods based on convolutional neural networks (CNNs) have also been proposed to model 3D content from 2D or 2.5D images. Relying on advanced 3D convolutions, these solutions can predict voxels, point clouds, or meshes [48, 68, 149]. Trained in a supervised manner, these CNNs, however, require 3D data to learn their model regression.

Capitalizing on the advances in computer graphics, all the aforementioned methods yield satisfying results when enough source samples are provided to build a dense 3D representation. However, they cannot compensate for sparse information (*i.e.*, the graphics engines can only render images from observed and registered content; missing elements cannot be inferred, and may create artifacts in the resulting images).

Novel View Synthesis

Motivated by the idea of an end-to-end neural framework, the task of generating new images is tackled from a different angle in this chapter. Given a sequence of non-localized observations from a target scene, neural solutions should be able, at any time step, to directly return images from new, requested viewpoints.

A number of existing methods [65, 262, 297] follow the *encoder-decoder* paradigm. Either fed to the network altogether or sequentially (with an RNN used as encoder) along with the requested viewpoint, the features extracted from the input images are combined, before being decoded into the target view. Other methods try instead to predict a per-pixel mapping from the provided images to the requested one. For instance, Zhou et al. [309] apply a CNN to estimate the visual flows which map the pixels in the source images to the target one. Comparably, Ji et al. [112] teach a neural network (NN) to predict the dense correspondences to infer the intermediate image between two consecutive observations.

Extending the generative query network (GQN) proposed by Eslami et al. [65] with a derivable memory structure, the generative temporal model with spatial memory (GTM-SM) developed by Fraccaro et al. [72] is most probably the pipeline closest to the one proposed in this chapter, both in terms of target applications and overall structure. GTM-SM takes for inputs a sequence of observations x_t from an agent and the corresponding encoded actions a_t . The actions are passed to a first network tasked to infer from them the agent's states s_t

(*i.e.*, its pose at each time step), while the observations are passed in parallel to the GQN in order to be encoded. The code vectors are then stored into a differentiable neural dictionary (DND) used as memory [211], with the inferred states s_t used as keys. Once the observation phase over, new images can be synthesized while the agent continues to virtually navigate in the scene. Inferring again its state from its latest action, the DND can be queried to retrieve relevant features from the closest memorized states. Interpolating and decoding these features, a novel image is then generated.

While yielding impressive results when agents densely explore and observe the scenes first, GTM-SM—like other methods mentioned in this subsection—cannot extrapolate beyond the observed domain, unlike the proposed framework.

4.3 Methodology: Neural Pipeline for Incremental Scene Synthesis

While the current state of the art in scene registration yields satisfying results, methods rely on several assumptions, including prior knowledge of the agent’s range of actions, as well as the actions a_t themselves at each time step. Here, unknown agents are considered, with only their observations x_t provided during the memorization phase. In the spirit of the MapNet solution [98], an allocentric spatial memory map is used. Projected features from the input observations are registered together in a coordinate system relative to the first inputs, allowing to regress the position and orientation (*i.e.*, *pose*) of the agent in this coordinate system at each step. Moreover, given viewpoints and camera intrinsic parameters, features can be extracted from the spatial memory (*frustum culling*) to recover views. Crucially, at each step, memory “holes” can be temporarily filled by a NN trained to generate domain-relevant features while ensuring global consistency. Put together (*c.f.* Figure 4.2), this pipeline (trainable both separately and end-to-end) can be seen as an explicit topographic memory system with localization, registration, and retrieval properties, as well as consistent memory-extrapolation from prior knowledge. The proposed approach is detailed in this section.

4.3.1 Localization and Memorization

The solution first takes a sequence of observed images $x_t \in \mathbb{R}^{c \times h \times w}$ (*e.g.*, with $c = 3$ for RGB images or 4 for RGB-D ones) for $t = 1, \dots, \tau$ as input, localizing them and updating the spatial memory $\psi \in \mathbb{R}^{n \times u \times v}$ accordingly. The memory ψ is a discrete global map of dimensions $u \times v$ and feature size n . ψ_t represents its state at time t , after updating ψ_{t-1} with features from x_t .

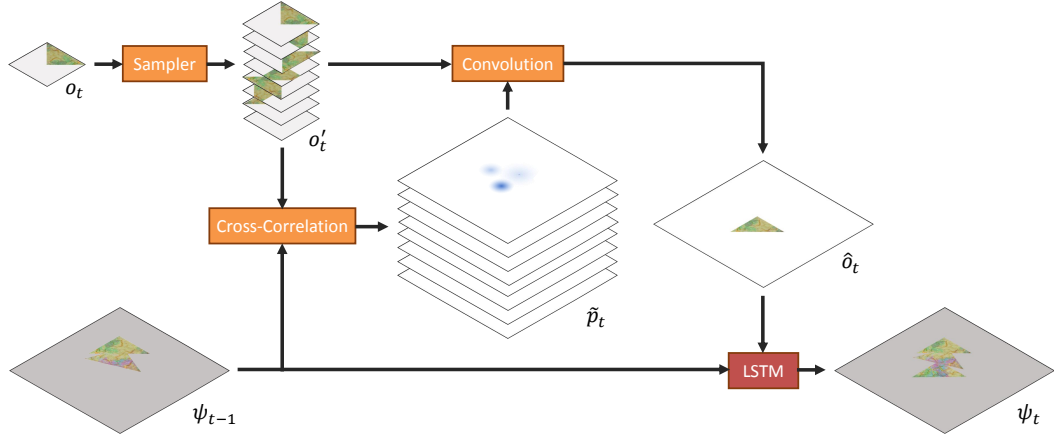


Fig. 4.3: Localization and memorization, based on MapNet [98].

Encoding Memories. Observations are encoded to fit the memory format. For each observation, a feature map $x'_t \in \mathbb{R}^{n \times h' \times w'}$ is extracted by an encoding convolutional neural network (with n the feature size). Each feature map is then projected from the 2D image domain into a tensor $o_t \in \mathbb{R}^{n \times q \times q}$ representing the agent's spatial neighborhood (to simplify later equations, u, v, q are supposed to be odd numbers). This operation is data and use-case dependent.

For instance, for 2D scenes (*i.e.*, agents walking on an image plane), this operation can be done by cropping x'_t into a square tensor of shape $n \times q' \times q'$ with $q' = \min(h', w')$, followed by scaling the features from $q' \times q'$ to $q \times q$ using bilinear interpolation.

For RGB-D observations of 3D scenes (or for RGB images extended by some monocular depth estimation method, *e.g.*, [36, 61, 123, 137, 222, 296]), the feature maps are first converted into 3D point clouds (after registering color and depth images together) using the depth values and the camera intrinsic parameters (assuming like Henriques and Vedaldi [98] that the ground plane is approximately known). To project the inputs into point clouds, $\forall i \in \{0, \dots, h-1\}$ and $\forall j \in \{0, \dots, w-1\}$, each feature $x'_{t,i,j} \in \mathbb{R}^n$ in x'_t is assigned the coordinates (x, y, z) , as performed in [98]:

$$\begin{aligned}
 z &= x'_{t,i,j}{}^D, \\
 x &= (j - c_x) \frac{z}{f_{px,x}}, \\
 y &= (i - c_y) \frac{z}{f_{px,y}},
 \end{aligned} \tag{4.1}$$

with x^D the depth (2.5D) map, $f_{px,x}, f_{px,y}$ the horizontal and vertical pixel focal lengths of the 2.5D sensor, and c_x, c_y its pixel focal center.

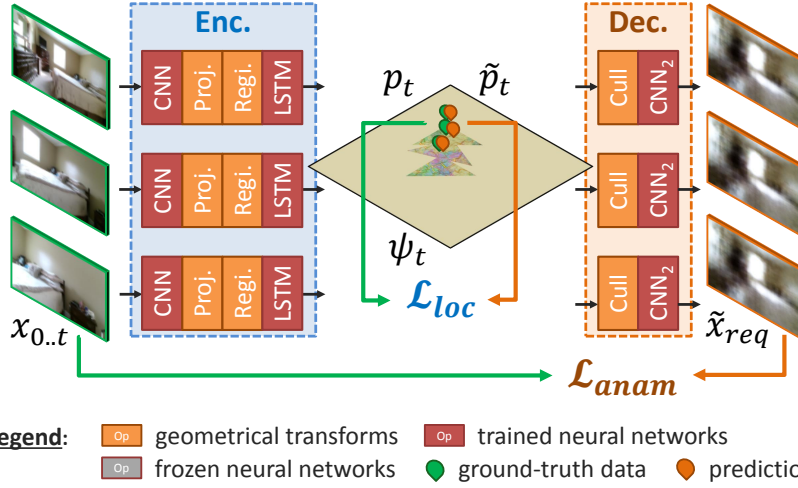


Fig. 4.4: Training of the memorization and anamnesis modules. These pipeline components can be seen as a particular auto-encoder (*i.e.*, with memory), and can be trained end-to-end as such. \mathcal{L}_{loc} measures the accuracy of the predicted allocentric poses *i.e.*, training the encoding CNN to extract meaningful features and the LSTM to update the memory properly. \mathcal{L}_{anam} measures the quality of the recalled views—rendered from ψ_t using the ground-truth poses—compared to the original ones (note: though the training steps are shown separately in Figures 4.4-4.6, the whole method is trained in a single pass).

Each set of coordinates is then discretized to obtain the neighborhood bin that the feature belongs to. Given $q \times q$ bins of dimensions (x_q, z_q) in world units, the bin coordinates (x_b, z_b) of each feature are computed as follows:

$$\begin{aligned} x_b &= \lfloor \frac{x}{x_q} \rfloor + \frac{q-1}{2}, \\ z_b &= \lfloor \frac{z}{z_q} \rfloor + \frac{q-1}{2}, \end{aligned} \quad (4.2)$$

with $\lfloor \cdot \rfloor$ the integer flooring operation. Features projected out of the $q \times q$ area are ignored. Finally, o_t is obtained by applying a max-pooling operation over each bin (*i.e.*, handling many-to-one feature aggregation by keeping only the maximum values for features projected into the same bin [212]). Empty bins result in a null value¹.

Localizing and Storing Memories. Given a projected feature map o_t and the current memory state ψ_{t-1} , the registration process involves densely matching o_t with ψ_{t-1} , considering all possible positions and rotations. As explained by Henriques and Vedaldi [98], this can be efficiently done through cross-correlation (*c.f.* Figure 4.3). Considering a set of r yaw rotations, a bank $o'_t \in \mathbb{R}^{r \times n \times q \times q}$ is built by rotating o_t r times:

$$o'_t = \{R(o_t, 2\pi \frac{i}{r}, c_{q,q})\}_{i=0}^r, \quad (4.3)$$

¹Max-pooling over a sparse tensor (point cloud) as done here is a complex operation not yet covered by all deep learning frameworks at the time of this project. I thus implemented my own (for the PyTorch library [198]).

with $c_{q,q} = (\frac{q+1}{2}, \frac{q+1}{2})$ horizontal center of the patch, and $R(o, \alpha, c)$ the function rotating each element in o around the position c by an angle α in the horizontal plane.

The dense matching can, therefore, be achieved by sliding this bank of r feature maps across the global memory ψ_{t-1} and comparing the correlation responses. In other terms, the localization probability field $\tilde{p}_t \in \mathbb{R}^{r \times u \times v}$ is efficiently obtained by computing the cross-correlation (*i.e.*, “convolution”, operator \star , in deep learning literature) between ψ_{t-1} and o'_t and normalizing the response map (*softmax* activation σ). The higher a value in \tilde{p}_t , the stronger the belief the observation comes from the corresponding pose. Given this probability map, it is possible to register o_t into the global map space (*i.e.*, rotating and translating it according to \tilde{p}_t estimation) by directly convolving o_t with \tilde{p}_t . This registered feature tensor $\hat{o}_t \in \mathbb{R}^{n \times u \times v}$ can finally be inserted into memory:

$$\hat{o}_t = \tilde{p}_t * o'_t \quad \text{with} \quad \tilde{p}_t = \sigma(\psi_{t-1} \star o'_t), \quad (4.4)$$

$$\psi_t = \text{LSTM}(\psi_{t-1}, \hat{o}_t, \theta_{\text{LSTM}}). \quad (4.5)$$

A long short-term memory (LSTM) unit is used, to update ψ_{t-1} (the unit’s *hidden* state) with \hat{o}_t (the unit’s input) in a knowledgeable manner (*c.f.* trainable parameters θ_{LSTM}). During training, the RNN will indeed learn to properly blend overlapping features, and to use \hat{o}_t to solve potential uncertainties in previous insertions (uncertainties in p result in blurred \hat{o} after convolution). The LSTM is also trained to update an occupancy mask of the global memory, later used for constrained hallucination (*c.f.* Section 4.3.3).

Training. The aforementioned process is trained in a supervised manner given the ground-truth agent’s poses. For each sequence, the feature vector $o_{t=0}$ from the first observation is registered at the center of the global map without rotation (origin of the allocentric system). Given p_t the one-hot encoding of the actual state at time t , the network’s loss \mathcal{L}_{loc} at time τ is computed over the remaining predicted poses using binary cross-entropy:

$$\mathcal{L}_{loc} = \frac{1}{\tau} \sum_{t=1}^{\tau} [-p_t \cdot \log(\tilde{p}_t) + (1 - p_t) \cdot \log(1 - \tilde{p}_t)]. \quad (4.6)$$

4.3.2 Anamnesis

Applying a novel combination of geometrical transforms and decoding operations, memorized content can be recalled from ψ_t and new images from unexplored locations synthesized. This process can be seen as a many-to-one recurrent generative network, with image synthesis conditioned by the global memory and the requested viewpoint. The following subsection presents how the entire neural network can thus be advantageously trained as an auto-encoder (AE) with a recurrent neural encoder and a persistent latent space.

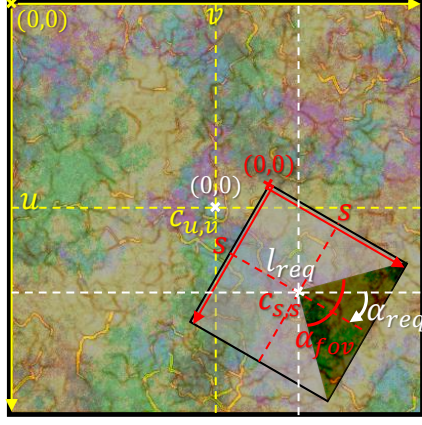


Fig. 4.5: Geometrical Memory Culling (2D representation). The feature vector o_{req} , defining the observation for an agent positioned at l_{req} and rotated by an angle α_{req} in the allocentric system, is extracted from ψ_t through a series of geometrical transforms (rotation, translation, clipping, culling). Coordinates and distances in the allocentric coordinate system are represented in white; in yellow for the ψ_t matrix system; and red for the o_t one.

Culling Memories. While a decoder can retrieve observations conditioned by the full memory and requested pose, it would have to disentangle the visual and spatial information itself, which is not a trivial task to learn (*c.f.* ablation study in Section 4.4.2). Instead, taking advantage of the spatial properties of the proposed allocentric memory, the features in the requested viewing volumes are first *culled*, before being passed as only inputs to the image decoder. More formally, given the allocentric coordinates $l_{req} = (u_{req}, v_{req})$, orientation $\alpha_{req} = 2\pi \frac{r_{req}}{r}$, and view field α_{fov} , $o_{req} \in \mathbb{R}^{n \times q \times q}$ representing the requested neighborhood is filled as follows:

$$o_{req,kij} = \begin{cases} \hat{o}_{req,kij} & \text{if } \text{atan2}(j - \frac{q+1}{2}, i - \frac{q+1}{2}) < \frac{\alpha_{fov}}{2}, \\ -1 & \text{otherwise,} \end{cases} \quad (4.7)$$

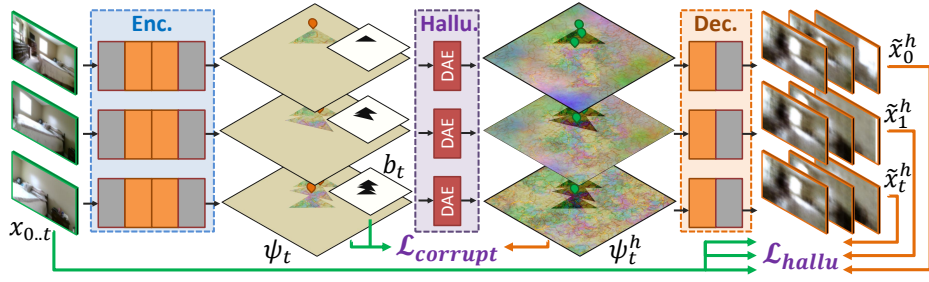
with \hat{o}_{req} the unculled feature patch extracted from ψ_t rotated by $-\alpha_{req}$. Formally, $\forall k \in [0 \dots n-1]$, $\forall (i, j) \in [0 \dots q-1]^2$:

$$\hat{o}_{req,kij} = R(\psi_t, -\alpha_{req}, c_{u,v} + l_{req})_{k\xi\eta} \quad (4.8)$$

with $(\xi, \eta) = (i, j) + c_{u,v} + l_{req} - c_{q,q}$.

Note that for this step, one can use a larger view field than requested (α_{fov}), in order to provide the feature decoder with more context (*e.g.*, to properly recover visual elements at the limit of the agent's view field).

This differentiable operation, illustrated in Figure 4.5, combines feature extraction (through translation and rotation) and *viewing frustum culling* (*c.f.* computer graphics to render large 3D scenes).



Legend: ■ geometrical transforms ■ trained neural networks ■ frozen neural networks ● ground-truth data ● predictions

Fig. 4.6: Training of the hallucination module. The goal of this sub-network is to fill the “holes” in the global memory/map, hallucinating domain-relevant features which could be used to generate novel views (while keeping existing/memorized features uncorrupted). Therefore, it is trained to predict all future observations at each step (\mathcal{L}_{hallu}), while being punished for any corruption to the global map and recalled observations ($\mathcal{L}_{corrupt}$) (note: though the training steps are shown separately in Figures 4.4-4.6, the whole method is trained in a single pass).

Decoding Memories. As input observations undergo encoding and projection, feature maps culled from the memory go through a reverse procedure to be projected back into the image domain. With the synthesis conditioning covered in the previous step, a decoder directly takes o_{req} (*i.e.*, the view-encoding features) and returns \tilde{x}_{req} , the corresponding image.

This back-projection is still a complex task. The decoder must both project the features from voxel domain to image plane, and decode them into visual stimuli. Previous works (*e.g.*, [120, 212]) and qualitative results demonstrate that a well-defined (*e.g.*, *geometry-aware*) network can successfully accomplish this task.

Training. The pipeline can be trained end-to-end as an image-sequence AE, by requesting it to recall given observations—*i.e.*, setting $l_{req,t} = l_t$ and $r_{req,t} = r_t, \forall t \in [1, \tau]$, with l_t and r_t the agent’s ground-truth position/orientation at each step t (*c.f.* Figure 4.4). Therefore, its loss \mathcal{L}_{anam} is computed as the L1 distance between x_t and $\tilde{x}_{req,t}, \forall t \in [0, \tau]$, averaged over the sequences. Note that thanks to the modularity of the proposed framework, the global map and registration steps can be removed to pre-train the encoder and decoder together (passing the features directly from one to the other). It has been observed that such a pre-training tends to stabilize the overall learning process.

4.3.3 Mnemonic Hallucination

While the presented pipeline can generate novel views (*i.e.*, from poses not explored by the agent yet), these views have to overlap previous observations for the solution to extract enough features for anamnesis. Therefore, the memory system is extended with an *extrapolation* module. Given prior domain-relevant knowledge (*i.e.*, an estimation of the

scene distributions learned during training), this module can *hallucinate* relevant features for unexplored regions.

Hole Filling with Global Constraints. Under global constraints, an AE is defined to perform in the feature domain, taking ψ_t as input and returning a convincingly hole-filled version ψ_t^h while leaving registered features uncorrupted. In other words, this module should provide relevant features which seamlessly integrate existing content according to prior domain knowledge.

Training. Assuming the agent homogeneously explores training environments, the hallucinatory module is trained at each step $t \in [0, \tau_{cur}]$ by generating ψ_t^h , hole-filled memory used to predict yet-to-be-observed views $\{x_t\}_{t=\tau_{cur}+1}^\tau$. To ensure that registered features are not corrupted, the training process also verifies that all observations $\{x_t\}_{t=0}^{\tau_{cur}}$ can be retrieved from ψ_t^h (c.f. Figure 4.6). This generative loss is computed as follows:

$$\mathcal{L}_{hallu} = \frac{1}{\tau(\tau-1)} \sum_{t=0}^{\tau-1} \sum_{i=0}^{\tau} \|\tilde{x}_i^h - x_i\|_1, \quad (4.9)$$

with \tilde{x}_i^h the view recovered from ψ_i^h using the agent’s true location l_i and orientation r_i for its observation x_i .

Additionally, another loss is directly computed in the feature domain, using memory occupancy masks b_t to penalize any changes to the registered features (given \odot Hadamard product):

$$\mathcal{L}_{corrupt} = \frac{1}{\tau} \sum_{t=0}^{\tau} \|(\psi_t^h - \psi_t) \odot b_t\|_1. \quad (4.10)$$

Trainable end-to-end, the model efficiently acquires domain knowledge to register, hallucinate, and synthesize scenes.

4.4 Experiments and Results

In order to demonstrate the capabilities of the proposed mnemonic and generative system, the following section details several experiments on various synthetic and real 2D and 3D environments. For each experiment, an unknown agent is considered, which is exploring the scenes and only providing a short sequence of partial observations (limited view field). The solution has to localize and register the observations and build a global representation of the scene. Given a set of requested viewpoints, it should then render the corresponding views. This section proposes qualitative and quantitative evaluations of the predicted trajectories and views, comparing with GTM-SM [72].

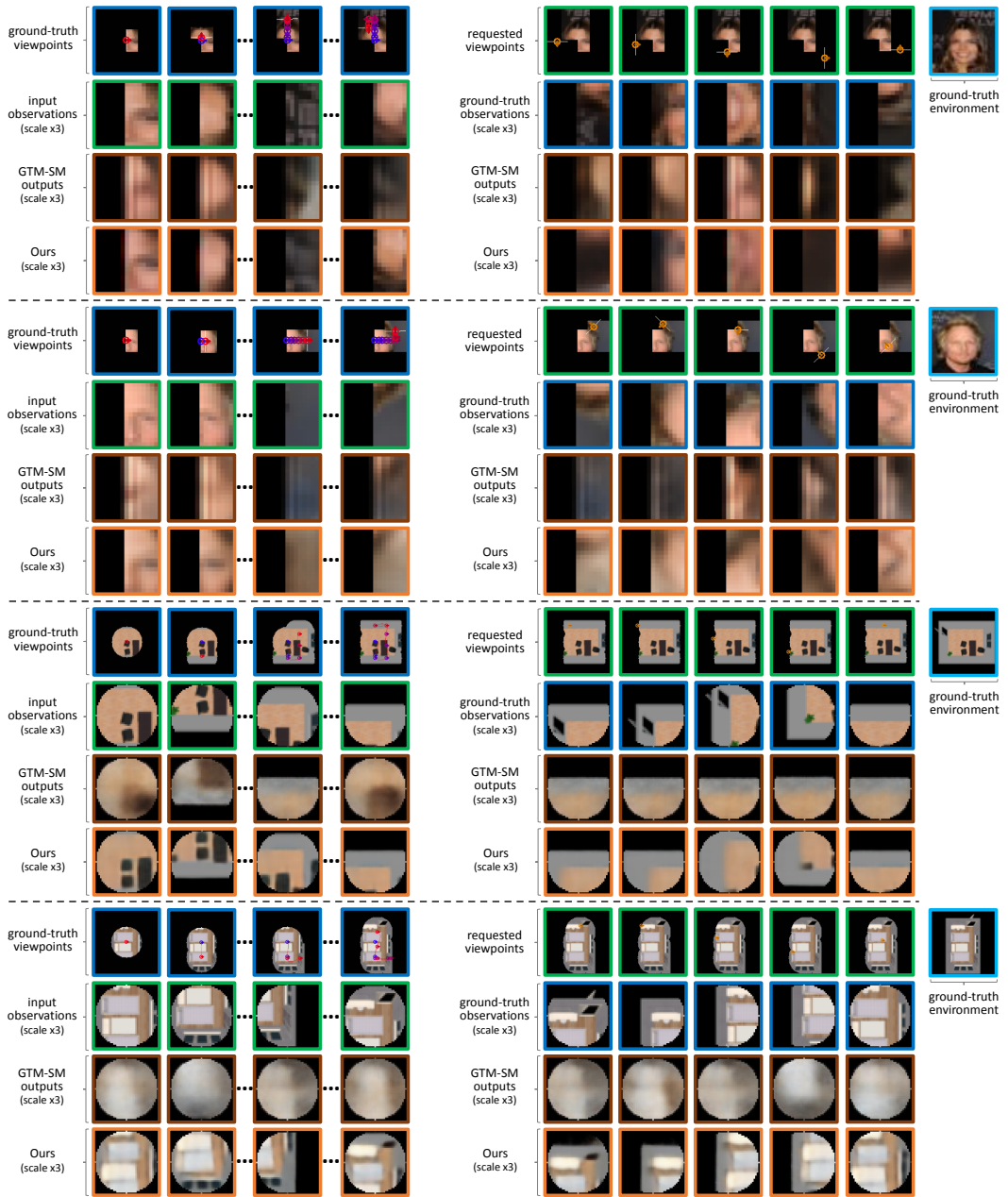


Fig. 4.7: Synthesis of memorized and novel views from 2D scenes (*c.f.* setups in Subsection 4.4.2). Compared to GTM-SM [72], the proposed method benefits from prior knowledge and global representation.

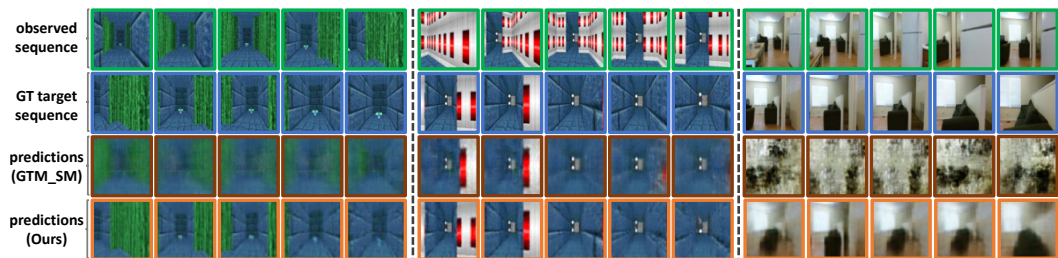


Fig. 4.8: Novel view synthesis from 3D scenes (*c.f.* setups in Subsection 4.4.3).

4.4.1 Implementation Details

Before developing on the experiments themselves, the following subsection provides further details regarding the several interlaced components of the pipeline and their implementation, for reproducibility.

Module Architectures

The solution presented in this chapter is orthogonal to the choice of neural networks for its various modules. Therefore, to evaluate the pipeline in easy-to-reproduce settings and to compare on an equal footing with GTM-SM in the following experiments, the observation-encoding network, the hallucinatory AE, and the image decoder are all implemented as shallow residual networks (ResNets) [97] with 4 blocks. Note that the whole pipeline is implemented using the PyTorch framework [198].

Encoder and Decoder. The encoder is thus configured to output feature maps $x'_t \in \mathbb{R}^{n \times h' \times w'}$ with the same dimensions as the inputs $x_t \in \mathbb{R}^{c \times h \times w}$, *i.e.*, $h = h'$, $w = w'$. The decoder network receives inputs feature tensors $o_{req} \in \mathbb{R}^{n \times s \times s}$ and its last convolutional layers are parametrized to output the image tensors $\tilde{x}_{req} \in \mathbb{R}^{c \times h \times w}$. For the experiments in which the global memory is directly sampled into an image, another ResNet-4 decoder is trained, directly receiving $\psi_t^h \in \mathbb{R}^{n \times u \times v}$ for input and returning $\tilde{x}_{g,t} \in \mathbb{R}^{c \times h_g \times w_g}$ (with h_g, w_g dimensions of global images for the target experiment), to compare the generated image with the original global image (L1 loss).

Hallucinatory AE. The hallucinatory AE is, by definition, parametrized to output a hole-filled tensor of same dimensions as the input memory structure. In order to improve the sampling of hallucinated features and the global awareness of this network, several concepts from SAGAN [304] are adopted:

- Spectral normalization [172] is applied to the weights of each convolution layer in the feature-extracting blocks of the AE network (as SAGAN authors demonstrated it can prevent unusual gradients and stabilize training);
- To model relationships between distant regions, self-attention layers [44, 195, 275, 304] replace the two last convolutions of the network.

Given a feature map $o \in \mathbb{R}^{n \times u \times v}$, the result o_{sa} of the self-attention operation is:

$$o_{sa} = o + \gamma(W_h \star o)\sigma\left((W_f \star o)^\top(W_g \star o)\right)^\top, \quad (4.11)$$

with $W_f \in \mathbb{R}^{\bar{n} \times n}$, $W_g \in \mathbb{R}^{\bar{n} \times n}$, $W_h \in \mathbb{R}^{n \times n}$ learned weight matrices (opting for $\bar{n} = n/8$ as in [304]); and γ a trainable scalar weight.

Additionally, following a generative adversarial network (GAN) strategy [86, 110, 213, 226], this conditioned generative network is also trained against a discriminative one whose task is to evaluate the *realism* of feature patches o_t^h culled from ψ_t^h . This discriminator is itself trained over o_t (*real* samples) and o_t^h (*fake* ones). Note that the architecture of the discriminator also draws from SAGAN [304]; *i.e.*, it is a simple convolutional architecture with spectral normalization and self-attention layers.

Given this setup, the losses \mathcal{L}_{hallu} and $\mathcal{L}_{corrupt}$ are combined to \mathcal{L}_{disc} , a discriminative loss obtained by playing the generator h_H against its discriminator h_D . As a conditional GAN with recurrent elements, the objective this module has to maximize over a complete training sequence is, therefore:

$$H^* = \arg \min_{h_H} \max_{h_D} \mathcal{L}_{disc} + \mathcal{L}_{hallu} + \mathcal{L}_{corrupt} , \quad (4.12)$$

$$\text{with } \mathcal{L}_{disc} = \sum_{t=0}^{\tau} [\log h_D(o_t)] + [\log (1 - h_D(o_t^h))] . \quad (4.13)$$

LSTM. Once the observation features are localized and registered into the allocentric memory (*c.f.* Figure 4.3), the LSTM is used to update the global memory accordingly. Following the original MapNet implementation [98], each spatial location is updated independently to preserve spatial invariance, sharing weights among the LSTM cells.

The occupancy mask b_t of the global memory is updated in a similar manner, using another LSTM with shared-weights to update the memory mask with a binary version of o_t (*i.e.*, 1 for bins containing projected features, 0 otherwise).

Additional Hyperparameters

The following lists complete the architectural and training choices for the experiments.

Architectural Hyperparameters:

- Instance normalization is applied inside the ResNet blocks;
- All Dropout layers have a dropout rate of 50%;
- All LeakyReLU layers have a leakiness of 0.2;

- As a trade-off between computational performance and feature density, the depth n of the global maps is set to either 16 to 32 (as specified for each experiment);
- Image values are normalized between -1 and 1.

Training Hyperparameters:

- Weights are initialized from a zero-centered Gaussian distribution, with a standard deviation of 0.02 ;
- The Adam optimizer [126] is used, with $\beta_1 = 0.5$;
- The base learning rate is initialized at 2×10^{-4} ;
- The overall training of the system is divided in three main phases:
 1. Feature encoder and decoder networks are first pre-trained together for 10,000 iterations;
 2. The complete memorization and anamnesis process (encoder, LSTM, decoder) is then trained for 10,000 more iterations;
 3. The hallucinatory GAN is then added and the complete solution is trained until convergence.

4.4.2 Navigation in 2D Images

In this first part, 2D experiments are proposed, which consider agents exploring images (randomly walking, accelerating, rotating) and observing the image patch in their view field at each step.

Experimental Setup

The two following datasets are selected. A synthetic dataset (named HoME-2D) of indoor 83×83 floor plans is rendered using the HoME platform [29] and SUNCG data [247] (8,640 training and 2,240 test images from random rooms “office”, “living”, and “bedroom”). Similar to Fraccaro et al. [72], the CelebA dataset [155], containing real portrait images of celebrities (cropped and scaled to 43×43 px), is also used. For each dataset, two types of agents are considered, with more or less realistic characteristics.

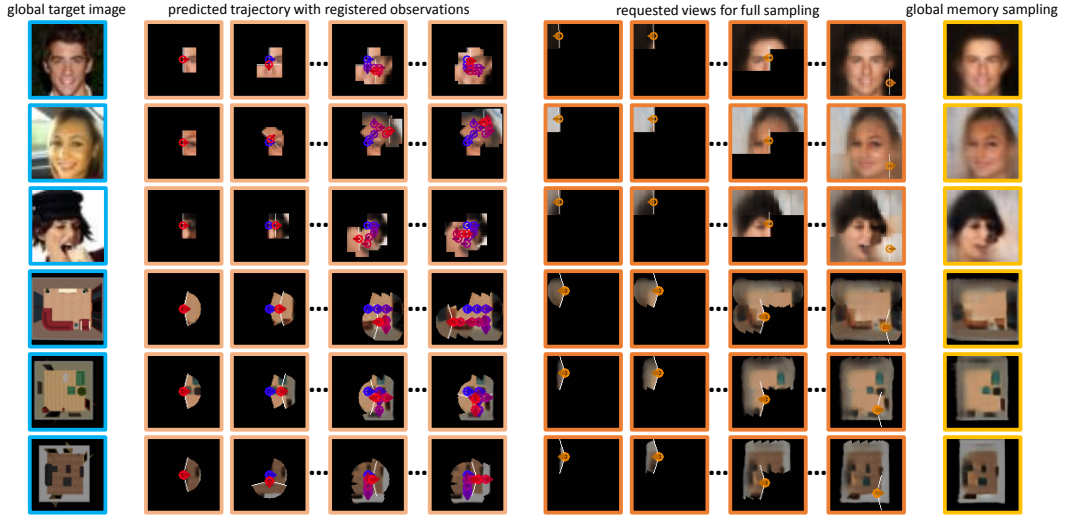


Fig. 4.9: Incremental and direct memory sampling of complete environments from partial observations (on CelebA).

To reproduce Fraccaro et al. [72] experiments, non-rotating agents are first considered. These agents A^s are only able to translate in the four directions and have a 360° view field covering an image patch centered on the agents' position. The CelebA agent A_{cel}^s has a 15×15 px square view field; while the view field of the HoME-2D agent A_{hom}^s reaches 20px away, and is, therefore, circular (in the 41×41 patches, pixels further than 20px are left blank).

To consider more complex scenarios, agents A_{cel}^c and A_{hom}^c are also designed. They can rotate and translate (in the gaze direction), observing patches rotated accordingly. On CelebA images, A_{cel}^c can rotate by $\pm 45^\circ$ or $\pm 90^\circ$ each step, and only observes 8×15 patches in front (180° rectangular view field); while for HoME-2D, A_{hom}^c can rotate by $\pm 90^\circ$ and has a 150° view field limited to 20px.

All agents can move from $1/4$ to $3/4$ of their view field each step. Input sequences are 10 steps long. For quantitative studies, methods have to render views covering the whole scenes w.r.t. the agents' properties.

To the best of our knowledge, no other neural method covers agent localization, topographic memorization, scene understanding and relevant novel view synthesis in an end-to-end, integrated manner. The closest state-of-the-art solution to compare with is the recent GTM-SM project [72], detailed in Subection 4.2.2.

Unlike the proposed method which localizes and registers together the views with no further context needed, GTM-SM requires an encoding of the agent's actions leading to each new observation, as additional inputs. Therefore, for each experiment, the data preparation

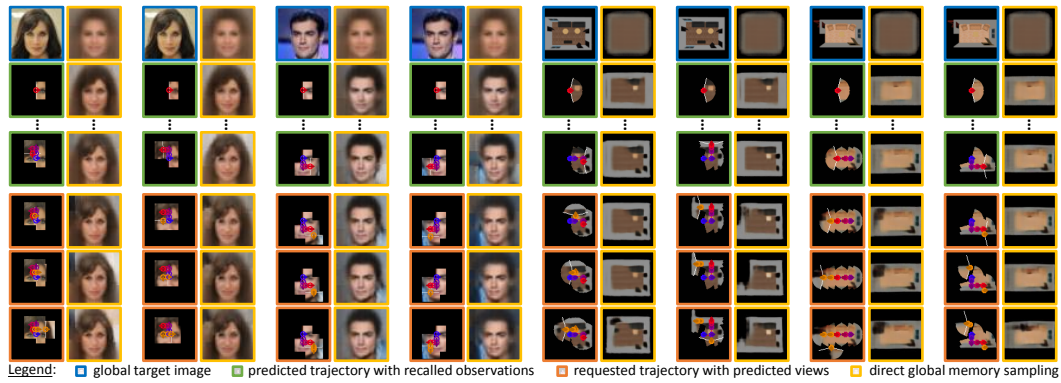


Fig. 4.10: Incremental exploration and hallucination (on 2D use-cases). Scene representations evolve with the registration of observed or hallucinated views (*e.g.*, adapting hair color, face orientation, furniture, *etc.*).

pipeline is adapted for GTM-SM so that the agent returns its actions (encoding the direction changes and step lengths) besides the observations.

Qualitative Results

As shown in Figure 4.7, the proposed method efficiently uses prior knowledge to register observations and extrapolate novel views, consistent with the global scene and requested viewpoints. While an encoding of the agent’s actions is also provided to GTM-SM [72] (guiding the localization), it cannot properly build a global representation from short input sequences, and thus fails at rendering completely novel views.

Moreover, unlike the dictionary-like memory structure of GTM-SM, the proposed method stores its representation into a single feature map, which can, therefore, be queried in several ways. As shown in Figure 4.9, for a varying number of conditioning inputs, one can request novel views one by one, culling and decoding features; with the option to register hallucinated views back into memory (*i.e.*, saving them as “valid” observations to be reused). But one can also directly query the full memory, training another decoder to convert all the features. Figure 4.10 also demonstrates how different trajectories may lead to different intermediate representations, although they converge as the scene coverage increases.

Quantitative Evaluations

Quantitative evaluations are now performed, measuring the methods’ ability to register observations at the proper positions in their respective coordinate systems (*i.e.*, to predict agent trajectories), to retrieve observations from memory, and to synthesize new ones.

Comparison Protocol. At each time step, GTM-SM uses the provided action a_t to regress the agent’s state s_t , *i.e.*, its relative pose in the experiments. For a fair comparison with the ground-truth trajectories, the relative pose sequences predicted by GTM-SM are converted into world coordinates. For that, a least-square optimization process is applied to fit its predicted trajectories over the ground-truth ones, *i.e.*, computing the most favorable transform to apply before comparison (scaling, rotating and translating the trajectories). For the proposed method, the allocentric coordinates are also converted into world units by scaling the values according to the bin dimensions (x_s, z_s) and applying an offset corresponding to the absolute initial pose of the agent.

Metrics. For localization, two metrics commonly used to evaluate SLAM and tracking systems [98, 116, 179, 251] are short-listed. The average position error (APE) is computed as the mean Euclidean distance between the predicted positions and their ground-truths for each sequence. The absolute trajectory error (ATE) is obtained by calculating the root-mean-squared error in the positions of each sequence, after transforming the predicted trajectory to best fit the ground-truth (giving an advantage to GTM-SM predictions given the post-processing operation explained in the previous paragraph).

For image synthesis, the distinction is made between recalling images already observed (*anamnesis*) and generating unseen views (*hallucination*). For each case, two metrics are used. The common L1 distance is computed as the per-pixel absolute difference between the predicted and expected values, averaged over each image. The structural similarity (SSIM) index [282, 284], prevalent in the assessment of perceptual quality [87, 283, 306], is computed over $N \times N$ windows extracted from the predicted and ground-truth images, as follows:

$$\text{SSIM}(x, \bar{x}) = \frac{(2\mu_x\mu_{\bar{x}} + c_1) + (2\sigma_{x\bar{x}} + c_2)}{(\mu_x^2 + \mu_{\bar{x}}^2 + c_1)(\sigma_x^2 + \sigma_{\bar{x}}^2 + c_2)} \quad (4.14)$$

with x and \bar{x} the windows extracted from the predicted and ground-truth images, μ_x and $\mu_{\bar{x}}$ the mean values of the respective windows, σ_x^2 and $\sigma_{\bar{x}}^2$ their respective variance, $c_1 = 0.001^2$ and $c_2 = 0.003^2$ two constants for numerical stability. The final index is computed by averaging the values obtained by sliding the windows over the whole images (no overlapping). For the CelebA experiments, $N = 5$, and for the HoME-2D ones, $N = 13$ (*i.e.*, splitting the observations in 9 windows). Note that the closer to 1 the computed index is, the better the perceived image quality.

Results and Analysis. Table 4.1.A-C shows the comparison on 2D cases. Though GTM-SM leverages the provided actions to infer trajectories, the proposed method is overall more precise, directly using the observations. Moreover, while GTM-SM fares well enough in recovering seen images from memory, it cannot synthesize views out of the observed domain. The proposed pipeline not only extrapolates adequately from prior knowledge, but also generates views which are consistent from one to another (*c.f.* Figure 4.10 showing views stitched into a consistent global image). Note that on a Nvidia Titan X, the whole

Tab. 4.1: Quantitative comparison on 2D and 3D scenes, *c.f.* setups in Subsections 4.4.2-4.4.3 (\searrow the lower the better; \nearrow the higher the better; “u” horizontal bin unit according to 3D setup).

Exp.	Methods	Average Position Error			Absolute Trajectory Error			Anam. Metr.		Hall. Metr.	
		Med. \searrow	Mean \searrow	Std. \searrow	Med. \searrow	Mean \searrow	Std. \searrow	L1 \searrow	SSIM \nearrow	L1 \searrow	SSIM \nearrow
A) A_{cel}^s	GTM-SM	4.0px	4.78px	4.32px	6.40px	6.86px	3.55px	0.14	0.57	0.14	0.41
	Proposed	1.0px	0.68px	1.02px	0.49px	0.60px	0.64px	0.06	0.80	0.09	0.72
B) A_{cel}^c	GTM-SM	3.60px	5.04px	4.42px	2.74px	1.97px	2.48px	0.21	0.50	0.32	0.41
	Proposed	1.0px	2.21px	3.76px	1.44px	1.72px	2.25px	0.08	0.79	0.20	0.70
C) A_{cel}^s	GTM-SM	4.0px	4.78px	4.32px	6.40px	6.86px	3.55px	0.14	0.57	0.14	0.41
	Proposed	1.0px	0.68px	1.02px	0.49px	0.60px	0.64px	0.06	0.80	0.09	0.72
D) Doom	GTM-SM	1.41u	2.15u	1.84u	1.73u	1.81u	1.06u	0.09	0.52	0.13	0.49
	Proposed	1.00u	1.64u	1.41u	1.75u	1.95u	1.24u	0.08	0.61	0.11	0.54
E) AVD	GTM-SM	1.00u	1.12u	1.01u	0.90u	1.02u	0.64u	0.37	0.14	0.48	0.09
	Proposed	1.00u	0.77u	0.79u	0.83u	0.92u	0.53u	0.24	0.29	0.27	0.22

Tab. 4.2: Ablation study on CelebA dataset [155] with agent A_{cel}^c . Removed modules are replaced by identity mappings; remaining ones are adapted to the new input shapes when necessary. LSTM, memory, and decoder are present in all instances (“Localization” is the MapNet module).

Pipeline Modules				Anamnesis Metrics		Hallucination Metrics	
Encoder	Localization	Hallucinatory AE	Culling	L1 \searrow	SSIM \nearrow	L1 \searrow	SSIM \nearrow
\emptyset	\emptyset	\emptyset	\emptyset	0.18	0.62	0.24	0.59
\checkmark	\emptyset	\emptyset	\emptyset	0.17	0.62	0.24	0.58
\checkmark	\checkmark	\emptyset	\emptyset	0.15	0.66	0.20	0.61
\checkmark	\checkmark	\checkmark	\emptyset	0.15	0.65	0.19	0.62
\checkmark	\emptyset	\checkmark	\checkmark	0.14	0.69	0.19	0.63
\emptyset	\checkmark	\checkmark	\checkmark	0.13	0.71	0.17	0.66
\checkmark	\checkmark	\emptyset	\checkmark	0.08	0.80	0.18	0.66
\checkmark	\checkmark	\checkmark	\checkmark	0.08	0.80	0.15	0.70

process (registering 5 views, localizing the agent, recalling the 5 images, and generating 5 new ones) takes less than one second.

Additionally, an ablation study is performed to demonstrate the contribution of each module, and its results shared in Table 4.2. Note that the APE/ATE are not represented, as they stay constant as long as the MapNet localization is included. In other words, the proposed extensions cause no regression in terms of localization.

One can observe from Table 4.2 that localizing and clipping features facilitate the decoding process, *i.e.*, by disentangling the visual and spatial information, thus improving the synthesis quality. Moreover, hallucinating features directly in the memory ensures image consistency, further improving the visual results.

4.4.3 Exploring Virtual and Real 3D Scenes

The capability of the proposed method is now demonstrated on the more complex case of 3D scenes.

Experimental Setup

As done in the previous set of experiments, two different datasets of RGB-D images are selected.

As a first 3D experiment, the Vizdoom platform [292] has been used to record 34 training and 6 testing episodes of 300 RGB-D observations from a human-controlled agent navigating in various static virtual scenes of a vintage video-game, *Doom*. At each time step, the agent can move in one of four directions with variable speed, or rotate by 30° . Poses are discretized into 2D bins of 30×30 game units. Trajectories of 10 continuous frames are sampled and passed to the methods (the first 5 images as observations, and the last 5 as training ground-truths).

Finally, the Active Vision Dataset (AVD) [8] is considered. This dataset is composed of $\sim 20,000$ RGB-D images from various real indoor scenes (most scenes are composed of several rooms each). Images are densely captured every 30cm on horizontal grids covering each scene, as well as every 30° in rotation. As suggested by Ammirato et al. [8], 15 scenes are used for the training of the methods, and 4 are kept for testing (*i.e.*, to have different rooms and content during testing than those trained over). For each scene, 5,000 trajectories of 10 continuous frames are randomly sampled and stored.

For both experiments, 10-frame sequences are passed to the methods—the first 5 images as observations and the last 5 as ground-truths during training. Again, GTM-SM also receives the action encodings. For the proposed method, ψ is defined as a $32 \times 43 \times 43$ tensor for the *Doom* setup and as a $32 \times 29 \times 29$ one for AVD.

Qualitative Results

Though a denser memory could be used to obtain less coarse results, Figure 4.8 shows that the proposed solution is able to register meaningful features and to understand the topography of new scenes simply from 5 partial observations. In comparison, GTM-SM generally fails to properly adapt its variational auto-encoder (VAE) prior, returning image sequences which are often not correlated to the target ones.

Quantitative Evaluations

Finally, the two methods are once again quantitatively compared, adopting the same metrics as in Section 4.4.2. As seen in Table 4.1.D-E, the proposed solution slightly underperforms

in terms of localization in the *Doom* environment. This may be due to the approximate rendering process that VizDoom uses for the depth observations, with discretized values which do not match the distances in game units. Unlike GTM-SM which relies on action encodings for localization, these unit discrepancies affect the proposed observation-based method.

As to the quality of retrieved and hallucinated images, the proposed method shows superior performance. Moreover, to further demonstrate the salient properties of the generated images (despite their lower visual quality), the following metrics have been also considered. First, the Wasserstein metric was computed between the histogram of oriented gradients (HOG) descriptors [55] extracted from the unseen ground-truth images and the corresponding predictions. GTM-SM scored 1.1, whereas the proposed method obtained 0.8 (the lower the better). Second, the saliency maps [34] of ground-truth and predicted images were compared, computing the area-under-the-curve (AUC) metric proposed by Judd et al. [115] and the normalized scanpath saliency (NSS) [202]. GTM-SM scored 0.40 for the AUC-Judd and 0.14 for the NSS, whereas the proposed framework obtained 0.63 and 0.38 respectively (the higher the better for both metrics).

While the current results are still far from visually pleasing, the proposed method is promising, with improvements expected from the adoption of more powerful generative networks.

4.5 Discussion

Research into incorporating read/write mnemonic systems with synthesis and localization capabilities into NN-based autonomous systems is still at its first steps. Therefore, while the proposed solution may be considered closer to a proof-of-concept than a deployable system, its contributions to the field are numerous.

This section concludes the current chapter with a summary of said contributions, as well as some insightful directions for the long-term development of this framework.

4.5.1 Contributions

Existing methods for novel view synthesis have multiple limitations. State-of-the-art solutions often require non-trivial 3D/2D supervision for their training (*e.g.*, computing the reprojection loss) [48, 68, 149], or they assume that the exact agent’s poses or actions are provided for all observations, which is not a practical requirement [40, 72, 211, 305]. Other solutions only consider the task of predicting pixel displacements between different views, without the capability to understand the underlying scenes [257, 262, 309].

This work addresses these issues with a unified framework that incrementally registers and samples complete 2D or 3D scene. The proposed solution builds upon the MapNet system [98], which offers an elegant solution to the registration problem but has no memory-reading capability. In comparison, the presented method not only provides a functional mnemonic system, but also displays superior generative results when compared to the other deep reinforcement learning methods (*e.g.*, [72]). Its key contributions are summarized below:

Mnemonic System for Incremental Scene Understanding and Synthesis. Starting with only scene observations from a non-localized agent (*i.e.*, no location/action inputs, as opposed to many similar methods [72, 219, 305]), novel mechanisms are proposed to update a global memory with encoded features, to hallucinate unobserved regions, and to query the memory for novel view synthesis.

Globally Consistent Registration and Hallucination. Memory updates are done with either observed or hallucinated data. In both cases, the proposed domain-aware operations to update the memory explicitly ensure the global consistency of the representation (*i.e.*, across sampled images) w.r.t. the underlying scene properties. The hallucinatory operation can be interpreted as the expectation of the agent at the current step through the scene and can be leveraged, *e.g.*, in autonomous navigation. In the limit of observing real data at each time step, the method converges to solving the SLAM problem; whereas in the limit of never observing real data, it samples entirely imagined scenes from the prior distribution.

End-to-End Derivable Framework. The proposed framework integrates observation, localization, globally consistent scene learning, and hallucination-aware representation updating to enable incremental scene synthesis. Trainable end-to-end, this framework benefits from the synergy among its modules' sub-tasks.

In short, to the best of our knowledge, the proposed solution is the first end-to-end trainable read/write allocentric spatial memory for scarce visual inputs.

4.5.2 Limitations

Given the multiplicity of the challenges addressed in this chapter and the novelty of this domain in machine-learning, the present work has some limitations worth discussing.

Memory Densification and Computational Footprint. As highlighted through the qualitative experiments, the framework's results tend to be coarse. This statement applies both to the synthesized images and to the predicted trajectories, and can be linked to the structure of the memory.

The agent’s horizontal poses are discretized during ground-projection. The same applies to the projected and encoded observations, with their features being accumulated for each memory bin. Therefore, the localization precision and the density of the memorized content are both tied to the choice of hyperparameters for the global memory. The overall predictions would benefit from a denser memory—in terms of lower word units per bin and in terms of higher value for the feature depth. However, this densification of the memory would also make the localization and registration steps much more compute-intensive.

One possible direction for future work could be, for instance, the implementation of a pyramidal/multi-scale combination of memories for refined registration and synthesis w.r.t. the aforementioned constraints.

Image Quality in Complex Environments. Due to these current limitations affecting the resolution of synthesized images, it seems difficult to see the method applied as-is to the generation of high-definition datasets for downstream tasks (such as the training of recognition models, *c.f.* motivations in Subsection 4.1.2). The recalled and hallucinated images can still be utilized for a variety of applications (next-frame prediction, topography estimation, *etc.*). Moreover, the quality of decoded images could certainly gain from borrowing more recent and specialized generative network architectures advances (*e.g.*, [280]).

Showing superior performance in 2D and 3D experiments with harsh constraints (input scarcity, no oracle, *etc.*) compared to the state-of-the-art, the conceptual pipeline presented in this chapter is promising. This work also emphasizes the difficulty of modeling the complexity of real distributions. Circling back to bridging the gap between real and synthetic representations, one could, therefore, wonder if the common choice of approximating mappings from synthetic to real domains is actually optimal. This observation motivated the final domain adaptation scheme developed during the present thesis and presented in the next chapter.

Reversed Domain Adaptation Scheme for CAD-based Learning

” *Reality is not always probable, or likely.*

— **Jorge Luis Borges**
(Argentine writer, 20th century)

As demonstrated so far, defining a solution to generate samples closer to target real distributions is a difficult task, which require a careful study of relevant phenomena (*c.f.* Chapter 3) or statistical modeling (*c.f.* Chapter 4). However, the scarcer the information on the target domain, the harder it becomes to propose a generalizable solution (*c.f.* Theorem 2.6), which can handle the variability and complexity of real-world data.

Considering the problem of domain adaptation for recognition methods trained on synthetic images from a different angle, this chapter introduces a solution to map unseen target samples into the synthetic domain used to train the task-specific methods [208]. Denoising the data and retaining only the features that these recognition algorithms are familiar with, the solution greatly improves their performance after deployment. As this mapping is easier to learn than the opposite one (*i.e.*, to learn to generate realistic features to augment the source samples), the proposed pipeline can be trained purely on augmented synthetic data, while still performing better than adaptation methods trained over domain-relevant information (*e.g.*, over real images or synthetic images rendered from realistically-textured 3D models). Applying this approach to object recognition from texture-less computer-aided design (CAD) data, a custom generative network is presented, which fully utilizes the purely geometrical information to learn robust features and achieve a more refined mapping for unseen real color images.

Once again, this chapter opens with the motivation behind this new solution in Section 5.1, followed by a study of the state-of-the-art in Section 5.2. The solution itself is described in details in Section 5.3, before being thoroughly evaluated in Section 5.4. Finally, a discussion regarding the overall contributions and limitations of this approach is started in Section 5.5.

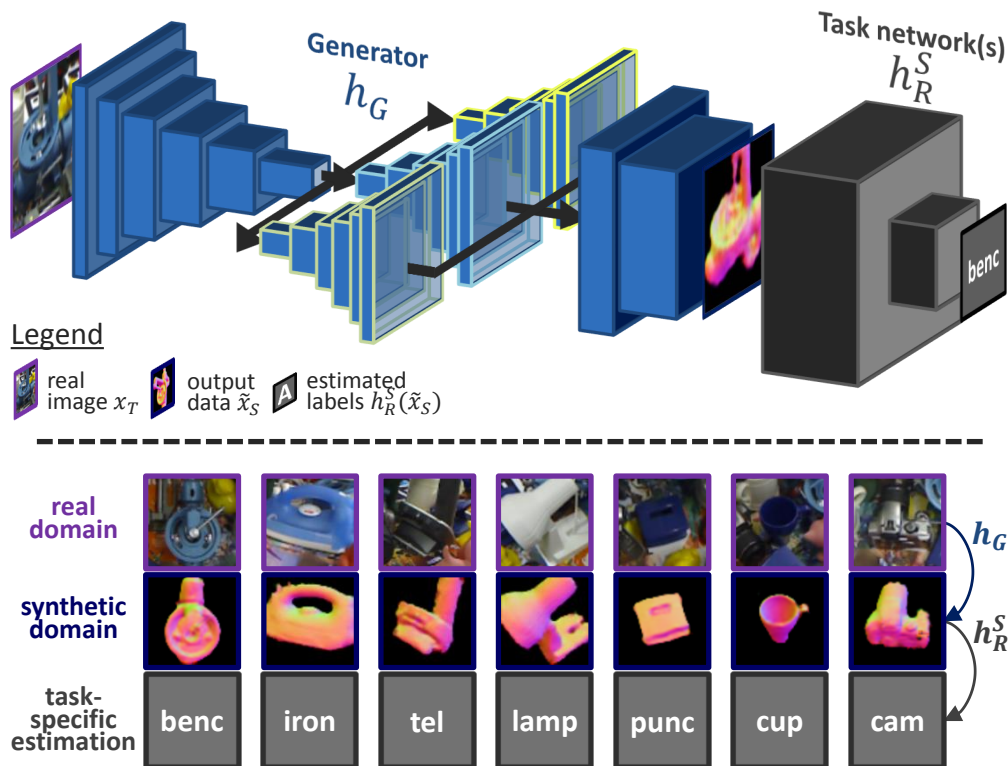


Fig. 5.1: Usage and results of the proposed domain adaptation method. h_G is a custom generative network which maps unseen real data into a discriminative synthetic domain defined in a space \mathcal{X}_S with a marginal distribution P_S^X known during training (*e.g.*, normal maps rendered from CAD models). The pre-processed data can then be handed to recognition methods h_R^S , themselves simply trained on S —a training set sampled *i.i.d.* according to P_S^X —for accurate results despite the realism gap.

5.1 Motivation

To contextualize this final work, this chapter also starts by detailing the challenges and intuitions which motivated the research.

5.1.1 Bridging the Realism Gap from the Other Side

In this chapter, a different approach to domain adaptation is proposed, questioning the dominant paradigm applied to real/synthetic distributions which consists of finding a mapping from the latter to the former.

Complexity of Mapping from Synthetic to Real Domains

As this thesis has been highlighting already, modeling real data is a demanding task. When it comes to partial observations of target objects or scenes, like images captured by sensing

devices, one should take into account both the complexity of the content itself (appearance of the target objects, presence of other elements, scene properties like lighting and motion, *etc.*) and the characteristics of the cameras (intrinsic properties, modality of the captured images, sensing limitations, *etc.*).

Extensive knowledge of both is, therefore, required – be it to develop an exhaustive simulation pipeline or to define a generative model able to mimic the target real distribution. While this knowledge can be manually or algorithmically extracted from relevant samples, one has rarely access to a dataset large enough—and to models complex enough—to understand and reproduce all the phenomena impacting the visual distribution.

Intuitively, however, the opposite process appears much simpler. For instance, learning to remove indiscriminately the background content in real images to bring them closer to the uncluttered synthetic data requires less skills than the creation from scratch of realistic backgrounds to ornate rendered images.

Teaching Recognition Methods in Controlled Environments

Moreover, when building a virtual representation of an object or concept (*e.g.*, a CAD model), efforts are focused first on characteristics key to the target applications. Therefore, while virtual models are inherently simpler, they usually contain all the information actually needed for downstream tasks. For instance, given the three-dimensional (3D) model representing the shape of an unknown object, a human can easily learn to recognize the actual object in real environments.

Most domain adaptation methods are basically defined to add complexity (*i.e.*, as noise) to the synthetic representations (*e.g.*, rendered images or features learned from them), actually cluttering the data with elements which do not directly benefit the semantic recognition (*e.g.*, sensor noise, background, *etc.*). As a result, task-specific models trained in such domain adaptation frameworks learn on noisier data, *smearing* their training in the hope that they will later better handle the harsh complexity of real situations.

The acknowledgement that such a strategy may not be the most *pedagogical* is a key motivation behind this last work. Indeed, it seems sensible to train instead task-specific recognition models on noiseless information so they learn clean discriminative features, and then develop a mapping function from real to synthetic data; rather than to focus on developing or learning pseudo-realistic noise models to train against (though the two can be complementary to bridge the realism gap both ways).

5.1.2 Hardening the Data Scarcity Constraint

Circling back to the key motivation behind the present thesis, this chapter also focuses on bridging the realism gap for the training of recognition methods, when real relevant data samples are unavailable.

Learning from Texture-less CAD Models Only

A critical shortcoming of many state-of-the-art domain adaptation solutions is their assumption that real relevant samples are available (even if unlabeled). This includes the access to a subset of real target images, or to synthetic models too precise for scalable real-world use-cases (*e.g.*, 3D models with realistic textures defined from real observations).

Considering industrial use-cases, one has often only access to 3D CAD models of the components to recognize. Originally meant for the manufacturing process, these 3D models are mostly texture-less, or assigned basic synthetic materials.

Without any information w.r.t. the target real distributions to learn a mapping from (in the image or feature domains), unsupervised domain adaptation methods cannot be applied. The proposed work is, therefore, a successful attempt at reducing the realism gap when assumptions w.r.t. the target real domain are kept to their bare minimum.

Shortcomings of Direct Domain Randomization

State-of-the-art solutions tackling the same level of data scarcity are based on the concept of *domain randomization* [225, 265]. Following this technique, synthetic training images are heavily augmented (*e.g.*, using random textures and random lighting conditions when rendering images, adding random background, *etc.*), in order to teach the recognition models to handle visual variability (even if most of the variations during training are not realistic).

While this flavor of solutions has been yielding promising results given the unfavorableness of considered scenarios, it follows the unintuitive learning scheme pointed out in Subsection 5.1.1. Clean synthetic data is being tampered with during the training of the task-specific recognition methods, which should learn both to extract the relevant features and to ignore the tampered ones. The present work is thus also motivated by the idea to divide these two tasks in order to better conquer them. On one hand, the recognition methods themselves are trained on clean, representative data. On the other hand, in parallel,

another model is trained using domain randomization to map unseen target images closer to the synthetic domain so that the recognition methods can then handle them.

To summarize, the proposed approach is based on the assumptions that real-world images can be mapped to the synthetic domain; and that, in absence of any real training data, this mapping can be learned by recovering the synthetic samples altered by a stochastic noise source. Since this novel method only needs to eliminate noise and retain features, it performs better than usual generative solutions for domain adaptation, which learns the more difficult task of generating complex features to mimic the target data. As long as the synthetic domains contain all relevant features and as long as those features are present in real images, the proposed approach successfully enhances CAD-based recognition in color pictures, as demonstrated through the empirical evaluation in Section 5.4.

5.2 Related Work

Domain adaptation became an increasingly present challenge with the rise of deep-learning methods. Most of the following literature review is, therefore, dedicated to listing such solutions developed to bridge the gap between real and synthetic data. In a second time, convolutional neural network (CNN) methods for shape regression are introduced, as the present work puts an emphasis on the mapping from real color images to synthetic geometrical domains.

5.2.1 Domain Adaptation to Bridge the Realism Gap

Previous chapters have already highlighted how the realism gap is a well known problem for computer vision methods that rely on synthetic data, as the knowledge acquired on these modalities usually poorly translates to the more complex real domain, resulting in a dramatic accuracy drop.

Generation of Realistic Color Images

Several ways to tackle the realism gap issue have been investigated so far, starting with the generation of more realistic images from 3D models. Chapter 3 presented how, in the case of ranging imagery, in-depth studies of sensing devices and other visual phenomena can lead to the development of realistic simulation tools. Such an approach works well for depth (2.5D) data, as the mechanisms impairing the quality of these scans can be rather well reproduced



Fig. 5.2: Qualitative results for *PixelDA* [26] trained with or without realistic texturing of the target objects. The method fails to bridge the realism gap when the latter is too wide.

In case of color data however, the problem lies not in the sensor simulation but in the actual complexity and variability of the color domain (*e.g.*, sensibility to lighting conditions, texture changes with wear-and-tear, *etc.*). This makes it extremely arduous to come up with a satisfactory mapping, unless precise, exhaustive, synthetic models are provided (*e.g.*, by capturing realistic textures). Proper modeling of target classes is however often not enough, as recognition methods would also need information on their environment (background, occlusions, *etc.*) to be applied to real-life scenarios.

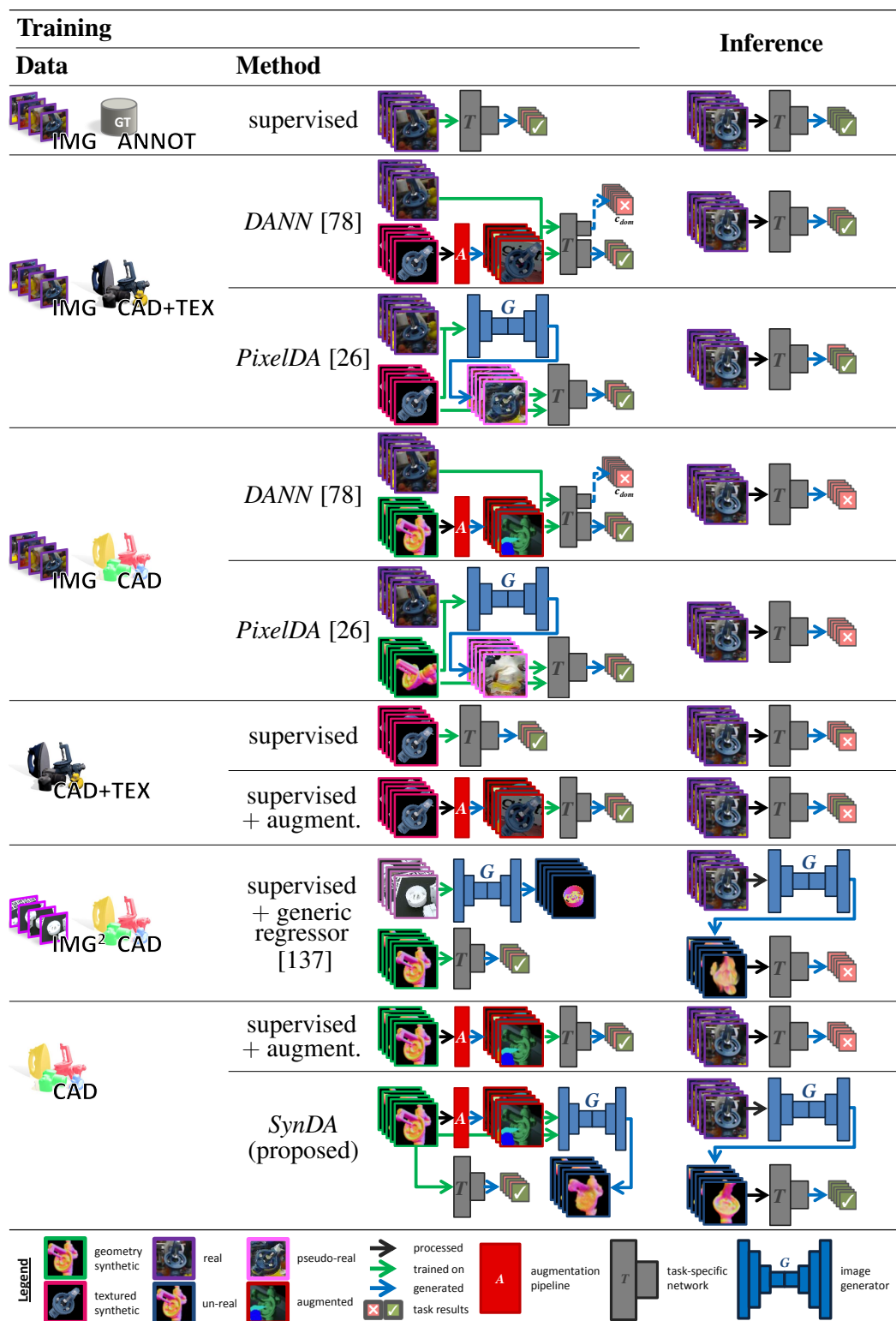
For this reason, and sometimes in complement of simulation tools, recent CNN-based methods are trying to further bridge the realism gap by learning a mapping from rendered to real data, directly in the image domain. Mostly based on unsupervised conditional generative adversarial networks (GANs) [26, 240, 261] or style-transfer solutions [79], these methods still need a set of real samples to learn their mapping.

Training Schemes for Cross-Domain Recognition

Other approaches are instead focusing on adapting the recognition methods themselves, to make them more robust to domain changes. For instance, solutions like *DANN* [78] or *ADDA* [271] are also using unlabeled samples from the target domain along the source data to teach the task-specific method domain-invariant features.

Table 5.1 contains a schematic comparison of such training and testing solutions for recognition tasks, which are addressed in the chapter, depending on the type of data

Tab. 5.1: Visual comparison of recognition schemes, depending on the available modalities w.r.t. training data.



available at training time (relevant real images, related annotations, CAD models, corresponding realistic textures, or real images from different domains). As mentioned already, state-of-the-art domain adaptation methods such as *PixelDA* [26] and *DANN* [78] require real-world data – unlabeled target images but also realistic textures for the 3D models. When such texture information cannot be provided and the resulting synthetic images are far from realistic, both *PixelDA* [26] and *DANN* [78] fail to map the two image domains, as illustrated in Figure 5.2.

Considering real-world and industrial use-cases when only texture-less CAD models are provided, some researchers [225, 265] are compensating the lack of target domain information by training their recognition algorithms against heavy image augmentations or against a randomized rendering engine. The claim behind this *domain randomization* scheme (*c.f.* Subsection 5.1.2) is that with enough variability added to the training set, real data may appear just as another variation to the model. This idea was also adopted by Sundermeyer et al. [258] who are training auto-encoders on augmented synthetic data, then using the resulting encoders inside an object detection and pose estimation pipeline. However, their auto-encoders are trained on single objects, leaving the recognition to a detection module itself trained on real/realistic images.

Considering applications when no real samples nor texture information are available, the proposed method follows the same principle, but applies it to the training of a domain-mapping function instead of the recognition networks. This chapter demonstrates how this different approach not only improves the end accuracy but also makes the overall solution more modular.

5.2.2 Regression of Discriminative Representations

As no textural information is provided for training, the proposed domain adaptation method is applied to the mapping of real cluttered color images into the only prior domain: the geometrical representation of target objects, extracted from their CAD data.

Geometry Regression from Monocular Images

The regression of such view-based shape information (*e.g.*, normal or depth maps) from monocular color images is not a new task in the field of computer vision, and it has been already explored in several works.

Pioneer methods tackled this complex mapping either by using probabilistic graphical models relying on hand-crafted features [104, 152], or by using feature matching between

an RGB image and a set of RGB-D samples to find the nearest neighbors and warp them into a final result [119, 154].

Unsurprisingly, the latest solutions employ CNNs as a basis for their algorithms [61, 137, 222]. Eigen et al. [62] are the first ones to apply a CNN (the popular AlexNet [130]) to this problem, making predictions in a two-stage fashion: coarse prediction and refinement. This approach was further improved by additionally regressing labels and normals, with a refinement step for the final estimation [61].

Advanced Architectures for Multimodal Regression

Another way of improving the quality of predicted 2.5D or normal data is to use neural networks together with graph probabilistic models. Liu et al. [153] use a unified deep convolutional neural field (DCNF) framework based on the combination of a CNN and conditional random field (CRF) to regress depth from monocular color images of various scenes. Their pipeline consists of two sub-CNNs with a common CRF loss layer and yields detailed 2.5D maps.

Building on the previous framework, Cao et al. [36] train the DCNF model jointly for depth regression and semantic segmentation, demonstrating how joint training can improve the overall results. Similarly, Kendall et al. [123] proposed a multi-task Bayesian network approach (including 2.5D regression) which weighs multiple loss functions by considering the uncertainty of each task. Another way of efficiently combining a multi-task output was presented by Xu et al. [296], with the use of so-called distillation modules to supervise and improve the output result.

Unfortunately, all aforementioned methods require real labeled images from the target domain for their training, which is too strong a constraint for real-life scalable applications. The present method does build upon their conclusions [61, 92, 123, 131, 279, 296], making use of a custom cross-modality network with advanced distillation to learn a robust mapping from noisy domains to synthetic ones.

5.3 Methodology: Real-to-Synthetic Mapping through Multi-Modal Distillation

Driven by the necessity of learning only from synthetic data for scalable recognition processes, the following method has been developed to map unseen real samples (*e.g.*, color images) into the noiseless synthetic domain (*e.g.*, normal maps rendered from CAD models)

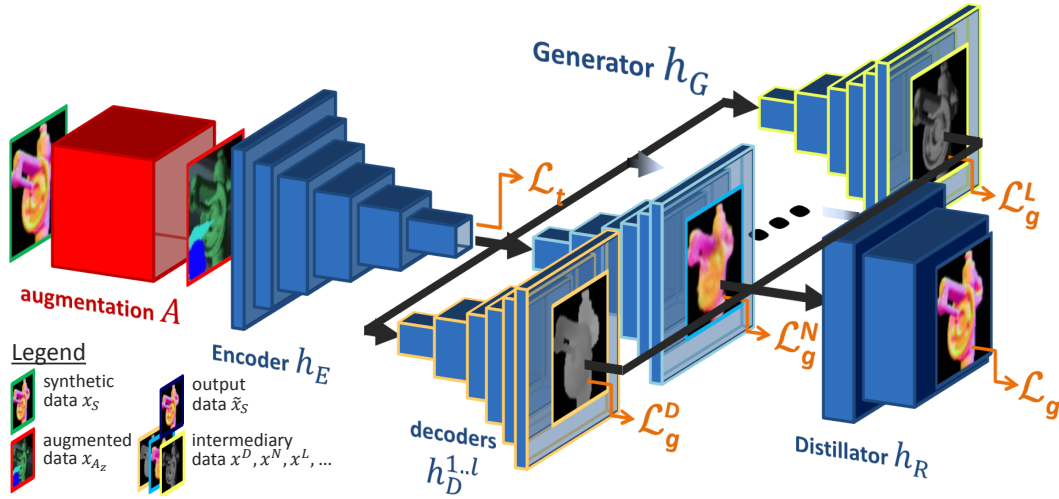


Fig. 5.3: Training of the proposed network h_G . Taking full advantage of available synthetic data, *e.g.*, texture-less CAD models, h_G consists of a custom multi-modal pipeline with self-attentive distillation, trained to recover noiseless geometrical and semantic modalities from randomly augmented synthetic samples (detailed architecture in Figure 5.4).

that the task-specific solutions were trained on (*c.f.* Figure 5.1), to enable recognition. Conceived to itself be trained on synthetic data, the proposed *SynDA* pipeline is able to segment the objects in color images and recover their geometrical information, even under partial occlusion.

Let $S = \{x_{S,i}\}_{i=1}^m$ be a dataset made of a number m of uncluttered, noiseless training samples $x_S \in \mathcal{X}_S$, *i.i.d.* drawn according to a marginal probability distribution $P_S^X = P_S(X)$ and representing the target elements to be recognized (*e.g.*, synthetic images of objects from C semantic classes). Similarly, let T be the set of target real data $x_T \in \mathcal{X}_T$ following an unknown distribution P_T^X and completely unavailable at training time.

Note that samples x_T can also be of a different modality than x_S , *i.e.*, $\mathcal{X}_T \neq \mathcal{X}_S$ (*e.g.*, x_T being color images and x_S normal maps, when no texture were available to render synthetic color images). The only assumption regarding the two domains is that they contain the same semantic content w.r.t. the recognition task (*i.e.*, $P_S(Y|X) \approx P_T(Y|X)$, *c.f.* Subsection 2.3.1).

Finally, let $h_R(x; \theta_R) \rightarrow \tilde{y}$ be any recognition algorithm with a set of parameters θ_R which given a sample x returns an estimate \tilde{y} of a task-specific label or feature $y \in \mathcal{Y}$ (*e.g.*, class, pose, mask image, hash vector, *etc.*). The hypothesis $h_R^S: \mathcal{X}_S \rightarrow \mathcal{Y}$ represents the version (parameterized by θ_R^S) performing the task-specific recognition in the synthetic domain.

Given this setup, the proposed pipeline trains a function $h_G: \mathcal{X}_T \rightarrow \mathcal{X}_S$ purely on synthetic data (and, therefore, with no need for direct human supervision), to learn a mapping from complex instances to their corresponding clean signal (*c.f.* Figure 5.3). To achieve this when

no domain-relevant data is available for training, the following section describes how h_G is trained against a data augmentation pipeline $A : \mathcal{X}_S \times \mathbb{Z}^k \rightarrow \mathcal{X}_T$, *i.e.*, $A(x_S, z) \rightarrow x_{A_z}$, with z a k -dimensional noise vector randomly defined at every training iteration and $x_{A_z}^z$ the resulting noisy data (note that, while A is defined to project samples from \mathcal{X}_S to \mathcal{X}_T , it is not assumed that the augmented images follow P_T^X , *e.g.*, that they look like real samples).

The proposed training scheme assumes that h_G removes the artificially introduced noise z and maps the data such that only the original synthetic signals x_S are retained. Thus, h_G can be seen as a noise filter that removes unneeded elements in input data, and can be also applied over the domain \mathcal{X}_T of real samples as long as synthetic information can be extracted from them.

This work demonstrates that, in the case of CAD-based visual recognition, it is indeed possible to define a new generative method h_G fully utilizing the synthetic modalities, and a complex and stochastic augmentation pipeline A to train h_G against, such that h_G maps real images into the learned synthetic domain with high accuracy. The present work even highlights how this process increases the probability that $h_R^S \circ h_G(x_T)$ is accurate (*i.e.*, $\tilde{y} = y$) compared to $h_R^A(x_T)$, with h_R^A the task-specific algorithm directly trained on data augmented by A . Though the rest of the chapter is focused on CAD-based visual recognition for clarity, the principles behind our domain adaptation solution can be directly applied to other use-cases.

5.3.1 Cross-Domain Mapping via Multi-Modal Distillation

Multi-Modal U-Net Architecture

As demonstrated in previous works tackling multi-task learning [61, 92, 123, 131, 279, 296], it is often advantageous to train a network on several tasks (even when considering only one), as the synergy between the tasks can make each of them perform better, and make the common layers for feature extraction more robust and focused on abstract, cross-modality information.

Such a scheme is, therefore, adopted to guide the proposed generator in its main task of extracting the chosen synthetic features from noisy samples. Not limited to the scarce—sometimes imprecise—annotations of real training datasets, the pipeline can rely on a multitude of different synthetically-rendered modalities. For industrial CAD-based recognition, h_G would learn to map real images into a geometrical domain (normal and/or 2.5D maps), using for sub-tasks the regression of depth and normal maps, semantic or contour mask, *etc.* (*c.f.* Section 5.3.2).

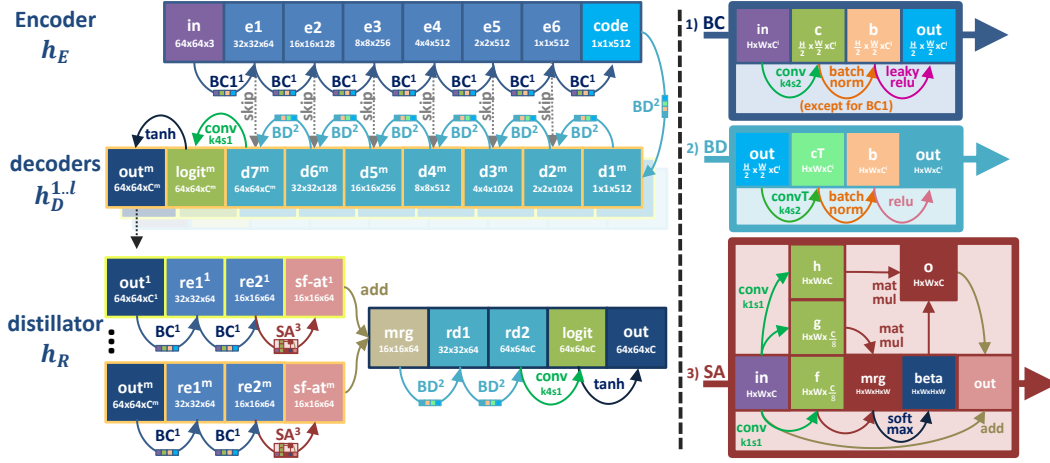


Fig. 5.4: Detailed architecture of the proposed network h_G , on the left. Reused layer blocks (BC for encoding, BD for decoding, SA for self-attention) are detailed on the right. $conv\ k4s2$ stands for a convolutional layer with 4×4 filters and a stride of 2; $convT$ stands for a transposed convolution.

Inspired by previous multi-modal generative pipelines [123, 131, 296], the network is composed of a single convolutional encoder h_E and l decoders h_D , with l the number of sub-tasks. For the rest of the chapter, only the four following sub-tasks are considered: normal regression (performed by h_D^N), depth regression (h_D^D), semantic segmentation (h_D^M), foreground lightness evaluation (h_D^L). Note that it would be straightforward to add more sub-tasks, *e.g.*, contour extraction as done by Xu et al. [296].

According to the proposed solution, each intermediary modality is fully decoded in order to be compared to its synthetic ground-truth. Each generative loss \mathcal{L}_g (L1 distance for real-valued images, cross-entropy for binary masks) is back-propagated through its decoder, then jointly through the common encoder (*c.f.* Figure 5.3).

A triplet loss \mathcal{L}_t , inspired by the work of Wohlhart and Lepetit [286] and Zakharov et al. [300] and presented in Subsection 3.2.1 (*c.f.* Equation 3.2), is optionally added at the network bottleneck to improve the feature distribution in the embedding space, using task-specific metrics to push apart encoded features from semantically-different images, while bringing together features from similar elements:

$$\mathcal{L}_t(h_E) = \sum_{(x_b, x_+, x_-) \in A(S)^3} \max \left(0, 1 - \frac{\|h_E(x_b) - h_E(x_-)\|_2^2}{\|h_E(x_b) - h_E(x_+)\|_2^2 + \epsilon} \right), \quad (5.1)$$

with x_b the input image used as binding anchor, x_+ a positive or similar sample, x_- a negative or dissimilar one, and ϵ the task-specific margin setting the minimum ratio for the distance between similar and dissimilar pairs of samples. For instance, for the task of instance classification and pose estimation (ICPE), Zakharov et al. [300] set $\epsilon = 2 \arccos(|q_b \cdot q_+|)$ if x_b and x_+ are images of the same class, else $\epsilon = n$ (with q_b and q_+ the pose quaternions corresponding to x_b and x_+ , and $n > \pi$ a fixed margin).

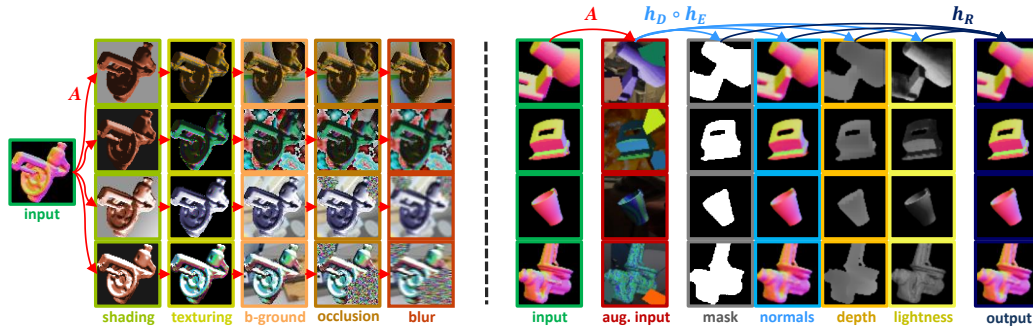


Fig. 5.5: Augmentation and training results. On the *left* is represented the step-by-step transformation of normal maps into complex, random color images by our online augmentation pipeline. On the *right* is shown how h_G is trained on these images, learning to map them back to the noiseless geometrical information.

Further distinguishing our solution from usual multi-modal auto-encoders, skip connections are added from each encoding block to its reciprocal decoding block. As demonstrated with the U-Net work [218] and other solutions making use of it (*e.g.*, [131, 301, 310]), passing high-resolution features from the contracting layers along with the outputs of previous decoding blocks not only improves the training by avoiding vanishing gradients, but also guides the decoding blocks in upsampling and localizing the features. This change results in a clear performance boost, as shown in Table 5.4.

Distillation with Self-Attention

While training the target decoder in parallel to others already improves its performance by synergy, several works [92, 183, 266, 270, 296] demonstrated how one can further take advantage of multi-modal architectures by adding a distillation module on top of the decoders, merging their outputs to distill a final result.

In their work [296], *Pad-Net* authors present several distillation strategies, with the most efficient one making use of attention mechanism [9, 157, 173] to better weigh the cross-modality merging, bringing forward the most relevant features for the final modality.

Using this insight, a new module h_R is defined to refine the target results from the partially re-encoded intermediary modalities by using self-attention computations [44, 195, 275, 304]. As mentioned in Subsection 4.4.1 and formalized in Equation 4.11, this mechanism, proposed by Zhang et al. [304] for image generation (further detailed in Figure 5.4), is used to efficiently model relationships between widely-separated spatial regions. Instantiating and applying this process to each re-encoded modality, the resulting feature maps are summed together and then decoded to obtain the final output. This new distillation process not only allows to pass messages between the intermediary modalities, but also between distant regions in each of them.

The proposed distillator is trained jointly with the rest of the generator, with a final generative loss \mathcal{L}_g (L1 distance here) applied to the distillation results. Not only the whole generator can thus be efficiently trained in a single pass, but no manual weighing of the sub-task losses is needed, as the distillator implicitly covers it (this furthermore suits the considered use-cases, as manual fine-tuning is technically possible only when validation data from target domains are available).

5.3.2 Learning from Purely Geometrical CAD Data

Synthetic Data Generation

The aforementioned architecture has been developed to especially shine for one particular use-case, poorly covered in the literature despite being common in industrial applications: the training of recognition methods on pure 3D CAD data, *i.e.*, without any real relevant images and their annotations, nor captured textures for the 3D models to render realistic images.

Despite the apparent meagerness of the available training data, covering only the geometrical aspects of target classes with no appearance information, it is still possible to render multiple synthetic modalities from the CAD models in order to build a rich annotated dataset, to guide the training of complex generative networks such as our proposed one.

Without any relevant texture information, usual dataset rendering and training methods for the color domain cannot be directly applied. Since only geometrical information is made available, the surface normal and/or depth domains are selected as target modalities for the mapping performed by h_G . For this reason and similarly to other view-based training methods from CAD data [286, 300, 301], a simple 3D engine is set up to generate noiseless normal and depth maps for each target object or scene from a large set of relevant viewpoints (*e.g.*, defined as vertices of an icosahedron centered on the target elements).

This dataset of geometrical mappings is both used as ground-truth for the final outputs of h_G and some of its sub-tasks, and as inputs for the augmentation pipeline A deployed when no color data is available to train h_G .

Online Color Rendering and Augmentation

To train the mapping neural network h_G , an extensive online augmentation pipeline $A(x_S, z) \rightarrow x_{A_z}$ is considered, parametrized by a noise vector z randomly sampled at every call from a k -dimensional finite set \mathbb{Z}^k , with k the number of augmentation parameters. In

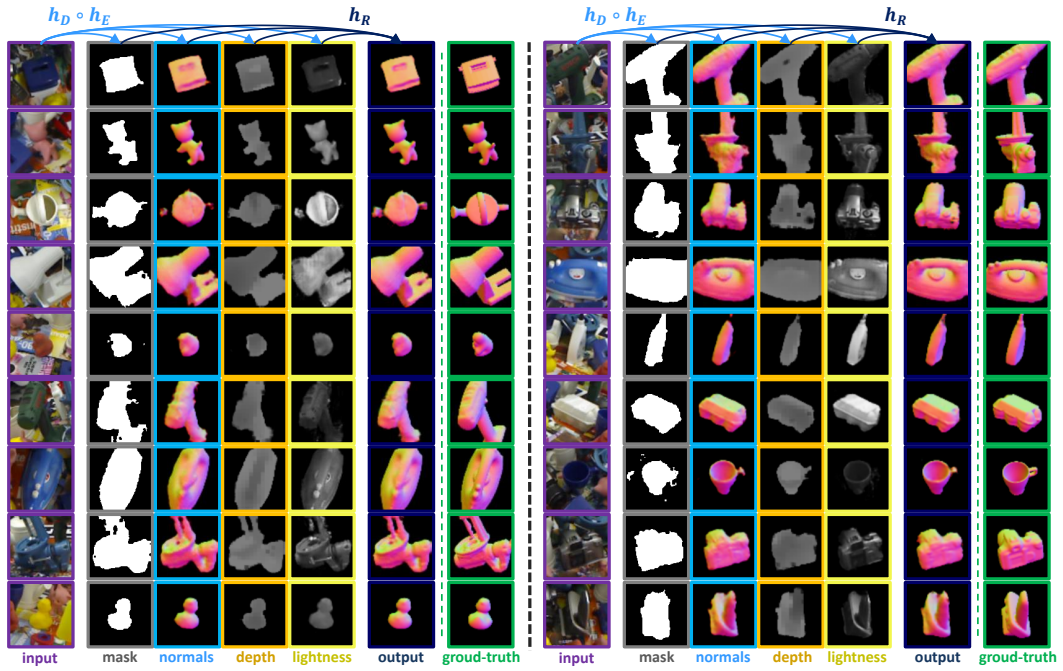


Fig. 5.6: Qualitative results of the proposed generator (intermediary and final mappings), when applying h_G purely trained on synthetic data to real samples, on LineMOD [99].

order to make up for the complete lack of appearance and clutter information, A follows the principle of *domain randomization* [25, 265, 301], *i.e.*, its purpose is to add enough visual variability to the training inputs so that the trained method can generalize to real unseen samples. This means conceiving an augmentation pipeline with k large enough.

By definition, this pipeline is, therefore, modality-specific (*i.e.*, different augmentation transforms are considered if the target domain is composed of RGB images, depth images, or other data modalities). In this present work focusing on visual recognition from RGB images, A first dynamically transforms the input geometrical views into color images through random shading and texturing, before applying further noise and clutter to the images, in order to prepare h_G for the complexity of real data.

Inspired by the literature both in computer vision [41, 49, 243] and computer graphics [22, 201, 204], the following operations are thus composing A (illustrated in Figure 5.5):

Simple Random Shading. The proposed A first takes the provided normal maps and convert them into color images by applying a custom version of the simple Blinn-Phong shading method [22, 204], as described in Algorithm 5.1. Randomly sampling ambient and directional light sources, as well as diffusion and specular color factors for the objects, the provided surface normals are used to compute the diffuse and specular lightness maps through direct matrix products. Since distance information is lost in normal maps, this shading model is simplified by supposing the light sources at an infinite distance, hence

Algorithm 5.1: Approximate Blinn-Phong shading [22] from normal maps.

Input: $X = x^N \in \mathbb{R}^{h \times w \times 3}$ normal map (x^N renamed into X here to clarify the usage of exponents), $L \in \mathbb{R}^3$ directional light vector, $a \in \mathbb{R}^3$ RGB ambient light coefficient, $d \in \mathbb{R}^3$ RGB diffusion coefficient, $s \in \mathbb{R}^3$ RGB specular coefficient, $\hat{s} \in \mathbb{R}$ specular hardness, $F = f_{px} \in \mathbb{R}^2$ pixel focal range used to render x^N (also renamed for clarity).

Output: $Y = x^L \in \mathbb{R}^{h \times w \times 3}$ color lightness map (also renamed for clarity).

```

/* - Simplification #1: we recover an approximate viewer
   vector V from x^N indices and f_px. */
/* - Simplification #2: we suppose the light source at
   +∞ distance, hence the same L for every surface
   point. */
/* - Note: we use Einstein notation for matrix-vector
   operations. */
/* */
/* View vector approximation: */
1 V ← {(j, i, 1)}_{j=0, i=0}^{h, w};
2 V_j ← -(V_j - h/2) / F_j; V_i ← -(V_i - w/2) / F_i;
3 V ← V / ||V||;
/* Computation of half-way vector map: */
4 H ← V + L;
/* Diffuse shading: */
5 D^{ji} ← X^{ji}_k L^k;
/* Specular shading: */
6 S^{ji} ← (X^{ji}_k H^k_{ji})^{\hat{s}};
/* Adding all contributions (given e_{ijc} = e_i ⊗ e_j ⊗ e_c) : */
7 Y^{ijc} ← min(max(a · e_{ijc} + d · D^{ji} e_c + s · S^{ji} e_c, 0), 1);
8 return Y

```

the same light source vector for every surface point. This way, one can easily simulate an infinite number of lighting conditions and obtain the resulting lightness map.

Stochastic Texturing. Given the lack of relevant texture information, random texture maps are procedurally generated using noise functions, such as fractal Perlin noise [201] and Voronoi texturing [288] (*i.e.*, generating random hue and saturation maps which are merged with the lightness map obtained from the shading operation). Downsampled to two-dimensional (2D) vector maps, the original normals are used to index the generated textures; to achieve a more “organic” appearance, with patterns sometimes following some of the shape features.

Background Addition. To simulate cluttered scenes, backgrounds are added to the rendered images, either re-using the previously-introduced noise functions, or using random patches from any publicly available image dataset. Lightness maps from the shading step are furthermore used to homogenize the background brightness.

Algorithm 5.2: Random polygon generation for image occlusion, adapted from the algorithm proposed by Ounsworth [191].

Input: $n \in \mathbb{N}$ number of points defining the polygon, $C \in \mathbb{R}^2$ center of polygon in image plane, $r_\mu \in \mathbb{R}_{>0}$ average radius of the polygon, $\sigma \in \mathbb{R}$ “spikeyness” factor, $\epsilon \in \mathbb{R}$ angular “irregularity” factor (note: all these input values are supposedly sampled from z).

Output: $p = \{p_i \in \mathbb{R}^2\}_{i=0}^{N_{vert}}$ points defining the polygon.

```

/* generation of angle steps: */
1 for  $i \in \{1, \dots, n\}$  do
2   |  $\delta\theta_i \leftarrow \mathcal{U}(2\pi/n - \epsilon, 2\pi/n + \epsilon)$ ;
3 end
/* steps normalization: */
4 for  $i \in \{1, \dots, N_{vert}\}$  do
5   |  $\delta\theta_i \leftarrow 2\pi \frac{\delta\theta_i}{\|\delta\theta\|_2}$ ;
6 end
/* polygon points generation: */
7  $\theta_1 \leftarrow \mathcal{U}(0, 2\pi)$ ;
8 for  $i \in \{1, \dots, n\}$  do
9   |  $r \leftarrow \mathcal{N}(r_\mu, \sigma)$ ;
10  |  $p_i \leftarrow C + \{r \cos(\theta_i), r \sin(\theta_i)\}$ ;
11  |  $\theta_{i+1} \leftarrow \theta_i + \delta\theta_i$ ;
12 end
13 return  $p$ 

```

Random Occlusion. Occlusions are introduced to further simulate clutter, but also so that h_G can learn to recover hidden or lost geometrical information. Based on the work of Ounsworth [191] (who generates 2D obstacles for a drone moving planning simulation), occluding polygons are generated by walking around the image, taking random angular steps and random radii at each step; then by painting it on top of the images with color noise (*c.f.* Algorithm 5.2). The complexity of the polygons is controlled by two parameters: σ (“spikeyness”), which controls how much point coordinates vary from the chosen average radius r_μ , and ϵ (“irregularity”), which sets an error to the default uniform angular distribution of the polygon points.

Blur. To reproduce possible motion blur or unfocused images, Gaussian, uniform or median blur is applied with variable intensity.

To maximize the training variability, A is built to run in parallel of GPU-based trainings (*online*), providing new randomized samples every iteration. Note that the proposed Algorithm 5.1 differs from traditional shading methods for this reason. It was edited so that it does not perform on 3D input data but instead on normal maps—themselves pre-rendered from 3D models. Therefore, the computation-heavy task of projecting 3D models into 2D images (normal maps here) can be conveniently performed in a preliminary step, before (and not in parallel of) the training itself. Computation-savvy, Algorithm 5.1—as well as

the other aforementioned operations composing A —can thus perform in parallel of the training loop (*online* augmentation). As a result, the CNNs undergoing this training can receive at each iteration different images (unlike with *offline* augmentation pipelines used to generate a fixed training dataset beforehand), conveniently maximize the variability of the training data and reducing the risks of overfitting.

Of course, additional operations could be considered to make A either more complex or more adapted to the target domains if some of their aspects are known at training time. As illustrated in the following section, the proposed generic augmentation pipeline already enables the training of robust methods for color-to-normal mapping.

5.4 Experiments and Results

Applying *SynDA* to CAD-based recognition tasks, the method is evaluated on two different tasks of localized instance classification and pose estimation, opting for well-known algorithms on datasets commonly used in this domain [26, 286, 301]. After some concise qualitative observations, extensive qualitative experiments are presented, comparing the performance of *SynDA* to state-of-the-art solutions depending on the available training modalities, and providing an ablation study.

5.4.1 Implementation Details

This first subsection contains more in depth details on the architecture and parameters of the network, and on the augmentation pipeline.

Network Architecture

Figure 5.4 already presents to the readers with an exhaustive overview of the proposed state-of-the-art architecture, layer by layer. It is implemented in Python using the TensorFlow framework [1].

Layer Parameters:

- All Convolution layers have 4×4 filter kernels;
- All Dropout layers have a dropout rate of 50%;
- All LeakyReLU layers have a leakiness of 0.2;

- Input and output images are 64×64 px, normalized between -1 and 1 .

Training Parameters:

- Weights are initialized from a zero-centered Gaussian distribution, with a standard deviation of 0.02 ;
- The Adam optimizer [126] is used, with $\beta_1 = 0.5$;
- The base learning rate is initialized at $2e^{-4}$.

Augmentation Pipeline Details

The augmentation operations are also implemented using TensorFlow. As detailed in the following paragraphs, each parameter of the augmentation pipeline is randomly picked according to a predefined distribution (with \mathcal{B} – Bernoulli, \mathcal{U} – Uniform, and \mathcal{N} – Gaussian), at each iteration (the set of all randomly sampled parameters forming z). The pipeline receives as input synthetic normal images from the 3D models of target objects, rendered using *OpenGL* [235].

Simple Random Shading. Lighting parameters are sampled using uniform distributions, *e.g.*, $a \sim \mathcal{U}(0.05, 0.3)^3$, $d \sim \mathcal{U}(0.1, 0.8)^3$, $s \sim \mathcal{U}(0, 0.1)^3$, $\hat{s} \sim \mathcal{U}(0.9, 1.1)$, *etc.* (*c.f.* Algorithm 5.1).

Stochastic Texturing. Perlin noise [201], cellular noise [288], and white noise are used, sampled from the uniform distribution $\mathcal{U}(0.0001, 0.9)$, to obtain hue and saturation maps.

Background Addition. Backgrounds are added to the synthetic images, by randomly cropping and resizing color images from a public dataset (COCO [150] here).

Random Occlusions. For the experiments defined thereafter, the following sampling distributions are used for the random occlusions: $C_y \sim \mathcal{B}(\mathcal{U}(0, h/4), \mathcal{U}(h/4, h))$, $C_x \sim \mathcal{B}(\mathcal{U}(0, w/4), \mathcal{U}(w/4, w))$, $r_\mu \sim \mathcal{U}(10, m/4)$ (with $m = \min(h, w)$), $n \sim \mathcal{U}(3, 10)$, $\epsilon \sim \mathcal{U}(0, 0.5)$, and $\sigma \sim \mathcal{U}(0, 0.5)$.

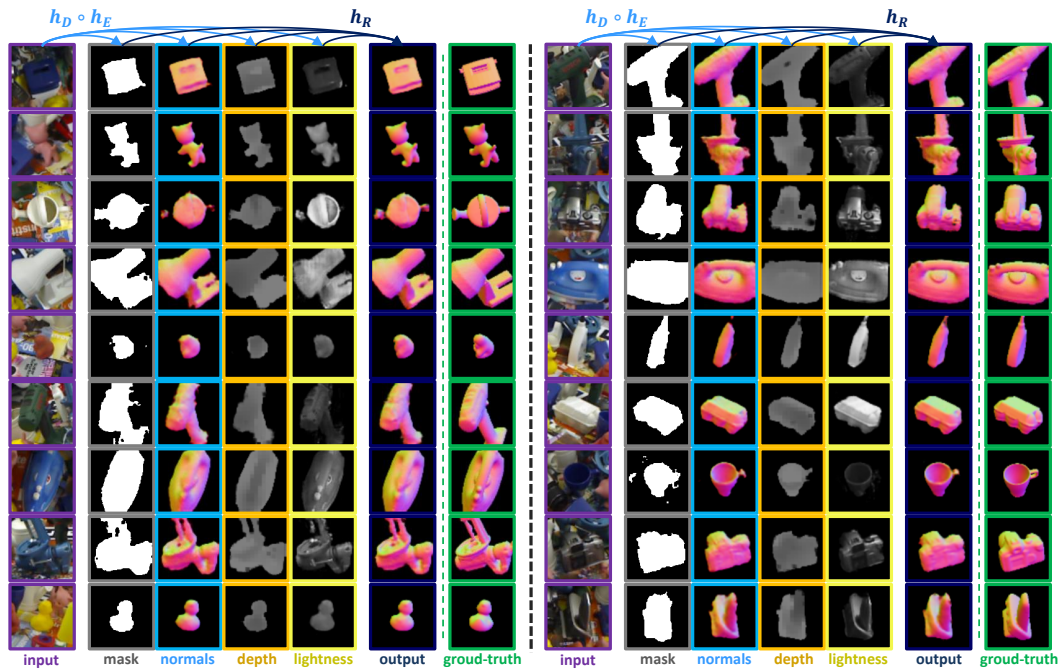


Fig. 5.7: Qualitative results of the proposed generator (intermediary and final mappings), when applying h_G purely trained on synthetic data to real samples, on T-LESS [102].

5.4.2 Experimental Setups

Instance Classification on T-LESS

Task. As a preliminary experiment, the task of localized classification is considered, performed on T-LESS [102], a dataset of industrial objects with texture-less CAD models and red-green-blue and depth (RGB-D) images from different complex scenes. Strong textural and geometrical similarities between the objects, as well as heavy occlusions, make it a challenging dataset for geometry-based instance classification (IC).

Data. The first three scenes and their eleven objects are selected (objects numbered 2, 5, 6, 7, 8, 11, 12, 18, 25, 29 and 30), building a set of 7,514 red-green-blue (RGB) test patches of 64×64 px, representing objects occluded up to 60%. The real color images are split 50/50 into a test set S_{test} , and a training set named S for simplicity. The latter is used for the training of some of state-of-the-art methods which require real data.

The synthetic training data S (normal, depth, and semantic maps) are generated with a basic 3D engine (while lightness maps are obtained from A), following the procedure described in Section 5.3.2. Images are rendered from the CAD models from viewpoint defined by the vertices of an subdivided icosahedron of radius 600mm centered on target objects, as done by Wohlhart and Lepetit [286] and Zakharov et al. [300] in their experiments. This resulted in the generation of 642 synthetic samples per object.

Finally, a dataset S_{na} is also used to train some methods $SynDA$ is compared with. It contains real images, but irrelevant to the task (to demonstrate the effect of other domain shifts besides the realism gap).

Recognition Method. Considering the task-specific recognition algorithm fixed and provided, a ResNet [96, 97] with 9 residual blocks is used.

Instance Classification and Pose Estimation (ICPE) on LineMOD

Task. The ICPE task performed on LineMOD [99] is addressed in a second time. Following the same setup as in Section 3.4.3 (for the evaluation of *DepthSynth*), this dataset (which contains fifteen 3D mesh models of distinctive textured objects, along with their RGB-D sequences and camera poses) is advantageously used to demonstrate how texture information is too often taken for granted in CAD-based application, and how its absence can heavily impact usual methods (*e.g.*, in industrial settings).





Data. As done in the previous experiment, the +15,000 real LineMOD images (cropped to 64×64 px) are split into a testing dataset and a training one (only for comparative methods requiring real training data).

Similarly, the synthetic images are once again generated using an icosahedron of radius 600mm, with 3 consecutive subdivisions. However in this case, only the upper part of the icosahedron is kept since all objects are captured from above in real image sequences. Furthermore, in-plane rotations are added for each vertex, parametrized from -45° to 45° with a stride of 15° .

However, LineMOD has four symmetric objects (*bowl*, *cup*, *eggbox*, and *glue*), meaning that images from different viewpoints may contain the same visual content, which may confuse algorithms tasked to regress the poses. While some works simply remove these ambiguous elements to facilitate the evaluation [26, 27], others only constrain the real views by keeping the unambiguous poses for these four objects (pruning the sampling vertices to take into account the rotation or plane invariance) [286, 300, 301]. The latter solution is opted for in this work, to highlight the generalization capabilities of *SynDA* w.r.t. the number of objects. All in all, 2,359 images are generated for each of the eleven irregular objects, 1,239 for the three plane-symmetric objects (*cup*, *glue*, and *eggbox*), and 119 for the axis-symmetric *bowl*.

Moreover, a dataset S_{tex} is rendered, containing realistic color images from the textured LineMOD 3D models to train some comparative state-of-the-art methods on (this applies only to LineMOD, as T-LESS industrial 3D models are not textured).

Tab. 5.2: Quantitative comparison of recognition pipelines, depending on the available training data, for the task of localized instance classification on T-LESS [102] with h_R ResNet-9 network [97] (refer to Table 5.1 for a visual description of each method).

Training		Classification accuracy	
Data	Method		
	$h_R(x_T) \rightarrow \tilde{y}$	(\emptyset)	99.34%
	$h_R^d(A(x_S); x_T) \rightarrow \{\tilde{y}, \tilde{c}_{dom}\}$	(DANN)	60.58%
	$h_G^{pix}(x_S, x_T) \rightarrow \tilde{x}_T, h_R(\tilde{x}_T) \rightarrow \tilde{y}$	(PixelDA)	63.12%
	$h_G^{rgb2d}(x_{T_{NA}}) \rightarrow \tilde{x}_S, h_R(x_S) \rightarrow \tilde{y}$	(Iro <i>et al.</i>)	36.03%
	$h_R(A(x_S)) \rightarrow \tilde{y}$	(\emptyset)	53.81%
	$h_G(A(x_S)) \rightarrow \tilde{x}_S, h_R(x_S) \rightarrow \tilde{y}$	(SynDA)	71.78%

Recognition Method. As in Subsection 3.4.3, to further demonstrate that the proposed method is tailored neither to a dataset nor to a specific recognition method and features, a different algorithm is selected for the task: the so-called *triplet* CNN [286, 300], which uses the aforementioned triplet loss to map images to an embedding space which enforces separation for distinct classes and poses.







5.4.3 Qualitative Observations

Qualitative results can be found in Figures 5.1, 5.5, 5.6, and 5.7. For both datasets, the proposed method clearly learns to recover the clean geometrical features of target objects in unseen real images (*c.f.* Figure 5.6), even though it has been trained with no information w.r.t. the real target domain (*c.f.* Figure 5.5).

The monochrome appearance of T-LESS objects may make the task easier; but as this information is not known during training, h_G is still trained on random noisy textures, and yet manages to map the real samples and even to recover occluded parts. As demonstrated on LineMOD, the proposed solution indeed learns to ignore visual properties such as textures to retain the synthetic features, using the prior CAD data.

GAN-based domain adaptation methods such as *PixelDA* [26] fail to learn their opposite mapping when only geometrical properties are provided, as shown in Figure 5.2. Indeed, learning both to add clutter and assign the proper texture to each class is a much more complex task, which would require further supervision (for instance, the foreground-similarity loss of *PixelDA* cannot be used in this setting to guide the network).

Tab. 5.3: Quantitative comparison of recognition pipelines, depending on the available training data, for the task of localized instance classification and pose estimation on LineMOD [99] with h_R triplet CNN [286, 300].

Training			Angular error		Classification accuracy
Data	Method		Median	Mean	
	$h_R(x_T) \rightarrow \tilde{y}$	(\emptyset)	9.50°	12.42°	99.72%
	$h_R^d(A(x_{S_{tex}}); x_T) \rightarrow \{\tilde{y}, \tilde{c}_{dom}\}$	(DANN)	14.33°	30.45°	89.84%
	$h_G^{pix}(x_{S_{tex}}, x_T) \rightarrow \tilde{x}_T, h_R(\tilde{x}_T) \rightarrow \tilde{y}$	(PixelDA)	15.38°	35.17°	91.06%
	$h_R^d(A(x_S); x_T) \rightarrow \tilde{y}, \tilde{c}_{dom}$	(DANN)	43.63°	68.59°	40.13%
	$h_G^{pix}(x_S, x_T) \rightarrow \tilde{x}_T, h_R(\tilde{x}_T) \rightarrow \tilde{y}$	(PixelDA)	95.14°	97.36°	35.39%
	$h_R(x_{S_{tex}}) \rightarrow \tilde{y}$	(\emptyset)	88.62°	92.35°	43.62%
	$h_R(A(x_{S_{tex}})) \rightarrow \tilde{y}$	(\emptyset)	70.18°	84.22°	49.11%
	$h_G^{rgb2d}(x_{T_{NA}}) \rightarrow \tilde{x}_S, h_R(x_S) \rightarrow \tilde{y}$	(Iro <i>et al.</i>)	52.43°	71.69°	41.49%
	$h_R(A(x_S)) \rightarrow \tilde{y}$	(\emptyset)	41.23°	67.50°	34.38%
	$h_G(A(x_S)) \rightarrow \tilde{x}_S, h_R(x_S) \rightarrow \tilde{y}$	(SynDA)	13.37°	27.46°	91.28%

Finally, one can observe the empirical improvements between the intermediary normal maps (directly from decoder h_D^N) and the refined outputs after self-attentive distillation, both in terms of segmentation and internal details. As mentioned in Section 5.3.1, one could easily add or replace intermediary modalities (for instance, regressing the objects lightness maps may not seem fully relevant, though it can be used to provide the latest layers of the network with information from the original color domain).

5.4.4 Quantitative Evaluation on Recognition Tasks

Given the two predefined evaluation tasks, the performance of the proposed pipeline is also quantitatively evaluated through a comparative study and an ablation one.

Comparison with other Domain Adaptation Approaches

First, a comparison is performed with well-known state-of-the-art methods, depending on the available training data (real images, corresponding annotations, CAD models, corresponding realistic textures, or real images from a different domain).

For each setup, multiple instances of the same task-specific network (ResNet for IC on T-LESS, triplet CNN for ICPE on LineMOD) are used, trained alone against the augmentation pipeline A presented in Subsection 5.3.2 (with texturing augmentation disabled for pre-textured data), or trained along with some auxiliary generators or sub-networks for domain adaptation (*e.g.*, for *PixelDA* [26] or *DANN* [78]; for h_R used with a pre-trained monocular-*RGB-to-depth* generator h_G^{rgb2d} [137]; or for *SynDA*).

Tab. 5.4: Architectural ablation study, considering the instance classification task on LineMOD.

Encoder	Decoders				Distill.	Layers		Losses		Angular error		Classification accuracy
	h_D^N	h_D^D	h_D^M	h_D^L		h_R	\overrightarrow{SA}	\overrightarrow{skip}	$\mathcal{L}_g^{1..l}$	\mathcal{L}_t	Median	
✓	✓						✓	✓		15.75°	32.80°	87.35%
✓	✓						✓	✓	✓	15.76°	33.76°	88.04%
✓	✓	✓			✓		✓	✓	✓	14.32°	30.31°	89.00%
✓	✓		✓		✓		✓	✓	✓	14.48°	30.71°	89.32%
✓	✓	✓	✓		✓		✓	✓	✓	14.22°	29.26°	89.67%
✓	✓	✓	✓	✓			✓	✓	✓	14.66°	30.83°	88.59%
✓	✓	✓	✓	✓	✓			✓	✓	16.07°	33.22°	87.69%
✓	✓	✓	✓	✓	✓		✓	✓	✓	14.43°	29.56°	90.38%
✓	✓	✓	✓	✓	✓		✓	✓	✓	13.37°	27.46°	91.28%

For both tasks, one can consistently observe the positive impact of *SynDA* on recognition, as shown in Tables 5.2 and 5.3. Despite being trained on the scarcest data, with the largest domain gap, the proposed generator h_G brings the performance of the task-specific methods h_R above other solutions trained on more relevant information. The accuracy improvement appears even more clearly for the pose regression task, as *SynDA* precisely recovers geometrical features.

It also appears that decoupling data augmentation and recognition training is beneficial, as illustrated by the accuracy difference between the last two lines of each table. This follows the initial intuition on the logic of teaching task methods in the available clean synthetic domain, while learning in parallel a mapping to project real data into this prior domain (cf Subsection 5.1.1). Furthermore, this separation makes it straightforward to train new task-specific methods, with h_G ready to be plugged on top (modularisation).

Architecture Validation through Ablation

Table 5.4 presents the results of an extensive ablation study done on the proposed network architecture. By consolidating several state-of-the-art works on generative networks [123, 131, 296, 304], a robust architecture was developed to tackle extreme domain mappings (e.g., real RGB images to synthetic normal maps).

As mentioned in Section 5.4.3, one can observe how the addition of decoders for auxiliary tasks improves the final output, by synergy. The inclusion of self-attention mechanism (*SA* layers) in the distillation module further enhances this effect, weighting the contribution of features between intermediary modalities, but also between distant internal regions. Finally, the benefits of passing messages directly between the encoder blocks and their opposite blocks for each decoder h_D , through the use of *skip* layers (c.f. U-Net architecture [131,

218, 310]), is clearly highlighted in the table, as well as the use of a triplet loss \mathcal{L}_t at the bottleneck to improve the quality of the embedding space.

All in all, the network relies on a powerful multi-task architecture, structured to tackle the challenges of real-to-synthetic mapping, by utilizing any synthetic modalities available to learn robust features. One could easily build on this solution by considering additional or more use-case relevant sub-tasks (*e.g.*, contour regression, part segmentation, *etc.*).

5.5 Discussion

Addressing the problem caused by the realism gap in CAD-based visual recognition, this final work proposes to bridge the two domains through a neural-based mapping antipodal to the one performed by state-of-the-art solutions. Its contributions are summarized in this section, ending with some possible directions for future work.

5.5.1 Contributions

Applied to several CAD-based recognition tasks and making use of an advanced generative network, *SynDA* outperforms other supervised or unsupervised methods. For the challenging task of localized instance classification and pose estimation (*e.g.*, on RGB LineMOD data), the proposed solution more than doubles the angular and class accuracy compared to other methods trained on synthetic data, and even surpasses previous domain adaptation methods requiring real data. Overall, the following contributions are made.

Synthetic Modality Regression for Domain Adaptation. A novel framework is defined to learn a mapping from unseen real data to relevant synthetic domains, denoising and recovering the information needed for further recognition. Therefore, this solution not only covers the real-synthetic gap, but also takes care of cross-modality mapping. More specifically, this chapter highlights how color images can be mapped to normal maps, to facilitate the tasks of pose-regression and classification in absence of reliable textural information for training.

Decoupling of Domain Adaptation and Recognition. Most domain adaptation schemes constrain the training of the recognition methods, by adding pseudo-realistic or noisy features to their training set, editing their architecture or losses, *etc.* In the proposed framework, task-specific algorithms simply learn on available, relevant synthetic data (*e.g.*, clean normal maps from CAD models), while the proposed network h_G is trained separately on augmented data to map them back into the selected synthetic domain. This decoupling makes training more straightforward, and makes it possible to use h_G along with any number of recognition

methods. Furthermore, better results are observed compared to recognition methods directly trained on augmented data. Results even compare to solutions using real information for training.

Performance in Complete Absence of Real Training Data. Domain adaptation approaches usually assume the realism gap to be already partially bridged, requiring access to some target domain images or realistic synthetic models (*e.g.*, textured 3D data). Opting for recognition tasks with texture-less CAD data for only prior, this chapter shows how our pipeline can be trained on purely synthetic data and still generalize well to real situations. To that purpose, an extensive augmentation pipeline is leveraged, used as a noise source applied to training samples so the solution learns to denoise and retain the relevant features.

Multi-Task Network with Self-Attentive Distillation. The one advantage of synthetic models such as CAD data is the possibility to easily extract various precise modalities and ground-truths to learn from. Consolidating several state-of-the-art works [123, 131, 296, 304], a custom generator is, therefore, defined with multiple convolutional decoders for each relevant synthetic modality (*e.g.*, normal maps, semantic masks, *etc.*), and a distillation module on top making use of self-attention maps to refine the final outputs.

Mirroring the work presented in Chapter 3, the domain-mapping method developed in this chapter could be applied to problems beyond visual recognition (*e.g.*, image denoising, augmented reality, *etc.*).

5.5.2 Limitations

Interestingly, the parallel with the opposite *DepthSynth* method presented in Chapter 3 also applies to their acknowledged limitations.

Importance of the Augmentation Operations. In spite of its ability to render realistic training images, *DepthSynth* is limited by the knowledge w.r.t. the target real domains (*c.f.* Subsection 3.5.2), and so is *SynDA*. For example, in the former work, if no image of the target scenes are provided, the simulation pipeline can only fill the synthetic images with backgrounds generated from random noise or picked from irrelevant datasets. *SynDA* has to rely on similar artifices to generate the augmented images it is learning from. Therefore, the quality of the real-to-synthetic mapping that the method learns is only as good as the quality of the augmentation pipeline, and its relevance w.r.t. the target domains.

The focus of this chapter was put on recognition tasks from RGB images, and assumptions regarding the target domains were kept minimal (RGB patches representing the target

objects). Given this setup, a generic and optimized augmentation pipeline has been developed. Yet, the selected operations are not meant to be exhaustive, and a variety of alternative solutions could be considered to generate the noisy data (*e.g.*, more advanced computer graphics engines).

As proposed for *DepthSynth*, it could also be interesting to explore solutions to integrate into the augmentation pipeline any prior w.r.t. the target real domains, *e.g.*, when the data scarcity constraint is relaxed.

Impact of Data Model Quality. To push the comparison with *DepthSynth* further, both methods are also dependent on the quality of the provided synthetic data models (*e.g.*, the 3D models representing the target objects). If the provided models deviate from the actual target objects (*e.g.*, if the real objects is damaged or if it has non-rigid parts), the generator h_G may fail to recover the representation from real images, impacting the performance of the downstream recognition methods too.

The recognition of deformable entities is a long-standing and well-documented problem in computer vision [24, 169, 181]. Though it goes beyond the scope of the current thesis, it could be interesting to evaluate the feasibility of including some developments in that domain into the present work, to train a more robust mapping function¹.

As it is, the proposed framework can be successfully applied to the training of a wide panel of recognition solutions for CAD-based uses-cases.

¹For industrial applications, when considering manufactured objects with movable parts, a direct solution is the generation of a larger dataset of synthetic images, discretizing the various states that the objects and its parts can take.

Discussion and Future Work

” *No book can ever be finished. While working on it we learn just enough to find it immature the moment we turn away from it.*

— **Karl Popper**

(Philosopher, in “The Open Society and Its Enemies: The spell of Plato”, 1971)

To conclude this document, a brief summary is first proposed in Section 6.1. It is followed by some higher-level considerations w.r.t. future work in Section 6.2. Section 6.3 ends the dissertation with some final, more philosophical thoughts.

6.1 Summary

This thesis has been focused on advancing the frontier of visual intelligence, proposing a variety of solutions for machines to gain a practical understanding for high-level recognition tasks, despite the scarcity or irrelevance of the available observations.

6.1.1 Significance

As current recognition algorithms require a large number of representative samples to learn their task properly, having to tediously gather and annotate such datasets represents a strong constraint, hindering the systemic adoption of computer vision solutions.

Many previous projects anchored in industrial contexts have been leveraging synthetic images—rendered from computer-aided design (CAD) models often pre-available in the industry—as proxy data to train convolutional neural networks (CNNs). However, empirical and theoretical studies have demonstrated how the discrepancies between synthetic and real domains heavily impairs the performance of the resulting recognition systems (*c.f.* Chapter 2). Furthermore, the assumption that relevant three-dimensional (3D) representations are available does not always hold, *e.g.*, for systems routinely confronted to news objects or scenes. State-of-the-art solutions considering data scarcity and realism gap

often make further assumptions—*e.g.*, availability of some real relevant samples, exhaustiveness of the available synthetic representation, *etc.*—that greatly limit their application to real scenarios.

Addressing consequential industrial use-cases, this thesis offers effective and novel solutions to build and leverage synthetic representations, in order to train more robust and versatile recognition systems.

6.1.2 Overall Contribution

The main solutions proposed in this thesis result from attempts at bridging the realism gap from either side. Instead of repeating the conclusions of the three previous chapters which already detail their respective contributions, the present section offers to take a step back and to address the distinct results of this thesis from a more encompassing and abstract perspective.

In order to bridge the realism gap and train efficient recognition models on synthetic images only, assumptions have to be made. Although this thesis also presents a novel framework to incrementally capture new representations (*c.f.* Chapter 4), a postulate central to most use-cases is that synthetic representations of target objects are available to generate training images (*i.e.*, their 3D models) and that they contain the necessary visual information to perform the target recognition task. While most state-of-the-art solutions assume that 3D models with realistic textures are provided, this thesis demonstrate how the 3D shape alone can be leveraged for accurate recognition in industrial applications. If the choice of visual sensors is known a training time, the present work illustrates—on depth sensors—how an in-depth study of the sensing mechanisms can enable the development of realistic simulation tools (*c.f.* Chapter 3). Otherwise, if no specific sensing devices are targeted or if the variability of the target image domain is not clearly bounded, results presented in this thesis suggest that teaching a method to instead project the target samples into a well-defined synthetic domain is an effective strategy (*c.f.* Chapter 5). Either way, the distance between synthetic and real domains is reduced, facilitating the task of machine learning algorithms as theorized by seminal research in domain adaptation (*c.f.* Chapter 2).

Pictures, semantic annotations, 3D models, textual descriptions, *etc.* A more abstract take-away message is that data is nowadays everywhere and bountiful—as long as expectations w.r.t. specific modalities are lowered. Although the usage of real images to train computer vision algorithms is historically a low-hanging fruit and a promise of success, studies like this thesis are proof that solutions can leverage less relevant modalities to achieve satisfactory results (as humans often intuitively do).

6.2 Future Work

Among the propositions made in previous chapters, two main directions for future work can be highlighted.

6.2.1 Integration of Novel Differentiable Image Generators

Orthogonal to the contributions brought by the present thesis, a number of papers in the past years have been proposing novel, sometimes groundbreaking, solutions for image generation that tightly integrate into existing neural-based frameworks. The adoption of such novel methods could certainly improve the results of the pipeline presented in this dissertation.

Improving Image Quality with Recent GANs

Generative adversarial networks (GANs), especially, have been the focus of much interest in the past five years and have known significant developments.

Compensating for Simulation Limitations with GAN Post-Processing. As presented in Chapter 5, the literature already contains a variety of solutions based on unsupervised GANs, trained to map synthetic images closer to the target real domains, when some relevant images are provided (*e.g.*, *SimGAN* [240] and *PixelDA* [26]). In these papers, the solutions are applied to images rendered using simple tools (*e.g.*, without simulated noise for depth images).

Therefore, it could be interesting to evaluate if post-processing with proper GANs synthetic images which already have realistic features could lead to enhanced image quality. To that end, a possible direction for future work could be toward extending the simulation pipeline proposed in Chapter 3 with such methods, to compensate for the approximations in the simulation of complex phenomena and for the lack of structured data representing the target scenes (*i.e.*, to generate relevant background).

Compensating for Coarse Memorized Features with Novel GANs. The solution for incremental scene synthesis presented in Chapter 4 could also benefit from the latest development in generative models (*e.g.*, the impressive work by Wang et al. [280]) in order to hallucinate sharper features in the memory structure, as well as to reproject these accumulated features into detailed images.

Indeed, the current pipeline suffers from the quantization of the 3D space in memory and from the coarseness of the memorized features. While the overall pipeline could capitalize

on more powerful hardware to densify the memory structure, it could be interesting to observe how far the quality of generated images could be brought by leveraging some of the aforementioned novel neural networks for conditioned image generation.

Doing so, it could also be worthwhile to measure if the hallucination of more detailed/realistic content impacts or not the consistency of the localization and memorization processes of the overall pipeline. Indeed, in the present solution, the hallucination of uncertain content can be of lower quality due to the trade-off between representing uncertainties w.r.t. missing content and unsure localization (which leads to blurred results), and synthesizing detailed (but likely incorrect) images. The statistical nature of soft registration and hallucinations can add “uncertainty”, whereas the generative components of the pipeline partially compensate for them (*e.g.*, the choice of GAN models to improve the image sampling). For data generation use-cases, relaxing hallucination constraints and scaling up the generative training losses (\mathcal{L}_{anam} and \mathcal{L}_{hallu} in Subsections 4.3.2 and 4.3.3) can improve image detail, at the price of possible memory corruption. Therefore, future work towards improving the quality of sampled images should certainly take into account the trade-off with memory consistency.

Leveraging Differentiable Image Renderer for 2D-3D Mapping

Arguably one of the most interesting developments in neural-based image generation the past two years is the refinement and democratization (*c.f.* the release of *TensorFlow Graphics* [10] for instance) of differentiable 3D renderers [120, 145, 186]. Like traditional 3D renderers, these solutions take for input a 3D mesh, the camera parameters, and for some some additional scene properties (*e.g.*, light sources), and they return a corresponding color image. However, by leveraging novel rendering operations, these solutions are differentiable end-to-end and, therefore, can be trained like—or integrated with—neural networks (NNs), tightening the possible correspondences between 2D and 3D data. Applications are numerous, and research based on these renderers is gaining impetus.

Learning Image Recognition Directly from 3D Data. CAD-based training of visual recognition methods is typically performed in two steps due to limitations in computing power (*c.f.* GPU usage): first, a set of images are rendered from the target 3D models, then the recognition method is trained by iterating over this image set. This two-step process implies, therefore, a discretization of the image domain obtainable from the 3D data which can impact the end accuracy (*i.e.*, only a certain number of images, from a limited number of viewpoints and scene setups, can be rendered, *e.g.*, due to storage limitations). The advanced augmentation pipeline proposed in Chapter 5 partially covers this issue, being able to generate a virtually infinite number of color images (with different textures, lighting,

background) from a set of pre-rendered normal maps. However, this pre-rendering step still implies a discretization of the viewpoint space. Moreover, there is no exchange between the rendering and training procedures, *e.g.*, to generate images which cover the current flaws of the trained algorithms (*e.g.*, bias towards some lighting or background conditions).

These aforementioned generation and augmentation solutions to provide training images from 3D models to a CNN could ultimately be replaced by a differentiable renderer. Besides the simplification of the training process this would bring (*i.e.*, directly learning from 3D models), the adoption of these renderers could lead to potential accuracy boost. By back-propagating the gradient of the recognition loss all the way back into the differentiable renderer, one could build an adversarial training scheme, optimizing the parameters of the renderer (viewpoints, lighting conditions, *etc.*) to generate images which would be challenging for the current recognition network and, therefore, to possibly push that network toward more robust optima.

Mapping 2D Information back to 3D Models. Differentiable renderers have many more possible applications, such as regressing/reprojecting useful information from the 2D domain back to the 3D one. Some researchers such as Wang et al. [278] are already exploring such usages, *e.g.*, recovering partial 3D meshes from single color images. For industrial CAD models, it could be interesting to try regressing the texture information from images, in order to later render more realistic images from that model.

6.2.2 Beyond Image Domains

While this thesis is grounded into the field of image-based recognition, with its specific challenges and use-cases, some of the presented contributions could be applied beyond the image domain.

Applications to Different Modalities

The pipeline for incremental synthesis of 2D and 3D scenes detailed in Chapter 4 could possibly be applied to other data modalities, such as the generation of musical samples from partial audio inputs. Also, by opting for one-dimensional convolutions and memory structure in the proposed pipeline, one could attempt to piece together fragments of texts and to extrapolate missing content.

It could be as interesting to apply the *SynDA* domain adaptation scheme presented in Chapter 5 to different domains like audio or text, training the method to map real samples into synthetic or more simple domains to facilitate the end recognition tasks.

Intermediate Solution to Domain Adaptation

Finally, the originality of this thesis resides in its dual approach to the realism gap in computer vision, addressing it from both sides. The proposed solutions suggest to bridge this gap either by generating synthetic training data with more realistic features, or by projecting unseen target real images into the synthetic domain mastered by the recognition algorithms. I would be inclined to merge these two antipodal approaches, searching for a possible intermediate domain in which synthetic data could be projected during training, as well as the target images later in production (*e.g.*, a feature domain such as the one used to store the observations in the mnemonic pipeline).

Defining such a comprehensive domain (which could also include samples projected from other modalities) would be undoubtedly challenging but would also be a significant step toward the development of more versatile (*i.e.*, cross-domain) intelligent systems.

6.3 Epilogue

This thesis could be considered, rightfully, as substantiating the usage of synthetic data in computer vision – data clearly underemployed in state-of-the-art solutions. The fact that access to texture-less CAD models only is considered as a case of data scarcity is symptomatic of the current limitations of machine learning algorithms, which are still far from human capabilities and plasticity. Given some 3D models and an interface to visualize them, any average person could easily learn to identify the represented objects in real settings. Some researchers are exploring new methods in transfer learning and zero-shot learning [39, 127, 199], but solutions may lie beyond deep learning, the current main paradigm in machine learning.

Nevertheless, artificial visual intelligence won't be solved at once, and the only ambition of the present work is to add its targeted contributions to the much larger and collective undertaking. Although still imperfect, the virtual *Golems* (*c.f.* Section 1.1) that humankind has already shaped from the effusive developments in machine learning are presently metamorphosing our society. That artificial intelligence (AI) would one day free humankind from labor is probably wishful thinking (and an underestimation of our industrious nature), but its potential w.r.t. improving the work and life conditions of many is manifest. However, as told in the tale of the docile *Golem of Chelm*, people looking for solutions to automate vital tasks should be careful that their creations do not end up threatening the fabric of society.

While leveraging the data that we now continuously produce could be the antidote to a variety of long-standing problems, we should be careful not to give the synthetic digital

world too much power over ours. Philosophically speaking, we should certainly never fully bridge the realism gap...

Bibliography

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, et al. “Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”. In: *arXiv preprint arXiv:1603.04467* (2016) (cit. on pp. 30, 116).
- [2] Yaser S Abu-Mostafa. “The Vapnik-Chervonenkis dimension: Information versus complexity in learning”. In: *Neural Computation* 1.3 (1989), pp. 312–317 (cit. on pp. 34, 35, 37).
- [3] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. Vol. 4. AMLBook New York, NY, USA: 2012 (cit. on pp. 34, 35).
- [4] IN Aizenberg, NN Aizenberg, and J Vandewalle. “Multi-Valued and Universal Binary Neurons: Theory, Learning, and Applications”. In: *IEEE Transactions on Neural Networks* 12.3 (2001), p. 647 (cit. on p. 24).
- [5] Naveed Akhtar and Ajmal Mian. “Threat of adversarial attacks on deep learning in computer vision: A survey”. In: *IEEE Access* 6 (2018), pp. 14410–14430 (cit. on p. 32).
- [6] Michael A Alcorn, Qi Li, Zhitao Gong, et al. “Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects”. In: *arXiv preprint arXiv:1811.11553* (2018) (cit. on p. 32).
- [7] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, et al. “Point Cloud Library”. In: *IEEE Robotics & Automation Magazine* 1070.9932/12 (2012) (cit. on pp. 46, 54).
- [8] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kosecka, and Alexander C. Berg. “A Dataset for Developing and Benchmarking Active Vision”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017 (cit. on p. 95).
- [9] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. “Multiple Object Recognition with Visual Attention”. In: *arXiv preprint arXiv:1412.7755* (2014) (cit. on p. 111).
- [10] Paige Bailey, Sofien Bouaziz, Shan Carter, et al. “Differentiable graphics with TensorFlow 2.0”. In: *ACM SIGGRAPH 2019 Courses*. ACM. 2019, p. 10 (cit. on p. 130).
- [11] Maria-Florina Balcan and Avrim Blum. “A PAC-style model for learning from labeled and unlabeled data”. In: *International Conference on Computational Learning Theory*. Springer. 2005, pp. 111–126 (cit. on p. 37).
- [12] Peter L Bartlett and Wolfgang Maass. “Vapnik-Chervonenkis dimension of neural nets”. In: *The handbook of brain theory and neural networks* (2003), pp. 1188–1192 (cit. on p. 37).
- [13] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. “Speeded-up robust features (SURF)”. In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359 (cit. on p. 17).
- [14] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. “High-quality single-shot capture of facial geometry”. In: *ACM Transactions on Graphics (ToG)*. Vol. 29. 4. ACM. 2010, p. 40 (cit. on p. 46).
- [15] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. “Analysis of representations for domain adaptation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2007, pp. 137–144 (cit. on pp. 18, 39–41, 44).
- [16] Shai Ben-David, John Blitzer, Koby Crammer, et al. “A theory of learning from different domains”. In: *Machine learning* 79.1-2 (2010), pp. 151–175 (cit. on pp. xix, 18, 39–41, 44).

- [17] Shai Bendavid, Nicolo Cesabianchi, David Haussler, and Philip M Long. “Characterizations of Learnability for Classes of 0,..., n-Valued Functions”. In: *Journal of Computer and System Sciences* 50.1 (1995), pp. 74–86 (cit. on p. 36).
- [18] Yoshua Bengio et al. “Learning deep architectures for AI”. In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127 (cit. on pp. 18, 19).
- [19] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. “Are artificial neural networks black boxes?” In: *IEEE Transactions on neural networks* 8.5 (1997), pp. 1156–1164 (cit. on p. 31).
- [20] P.J. Besl and D McKay. “Method for Registration of 3-D Shapes”. In: *Robotics-DL Tentative*. International Society for Optics and Photonics, 1992, pp. 586–606 (cit. on p. 68).
- [21] Shehroze Bhatti, Alban Desmaison, Ondrej Miksik, et al. “Playing doom with slam-augmented deep reinforcement learning”. In: *arXiv preprint arXiv:1612.00380* (2016) (cit. on p. 77).
- [22] James F Blinn. “Models of Light Reflection for Computer Synthesized Pictures”. In: *ACM SIGGRAPH Computer Graphics*. Vol. 11. 2. ACM. 1977, pp. 192–198 (cit. on pp. 113, 114).
- [23] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. “Learnability and the Vapnik-Chervonenkis dimension”. In: *Journal of the ACM (JACM)* 36.4 (1989), pp. 929–965 (cit. on pp. 35–37).
- [24] Davide Boscaini, Jonathan Masci, Simone Melzi, et al. “Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks”. In: *Computer Graphics Forum*. Vol. 34. 5. Wiley Online Library. 2015, pp. 13–23 (cit. on p. 125).
- [25] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, et al. “Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping”. In: *arXiv preprint arXiv:1709.07857* (2017) (cit. on pp. 46, 113).
- [26] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. “Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks”. In: *arXiv preprint arXiv:1612.05424* (2016) (cit. on pp. 71, 104–106, 116, 119–121, 129).
- [27] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. “Domain Separation Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 343–351 (cit. on pp. 5, 6, 39, 46, 119).
- [28] G. Bradski. “OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000) (cit. on pp. 17, 64).
- [29] Simon Brodeur, Ethan Perez, Ankesh Anand, et al. “HoME: A Household Multimodal Environment”. In: *arXiv preprint arXiv:1711.11017* (2017) (cit. on pp. 45, 90).
- [30] Rodney A. Brooks. “Intelligence without Reason”. In: *Artificial Intelligence* 47.1-3 (1991), pp. 139–159 (cit. on p. 16).
- [31] Rodney A. Brooks. “Intelligence without representation”. In: *Artificial intelligence* 47.1-3 (1991), pp. 139–159 (cit. on p. 16).
- [32] Rodney A. Brooks, Russell Creiner, and Thomas O. Binford. “The ACRONYM Model-Based Vision System”. In: *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1. IJCAI’79*. Morgan Kaufmann Publishers Inc., 1979, pp. 105–113 (cit. on p. 53).
- [33] Arthur E Bryson. “A gradient method for optimizing multi-stage allocation processes”. In: *Proc. Harvard Univ. Symposium on digital computers and their applications*. Vol. 72. 1961 (cit. on p. 22).
- [34] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. “What do different evaluation metrics tell us about saliency models?” In: *arXiv preprint* (2016) (cit. on p. 96).
- [35] John Canny. “A computational approach to edge detection”. In: *Readings in computer vision*. Elsevier, 1987, pp. 184–203 (cit. on p. 12).
- [36] Yuanzhouhan Cao, Chunhua Shen, and Heng Tao Shen. “Exploiting depth from single monocular images for object detection and semantic segmentation”. In: *IEEE Trans. Image Processing* 26.2 (2017), pp. 836–846 (cit. on pp. 81, 107).

- [37] Fabio Maria Carlucci, Paolo Russo, and Barbara Caputo. “A Deep Representation for Depth Images from Synthetic Data”. In: *arXiv:1609.09713 [cs]* (Sept. 2016). arXiv: 1609.09713 [cs] (cit. on pp. 6, 45, 50).
- [38] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, et al. “Shapenet: An Information-Rich 3d Model Repository”. In: *arXiv preprint arXiv:1512.03012* (2015) (cit. on pp. 45, 51).
- [39] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. “Synthesized classifiers for zero-shot learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5327–5336 (cit. on p. 132).
- [40] Devendra Singh Chaplot, Emilio Parisotto, and Ruslan Salakhutdinov. “Active Neural Localization”. In: *ICLR*. 2018 (cit. on pp. 74, 78, 96).
- [41] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Return of the Devil in the Details: Delving Deep into Convolutional Nets”. In: *arXiv preprint arXiv:1405.3531* (2014) (cit. on p. 113).
- [42] Chao Chen, Zhihong Chen, Boyuan Jiang, and Xinyu Jin. “Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation”. In: *arXiv preprint arXiv:1808.09347* (2018) (cit. on pp. 18, 39).
- [43] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. “On Visual Similarity Based 3D Model Retrieval”. In: *Computer Graphics Forum*. Vol. 22. Wiley Online Library, 2003, pp. 223–232 (cit. on p. 49).
- [44] Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: *arXiv preprint arXiv:1601.06733* (2016) (cit. on pp. 88, 111).
- [45] AIA Chervonenkis and VN Vapnik. “Theory of uniform convergence of frequencies of events to their probabilities and problems of search for an optimal solution from empirical data(Average risk minimization based on empirical data, showing relationship of problem to uniform convergence of averages toward expectation value)”. In: *Automation and Remote Control* 32 (1971), pp. 207–217 (cit. on pp. xix, xx, 36, 37).
- [46] Benjamin Choo, Michael Landau, Michael DeVore, and Peter A Beling. “Statistical Analysis-Based Error Models for the Microsoft Kinect™ Depth Sensor”. In: *Sensors* 14.9 (2014), pp. 17430–17450 (cit. on p. 64).
- [47] Siddharth Choudhary, Vadim Indelman, Henrik I Christensen, and Frank Dellaert. “Information-based reduced landmark SLAM”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 4620–4627 (cit. on p. 79).
- [48] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. “3d-R2n2: A Unified Approach for Single and Multi-View 3d Object Reconstruction”. In: *ECCV*. Springer. 2016, pp. 628–644 (cit. on pp. 79, 96).
- [49] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. “Multi-Column Deep Neural Networks for Image Classification”. In: *CVPR*. IEEE. 2012, pp. 3642–3649 (cit. on p. 113).
- [50] Cisco. *VNI Global Fixed and Mobile Internet Traffic Forecasts*. URL: <https://cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/> (visited on May 11, 2019) (cit. on p. 3).
- [51] Maxwell B Clowes. “On seeing things”. In: *Artificial intelligence* 2.1 (1971), pp. 79–116 (cit. on p. 16).
- [52] Corinna Cortes and Vladimir Vapnik. “Support-Vector Networks”. In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 18).
- [53] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. “Optimal transport for domain adaptation”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.9 (2017), pp. 1853–1865 (cit. on pp. 5, 18, 38, 39).
- [54] Gabriela Csurka. *Domain adaptation in computer vision applications*. Springer, 2017 (cit. on p. 18).
- [55] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *international Conference on computer vision & Pattern Recognition (CVPR’05)*. Vol. 1. IEEE Computer Society. 2005, pp. 886–893 (cit. on p. 96).

- [56] Rina Dechter. *Learning while searching in constraint-satisfaction problems*. University of California, Computer Science Department, Cognitive Systems . . . , 1986 (cit. on p. 24).
- [57] J Deng, A Berg, S Satheesh, et al. “ILSVRC-2012, 2012”. In: *URL [http://www. image-net. org/challenges/LSVRC](http://www.image-net.org/challenges/LSVRC)* (2012) (cit. on p. 28).
- [58] Jia Deng, Wei Dong, Richard Socher, et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on pp. 28, 43, 51).
- [59] Hugh Durrant-Whyte and Tim Bailey. “Simultaneous Localization and Mapping: Part I”. In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110 (cit. on p. 77).
- [60] Bryson Arthur Earl and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. 1969 (cit. on p. 22).
- [61] David Eigen and Rob Fergus. “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”. In: *CVPR*. 2015, pp. 2650–2658 (cit. on pp. 81, 107, 109).
- [62] 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 (NeurIPS)*. 2014, pp. 2366–2374 (cit. on p. 107).
- [63] Riyad A El-laithy, Jidong Huang, and Michael Yeh. “Study on the Use of Microsoft Kinect for Robotics Applications”. In: *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*. IEEE, 2012, pp. 1280–1288 (cit. on p. 43).
- [64] Unity Game Engine. “Unity Game Engine-Official Site”. In: *Online*[[Cited: October 9, 2008.] [http://unity3d. com](http://unity3d.com) (), pp. 1534–4320 (cit. on p. 64).
- [65] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, et al. “Neural scene representation and rendering”. In: *Science* 360.6394 (2018), pp. 1204–1210 (cit. on p. 79).
- [66] Adrian N Evans and Xin U Liu. “A morphological gradient approach to color edge detection”. In: *IEEE Transactions on Image Processing* 15.6 (2006), pp. 1454–1463 (cit. on p. 12).
- [67] Maurice F Fallon, Hordur Johannsson, and John J Leonard. *Point Cloud Simulation & Applications*. 2012. URL: [http://www. pointclouds. org/assets/icra2012/localization. pdf](http://www.pointclouds.org/assets/icra2012/localization.pdf) (visited on Sept. 23, 2015) (cit. on p. 54).
- [68] Haoqiang Fan, Hao Su, and Leonidas J Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image.” In: *IEEE CVPR*. 2017 (cit. on pp. 79, 96).
- [69] Pedro F Felzenszwalb and Daniel P Huttenlocher. “Efficient belief propagation for early vision”. In: *International journal of computer vision* 70.1 (2006), pp. 41–54 (cit. on p. 64).
- [70] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. “3d Object Detection and Viewpoint Estimation with a Deformable 3d Cuboid Model”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012, pp. 611–619 (cit. on pp. 7, 43, 45, 49, 50).
- [71] David A Forsyth and Jean Ponce. “Computer Vision: A Modern Approach”. In: *Computer vision: a modern approach* (2003), pp. 88–101 (cit. on p. 77).
- [72] Marco Fraccaro, Danilo Jimenez Rezende, Yori Zwols, et al. “Generative Temporal Models with Spatial Memory for Partially Observed Environments”. en. In: *arXiv:1804.09401 [cs, stat]* (Apr. 2018). arXiv: 1804.09401 [cs, stat] (cit. on pp. 78, 79, 86, 87, 90–92, 96, 97).
- [73] Kunihiko Fukushima. “Neocognitron: A hierarchical neural network capable of visual pattern recognition”. In: *Neural networks* 1.2 (1988), pp. 119–130 (cit. on p. 25).
- [74] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological cybernetics* 36.4 (1980), pp. 193–202 (cit. on p. 25).
- [75] Peter Fürsattel, Simon Placht, Michael Balda, et al. “A Comparative Error Analysis of Current Time-of-Flight Sensors”. In: *IEEE Transactions on Computational Imaging* 2.1 (2016), pp. 27–41 (cit. on p. 65).

- [76] Tomer Galanti, Lior Wolf, and Tamir Hazan. “A theoretical framework for deep transfer learning”. In: *Information and Inference: A Journal of the IMA* 5.2 (2016), pp. 159–209 (cit. on pp. 30, 36, 37, 44).
- [77] Yaroslav Ganin and Victor Lempitsky. “Unsupervised Domain Adaptation by Backpropagation”. In: *International Conference on Machine Learning*. 2015, pp. 1180–1189 (cit. on pp. 5, 6, 39).
- [78] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, et al. “Domain-Adversarial Training of Neural Networks”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2096–2030 (cit. on pp. 39–41, 46, 104–106, 121).
- [79] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “Image Style Transfer Using Convolutional Neural Networks”. In: *CVPR*. 2016, pp. 2414–2423 (cit. on p. 104).
- [80] Andreas Geiger, Martin Roser, and Raquel Urtasun. “Efficient Large-Scale Stereo Matching”. In: *ACCV*. 2011, pp. 25–38 (cit. on pp. 45, 50).
- [81] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. “A PAC-Bayesian approach for domain adaptation with specialization to linear classifiers”. In: *International conference on machine learning*. 2013, pp. 738–746 (cit. on pp. 39, 41).
- [82] Abhijeet Ghosh, Shruthi Achutha, Wolfgang Heidrich, and Matthew O’Toole. “BRDF acquisition with basis illumination”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8 (cit. on p. 46).
- [83] Steven Gold, Anand Rangarajan, Chien-ping Lu, and Eric Mjolsness. “New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence”. In: *Pattern Recognition* 31 (1997), pp. 957–964 (cit. on pp. 49, 51).
- [84] Ary L Goldberger, Luis AN Amaral, Leon Glass, et al. “PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals”. In: *Circulation* 101.23 (2000), pp. 215–220 (cit. on p. 44).
- [85] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. “Shape and spatially-varying brdfs from photometric stereo”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.6 (2009), pp. 1060–1071 (cit. on p. 71).
- [86] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014, pp. 2672–2680 (cit. on p. 89).
- [87] Akshay Gore and Savita Gupta. “Full Reference Image Quality Metrics for JPEG Compressed Images”. In: *AEU-International Journal of Electronics and Communications* 69.2 (2015), pp. 604–608 (cit. on p. 93).
- [88] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. “BlenSor: Blender Sensor Simulation Toolbox”. In: *Advances in Visual Computing*. Springer, 2011, pp. 199–208 (cit. on pp. 6, 46, 54, 58, 63, 64).
- [89] Darya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. “BRDF representation and acquisition”. In: *Computer Graphics Forum*. Vol. 35. 2. Wiley Online Library. 2016, pp. 625–650 (cit. on p. 71).
- [90] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, et al. “A survey of methods for explaining black box models”. In: *ACM computing surveys (CSUR)* 51.5 (2018), p. 93 (cit. on p. 31).
- [91] David Gunning. “Explainable artificial intelligence (xai)”. In: *Defense Advanced Research Projects Agency (DARPA), nd Web* (2017) (cit. on p. 31).
- [92] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. “Cross Modal Distillation for Supervision Transfer”. In: *CVPR*. 2016, pp. 2827–2836 (cit. on pp. 107, 109, 111).
- [93] Barbara Hammer and Pascal Hitzler. *Perspectives of neural-symbolic integration*. Vol. 77. Springer, 2007 (cit. on p. 31).
- [94] Christopher G Harris, Mike Stephens, et al. “A combined corner and edge detector.” In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244 (cit. on p. 12).
- [95] Nick Harvey, Christopher Liaw, and Abbas Mehrabian. “Nearly-tight VC-dimension bounds for piecewise linear neural networks”. In: *Proceedings of the 2017 Conference on Learning*

- Theory*. Ed. by Satyen Kale and Ohad Shamir. Vol. 65. Proceedings of Machine Learning Research. Amsterdam, Netherlands: PMLR, 2017, pp. 1064–1068 (cit. on p. 37).
- [96] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CVPR*. 2016, pp. 770–778 (cit. on p. 119).
- [97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Identity Mappings in Deep Residual Networks”. In: *ECCV*. Springer. 2016, pp. 630–645 (cit. on pp. 88, 119, 120).
- [98] Joao F Henriques and Andrea Vedaldi. “MapNet: An Allocentric Spatial Memory for Mapping Environments”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8476–8484 (cit. on pp. 8, 73, 78, 80–82, 89, 93, 97).
- [99] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, et al. “Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes”. In: *ACCV*. 2012, pp. 548–562 (cit. on pp. 43, 67, 113, 119, 121).
- [100] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554 (cit. on pp. 24, 27).
- [101] Heiko Hirschmuller and Daniel Scharstein. “Evaluation of cost functions for stereo matching”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8 (cit. on p. 57).
- [102] Tomáš Hodan, Pavel Haluza, Štěpán Obdržálek, et al. “T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects”. In: *WACV*. 2017 (cit. on pp. 43, 44, 118, 120).
- [103] Wassily Hoeffding. “Asymptotically optimal tests for multinomial distributions”. In: *The Annals of Mathematical Statistics* (1965), pp. 369–401 (cit. on p. 34).
- [104] Derek Hoiem, Alexei A Efros, and Martial Hebert. “Geometric Context from a Single Image”. In: *ICCV*. IEEE. 2005 (cit. on p. 106).
- [105] T Huang. “Computer vision: Evolution and promise”. In: *1996 CERN School of Computing* (1996), p. 21 (cit. on p. 12).
- [106] David H Hubel and Torsten N Wiesel. “Receptive fields of single neurones in the cat’s striate cortex”. In: *The Journal of physiology* 148.3 (1959), pp. 574–591 (cit. on p. 16).
- [107] David A Huffman. “Impossible object as nonsense sentences”. In: *Machine intelligence 6* (1971), pp. 295–324 (cit. on p. 16).
- [108] Georg Hummel. “On synthetic datasets for development of computer vision algorithms in airborne reconnaissance applications.” PhD thesis. Bundeswehr University Munich, Neubiberg (Munich), Germany, 2017 (cit. on p. 45).
- [109] Computer Society IEEE. *CVPR 2018 Opening*. Salt Lake City, UT, USA, June 2019 (cit. on p. 3).
- [110] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *IEEE CVPR*. 2017 (cit. on p. 89).
- [111] Oliver James, Eugénie von Tunzelmann, Paul Franklin, and Kip S Thorne. “Gravitational lensing by spinning black holes in astrophysics, and in the movie *Interstellar*”. In: *Classical and Quantum Gravity* 32.6 (2015), p. 065001 (cit. on p. 44).
- [112] Dinghuang Ji, Junghyun Kwon, Max McFarland, and Silvio Savarese. “Deep View Morphing”. In: *IEEE CVPR*. 2017 (cit. on p. 79).
- [113] Yangqing Jia, Evan Shelhamer, Jeff Donahue, et al. “Caffe: Convolutional architecture for fast feature embedding”. In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 675–678 (cit. on pp. 30, 66).
- [114] Alistair EW Johnson, Tom J Pollard, Seth Berkowitz, et al. “MIMIC-CXR: A large publicly available database of labeled chest radiographs”. In: *arXiv preprint arXiv:1901.07042* (2019) (cit. on p. 44).
- [115] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. “Learning to predict where humans look”. In: *IEEE ICCV*. 2009 (cit. on p. 96).
- [116] Simon Julier. “The Stability of Covariance Inflation Methods for SLAM”. In: *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference On*. Vol. 3. IEEE. 2003, pp. 2749–2754 (cit. on pp. 77, 93).

- [117] Maya Kabkab, Emily Hand, and Rama Chellappa. “On the size of convolutional neural networks and generalization performance”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 3572–3577 (cit. on p. 18).
- [118] Takeo Kanade and Masatoshi Okutomi. “A stereo matching algorithm with an adaptive window: Theory and experiment”. In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE. 1991, pp. 1088–1095 (cit. on p. 57).
- [119] Kevin Karsch, Ce Liu, and Sing Bing Kang. “Depth Transfer: Depth Extraction from Video Using Non-Parametric Sampling”. In: *TPAMI* (2014) (cit. on p. 107).
- [120] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. “Neural 3d Mesh Renderer”. In: *IEEE CVPR*. 2018, pp. 3907–3916 (cit. on pp. 85, 130).
- [121] Maik Keller and Andreas Kolb. “Real-Time Simulation of Time-of-Flight Sensors”. en. In: *Simulation Modelling Practice and Theory* 17.5 (May 2009), pp. 967–978. ISSN: 1569190X (cit. on p. 54).
- [122] Henry J Kelley. “Gradient theory of optimal flight paths”. In: *Ars Journal* 30.10 (1960), pp. 947–954 (cit. on p. 22).
- [123] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics”. In: *arXiv preprint arXiv:1705.07115* 3 (2017) (cit. on pp. 81, 107, 109, 110, 122, 124).
- [124] Christian Kerl, Jürgen Sturm, and Daniel Cremers. “Dense visual SLAM for RGB-D cameras”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 2100–2106 (cit. on pp. 77, 79).
- [125] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. “Detecting change in data streams”. In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment. 2004, pp. 180–191 (cit. on pp. 39, 40).
- [126] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 90, 117).
- [127] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. “Unsupervised domain adaptation for zero-shot learning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2452–2460 (cit. on p. 132).
- [128] Kurt Konolige. “Small Vision Systems: Hardware and Implementation”. In: *Robotics Research*. Springer, 1998, pp. 203–212 (cit. on pp. 57, 60).
- [129] Wouter M Kouw. “An introduction to domain adaptation and transfer learning”. In: *arXiv preprint arXiv:1812.11806* (2018) (cit. on p. 39).
- [130] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012, pp. 1097–1105 (cit. on pp. 18, 28, 29, 107).
- [131] Ryohei Kuga, Asako Kanezaki, Masaki Samejima, Yusuke Sugano, and Yasuyuki Matsushita. “Multi-Task Learning Using Multi-Modal Encoderdecoder Networks with Shared Skip Connections”. In: *ICCV Workshop*. 2017 (cit. on pp. 107, 109–111, 122, 124).
- [132] Wataru Kumagai. “Learning bound for parameter transfer learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 2721–2729 (cit. on pp. 18, 39).
- [133] Kiriakos N. Kutulakos and Eron Steger. “A Theory of Refractive and Specular 3D Shape by Light-Path Triangulation”. In: *IEEE ICCV*. 2005, pp. 1448–1455 (cit. on p. 48).
- [134] Alina Kuznetsova, Hassan Rom, Neil Alldrin, et al. “The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale”. In: *arXiv preprint arXiv:1811.00982* (2018) (cit. on p. 43).
- [135] E Lachat, H Macher, MA Mittet, T Landes, and P Grussenmeyer. “First Experiences with Kinect v2 Sensor for Close Range 3D Modelling”. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 40.5 (2015), p. 93 (cit. on pp. 43, 65).
- [136] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “A Large-Scale Hierarchical Multi-View Rgb-d Object Dataset”. In: *IEEE ICRA*. IEEE, 2011, pp. 1817–1824 (cit. on p. 49).

- [137] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. “Deeper Depth Prediction with Fully Convolutional Residual Networks”. In: *3DV*. IEEE. 2016 (cit. on pp. 81, 105, 107, 121).
- [138] Michael J Landau. “Optimal 6D Object Pose Estimation with Commodity Depth Sensors”. <http://search.lib.virginia.edu/catalog/hq37vn57m><http://search.lib.virginia.edu/catalog/hq37vn57m>. Accessed: 2017-10-20. PhD thesis. University of Virginia, 2016 (cit. on pp. 43, 48, 49, 54, 58, 63, 64).
- [139] Michael J. Landau, Benjamin Y. Choo, and Peter A. Beling. “Simulating Kinect Infrared and Depth Images”. In: *IEEE Transactions on Cybernetics* 46.12 (Dec. 2016), pp. 3018–3031. ISSN: 2168-2267, 2168-2275 (cit. on pp. 6, 54, 58, 63, 64).
- [140] Yann LeCun, Yoshua Bengio, et al. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995 (cit. on p. 18).
- [141] Yann LeCun, Bernhard E Boser, John S Denker, et al. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1990, pp. 396–404 (cit. on p. 27).
- [142] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on pp. 25, 27, 66).
- [143] Alexander LeNail. “NN-SVG: Publication-Ready Neural Network Architecture Schematics”. In: *The Journal of Open Source Software* 4 (2019), p. 747 (cit. on p. 27).
- [144] Jerome Y Lettvin, Humberto R Maturana, Warren S McCulloch, and Walter H Pitts. “What the frog’s eye tells the frog’s brain”. In: *Proceedings of the IRE* 47.11 (1959), pp. 1940–1951 (cit. on p. 16).
- [145] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. “Differentiable monte carlo ray tracing through edge sampling”. In: *SIGGRAPH Asia 2018 Technical Papers*. ACM. 2018, p. 222 (cit. on p. 130).
- [146] Andy Liaw, Matthew Wiener, et al. “Classification and regression by randomForest”. In: *R news* 2.3 (2002), pp. 18–22 (cit. on p. 18).
- [147] Joerg Liebelt and Cordelia Schmid. “Multi-View Object Class Detection with a 3D Geometric Model”. In: *IEEE CVPR*. 2010, pp. 1688–1695 (cit. on pp. 6, 45).
- [148] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba. “Parsing ikea objects: Fine pose estimation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2992–2999 (cit. on pp. 44, 49, 50).
- [149] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. “Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018 (cit. on pp. 79, 96).
- [150] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755 (cit. on pp. 43, 117).
- [151] Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. In: *Master’s Thesis (in Finnish), Univ. Helsinki* (1970), pp. 6–7 (cit. on p. 22).
- [152] Ce Liu, Jenny Yuen, and Antonio Torralba. “Sift Flow: Dense Correspondence across Scenes and Its Applications”. In: *TPAMI* (2011) (cit. on p. 106).
- [153] Fayao Liu, Chunhua Shen, and Guosheng Lin. “Deep Convolutional Neural Fields for Depth Estimation from a Single Image”. In: *CVPR*. 2015, pp. 5162–5170 (cit. on p. 107).
- [154] Miaomiao Liu, Mathieu Salzmann, and Xuming He. “Discrete-Continuous Depth Estimation from a Single Image”. In: *CVPR*. 2014, pp. 716–723 (cit. on p. 107).
- [155] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015 (cit. on pp. 90, 94).

- [156] David G Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110 (cit. on p. 17).
- [157] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. “Effective Approaches to Attention-Based Neural Machine Translation”. In: *arXiv preprint arXiv:1508.04025* (2015) (cit. on p. 111).
- [158] Yuhao Ma, Kyle Boos, Joshua Ferguson, Donald Patterson, and Kevin Jonaitis. “Collaborative Geometry-Aware Augmented Reality with Depth Sensors”. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. ACM, 2014, pp. 251–254 (cit. on pp. 43, 49).
- [159] Gary Marcus. “Deep learning: A critical appraisal”. In: *arXiv preprint arXiv:1801.00631* (2018) (cit. on pp. 31, 32).
- [160] D. Marr and H. K. Nishihara. “Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes”. In: *Proceedings of the Royal Society of London B: Biological Sciences* 200.1140 (1978), pp. 269–294 (cit. on p. 53).
- [161] Kevan AC Martin. “A brief history of the “feature detector””. In: *Cerebral cortex* 4.1 (1994), pp. 1–7 (cit. on p. 16).
- [162] Georgios Mastorakis. “Human fall detection methodologies: from machine learning using acted data to fall modelling using myoskeletal simulation”. PhD thesis. Kingston University, 2018 (cit. on pp. 6, 45).
- [163] Georgios Mastorakis and Dimitrios Makris. “Fall detection system using Kinect’s infrared sensor”. In: *Journal of Real-Time Image Processing* 9.4 (2014), pp. 635–646 (cit. on p. 49).
- [164] Daniel Maturana and Sebastian Scherer. “VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition”. In: *IEEE IROS*. Sept. 2015 (cit. on pp. 7, 49–51).
- [165] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. “A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955”. In: *AI magazine* 27.4 (2006), p. 12 (cit. on p. 15).
- [166] Sean McClure. *Sean McClure’s Answer to What Was Leo Breiman Trying to Convey in His Research Paper Statistical Modeling - The Two Cultures? - Quora*. URL: <https://www.quora.com/What-was-Leo-Breiman-trying-to-convey-in-his-research-paper-Statistical-Modeling-The-Two-Cultures-Was-he-trying-to-say-that-the-field-is-odd-Did-he-mean-that-algorithmic-modelling-machine-learning-is-superior-to-traditional-data-modelling/answer/Sean-McClure-3> (visited on Apr. 8, 2019) (cit. on p. 31).
- [167] Median Group. *How Rapidly Are GPUs Improving in Price Performance?* URL: <http://mediangroup.org/gpu.html> (visited on May 11, 2019) (cit. on p. 3).
- [168] Fabio Menna, Fabio Remondino, Roberto Battisti, and Erica Nocerino. “Geometric Investigation of a Gaming Active Device”. In: *SPIE Optical Metrology*. International Society for Optics and Photonics, 2011, 80850G–80850G (cit. on p. 64).
- [169] Dimitris N Metaxas. *Physics-based deformable models: applications to computer vision, graphics and medical imaging*. Vol. 389. Springer Science & Business Media, 2012 (cit. on p. 125).
- [170] Marvin Minsky. “Steps toward Artificial Intelligence”. In: *Proceedings of the IRE* 49.1 (1961), pp. 8–30 (cit. on p. 15).
- [171] Marvin Minsky and Seymour A Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT press, 2017 (cit. on p. 22).
- [172] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. “Spectral Normalization for Generative Adversarial Networks”. In: *arXiv preprint arXiv:1802.05957* (2018) (cit. on p. 88).
- [173] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. “Recurrent Models of Visual Attention”. In: *NIPS*. 2014, pp. 2204–2212 (cit. on p. 111).

- [174] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem”. In: *Aaai/iaai* 593598 (2002) (cit. on p. 77).
- [175] Gordon E Moore et al. *Cramming more components onto integrated circuits*. 1965 (cit. on pp. 2, 29).
- [176] Hans Moravec. “Locomotion, Vision and Intelligence”. In: (1984). Ed. by Michael Brady and Richard Paul, pp. 215–224 (cit. on p. 15).
- [177] Marius Muja and David G. Lowe. “Scalable Nearest Neighbor Algorithms for High Dimensional Data”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36 (2014) (cit. on p. 68).
- [178] Aaftab Munshi. “The opencl specification”. In: *2009 IEEE Hot Chips 21 Symposium (HCS)*. IEEE, 2009, pp. 1–314 (cit. on p. 29).
- [179] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163 (cit. on pp. 77, 79, 93).
- [180] Loris Nanni, Stefano Ghidoni, and Sheryl Brahmam. “Handcrafted vs. non-handcrafted features for computer vision classification”. In: *Pattern Recognition* 71 (2017), pp. 158–172 (cit. on p. 30).
- [181] Chahab Nastar and Nicholas Ayache. “Fast segmentation, tracking, and analysis of deformable objects”. In: *1993 (4th) International Conference on Computer Vision*. IEEE, 1993, pp. 275–279 (cit. on p. 125).
- [182] Balas K Natarajan. “On learning sets and functions”. In: *Machine Learning* 4.1 (1989), pp. 67–97 (cit. on p. 36).
- [183] Natalia Neverova, Pauline Luc, Camille Couprie, Jakob Verbeek, and Yann LeCun. “Predicting Deeper into the Future of Semantic Segmentation”. In: *ICCV*. 2017, pp. 1–10 (cit. on p. 111).
- [184] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, et al. “KinectFusion: Real-Time Dense Surface Mapping and Tracking”. In: *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR ’11. 2011, pp. 127–136 (cit. on pp. 43, 49).
- [185] Chuong V Nguyen, Shahram Izadi, and David Lovell. “Modeling Kinect Sensor Noise for Improved 3d Reconstruction and Tracking”. In: *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference On*. IEEE, 2012, pp. 524–530 (cit. on p. 64).
- [186] Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. “RenderNet: A deep convolutional network for differentiable rendering from 3d shapes”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 7891–7901 (cit. on p. 130).
- [187] Jannik Boll Nielsen, Henrik Wann Jensen, and Ravi Ramamoorthi. “On optimal, minimal BRDF sampling for reflectance acquisition”. In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), p. 186 (cit. on p. 71).
- [188] David Nistér. “Preemptive RANSAC for Live Structure and Motion Estimation”. In: *Machine Vision and Applications* 16.5 (2005), pp. 321–329 (cit. on p. 77).
- [189] Kyoung-Su Oh and Keechul Jung. “GPU implementation of neural networks”. In: *Pattern Recognition* 37.6 (2004), pp. 1311–1314 (cit. on pp. 27, 29).
- [190] Nuria M Oliver, Barbara Rosario, and Alex P Pentland. “A bayesian computer vision system for modeling human interactions”. In: *IEEE transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 831–843 (cit. on p. 18).
- [191] Michael Ounsworth. “Anticipatory Movement Planning for Quadrotor Visual Servoing”. PhD thesis. McGill University Libraries, 2015 (cit. on p. 115).
- [192] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. “Domain adaptation via transfer component analysis”. In: *IEEE Transactions on Neural Networks* 22.2 (2011), pp. 199–210 (cit. on pp. 5, 6, 18, 39).
- [193] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359 (cit. on p. 18).

- [194] S. Papert. “The Summer Vision Project”. In: AI memo (1966) (cit. on p. 14).
- [195] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. “A Decomposable Attention Model for Natural Language Inference”. In: *arXiv preprint arXiv:1606.01933* (2016) (cit. on pp. 88, 111).
- [196] Emilio Parisotto and Ruslan Salakhutdinov. “Neural Map: Structured Memory for Deep Reinforcement Learning”. In: *ICLR*. 2018 (cit. on p. 78).
- [197] Emilio Parisotto, Devendra Singh Chaplot, Jian Zhang, and Ruslan Salakhutdinov. “Global pose estimation with an attention-based recurrent network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 237–246 (cit. on p. 78).
- [198] Adam Paszke, Sam Gross, Soumith Chintala, et al. “Automatic Differentiation in PyTorch”. In: *OpenReview - NIPS 2017 Workshop Autodiff Submission* (2017) (cit. on pp. 30, 82, 88).
- [199] Kuan-Chuan Peng, Ziyang Wu, and Jan Ernst. “Zero-shot deep domain adaptation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 764–781 (cit. on p. 132).
- [200] Patrick Pérez, Michel Gangnet, and Andrew Blake. “Poisson image editing”. In: *ACM Transactions on graphics (TOG)* 22.3 (2003), pp. 313–318 (cit. on p. 12).
- [201] Ken Perlin. “Improving Noise”. In: *ACM Transactions on Graphics (TOG)*. Vol. 21. 3. ACM. 2002, pp. 681–682 (cit. on pp. 113, 114, 117).
- [202] Robert J Peters, Asha Iyer, Laurent Itti, and Christof Koch. “Components of bottom-up gaze allocation in natural images”. In: *Vision research* 45.18 (2005), pp. 2397–2416 (cit. on p. 96).
- [203] Valerij Peters and Otmar Loffeld. “A Bistatic Simulation Approach for a High-Resolution 3D PMD (Photonic Mixer Device)-Camera”. In: *International Journal of Intelligent Systems Technologies and Applications* 5.3-4 (2008), pp. 414–424 (cit. on p. 54).
- [204] Bui Tuong Phong. “Illumination for Computer Generated Pictures”. In: *Communications of the ACM* 18.6 (1975), pp. 311–317 (cit. on p. 113).
- [205] Benjamin Planche and Eliot Andres. *Hands-On Computer Vision with TensorFlow 2*. Packt Publishing, May 2019. ISBN: 9781788830645 (cit. on pp. 12, 14, 32).
- [206] Benjamin Planche, Xuejian Rong, Ziyang Wu, et al. “Incremental Scene Synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019 (cit. on p. 73).
- [207] Benjamin Planche, Ziyang Wu, Kai Ma, et al. “DepthSynth: Real-Time Realistic Synthetic Data Generation from CAD Models for 2.5 D Recognition”. In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 1–10 (cit. on pp. 47, 57).
- [208] Benjamin Planche, Sergey Zakharov, Ziyang Wu, et al. “Seeing Beyond Appearance - Mapping Real Images into Geometrical Domains for Unsupervised CAD-based Recognition”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019 (cit. on p. 99).
- [209] DC Plaut, SJ Nowlan, and GE Hinton. “Experiments on learning by Backpropagation Technical Report CMU-CS-86-126”. In: *Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA* (1986) (cit. on p. 22).
- [210] David Pollard. *Convergence of stochastic processes*. Springer Science & Business Media, 2012 (cit. on p. 36).
- [211] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, et al. “Neural Episodic Control”. en. In: *arXiv:1703.01988 [cs, stat]* (Mar. 2017). arXiv: 1703.01988 [cs, stat] (cit. on pp. 78, 80, 96).
- [212] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. “Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation”. In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1.2 (2017), p. 4 (cit. on pp. 82, 85).
- [213] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015) (cit. on p. 89).

- [214] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. “Do CIFAR-10 Classifiers Generalize to CIFAR-10?” In: *arXiv preprint arXiv:1806.00451* (2018) (cit. on p. 32).
- [215] Ievgen Redko, Amaury Habrard, and Marc Sebban. “Theoretical analysis of domain adaptation with optimal transport”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017, pp. 737–753 (cit. on pp. 18, 39).
- [216] Konstantinos Rematas, Tobias Ritschel, Matt Fritz, and Tinne Tuytelaars. “Image-Based Synthesis and Re-Synthesis of Viewpoints Guided by 3d Models”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference On*. IEEE, 2014, pp. 3898–3905 (cit. on pp. 6, 45, 51).
- [217] C. Rennie, R. Shome, K. E. Bekris, and A. Ferreira De Souza. “A Dataset for Improved RGBD-Based Object Detection and Pose Estimation for Warehouse Pick-and-Place”. In: *IEEE Robotics and Automation Letters* 1 (Feb. 2016), pp. 1179–1185 (cit. on p. 67).
- [218] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241 (cit. on pp. 111, 122).
- [219] Dan Rosenbaum, Frederic Besse, Fabio Viola, Danilo J Rezende, and SM Eslami. “Learning models for visual 3D localization with implicit mapping”. In: *arXiv preprint arXiv:1807.03149* (2018) (cit. on pp. 78, 97).
- [220] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386 (cit. on pp. 18, 21).
- [221] William B Rouse and Nancy M Morris. “On looking into the black box: Prospects and limits in the search for mental models.” In: *Psychological bulletin* 100.3 (1986), p. 349 (cit. on p. 31).
- [222] Anirban Roy and Sinisa Todorovic. “Monocular Depth Estimation Using Neural Regression Forest”. In: *CVPR*. 2016, pp. 5506–5514 (cit. on pp. 81, 107).
- [223] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. “On Rendering Synthetic Images for Training an Object Detector”. In: *Computer Vision and Image Understanding* (2015) (cit. on pp. 7, 43, 49, 50, 53).
- [224] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. “Learning representations by back-propagating errors”. In: *Cognitive modeling* 5.3 (1988), p. 1 (cit. on p. 22).
- [225] Fereshteh Sadeghi and Sergey Levine. “(CAD)2RL: Real Single-Image Flight without a Single Real Image”. In: *arXiv preprint arXiv:1611.04201* (2016) (cit. on pp. 8, 102, 106).
- [226] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, et al. “Improved Techniques for Training Gans”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242 (cit. on p. 89).
- [227] Mostafa Samir. *Machine Learning Theory - Part 2: Generalization Bounds*. URL: <https://mostafa-samir.github.io/ml-theory-pt2/> (visited on Apr. 11, 2019) (cit. on pp. 35, 36).
- [228] A.L. Samuel. “Some studies in machine learning using the game of checkers”. In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229 (cit. on pp. 14, 15).
- [229] Jason Sanders and Edward Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010 (cit. on p. 29).
- [230] M Saquib Sarfraz and Olaf Hellwich. “Head Pose Estimation in Face Recognition Across Pose Scenarios.” In: *VISAPP (1)* 8 (2008), pp. 235–242 (cit. on p. 17).
- [231] Norbert Sauer. “On the density of families of sets”. In: *Journal of Combinatorial Theory, Series A* 13.1 (1972), pp. 145–147 (cit. on p. 36).
- [232] John G Saw, Mark CK Yang, and Tse Chin Mo. “Chebyshev inequality with estimated mean and variance”. In: *The American Statistician* 38.2 (1984), pp. 130–132 (cit. on p. 34).
- [233] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. “Learning 3-D Object Orientation from Images”. In: *IEEE ICRA*. 2009, pp. 4266–4272 (cit. on pp. 6, 45).
- [234] Ashutosh Saxena, Min Sun, and Andrew Y Ng. “Learning 3-d Scene Structure from a Single Still Image”. In: *ICCV*. IEEE. 2007, pp. 1–8 (cit. on pp. 49–51).

- [235] Mark Segal and Kurt Akeley. “The OpenGL® Graphics System: A Specification (Version 4.5 (Core Profile)-May 28, 2015)”. In: *The Khronos Group Inc* (2015) (cit. on p. 117).
- [236] Claude E. Shannon. “Programming a Computer for Playing Chess”. In: *Philosophical Magazine* 41.314 (1950), pp. 256–275 (cit. on pp. 14, 15).
- [237] Saharon Shelah. “A combinatorial problem; stability and order for models and theories in infinitary languages”. In: *Pacific Journal of Mathematics* 41.1 (1972), pp. 247–261 (cit. on p. 36).
- [238] Bonggun Shin, Falgun H Chokshi, Timothy Lee, and Jinho D Choi. “Classification of radiology reports using neural attention models”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 4363–4370 (cit. on p. 31).
- [239] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, et al. “Real-Time Human Pose Recognition in Parts from a Single Depth Image”. In: *IEEE CVPR*. June 2011 (cit. on pp. 43, 49).
- [240] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, et al. “Learning from Simulated and Unsupervised Images through Adversarial Training”. In: *arXiv preprint arXiv:1612.07828* (2016) (cit. on pp. 71, 104, 129).
- [241] Siemens. *Easy Spares Idea*. URL: <https://new.siemens.com/global/en/products/mobility/rail-solutions/services/spare-part-services/easy-spare-idea.html> (visited on May 11, 2019) (cit. on p. 4).
- [242] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. “Indoor Segmentation and Support Inference from RGBD Images”. In: *ECCV*. Springer, 2012, pp. 746–760 (cit. on pp. 7, 49).
- [243] Patrice Y Simard, Dave Steinkraus, and John C Platt. “Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis”. In: *Null*. IEEE. 2003, p. 958 (cit. on pp. 25, 113).
- [244] Ashutosh Singh, Jin Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. “Bigbird: A Large-Scale 3d Database of Object Instances”. In: *IEEE ICRA*. IEEE, 2014, pp. 509–516 (cit. on pp. 45, 49, 50).
- [245] Jan Smisek, Michal Jancosek, and Tomas Pajdla. “3D with Kinect”. In: *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 3–25 (cit. on p. 43).
- [246] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. “Convolutional-Recursive Deep Learning for 3d Object Classification”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012, pp. 665–673 (cit. on pp. 49, 50).
- [247] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite”. In: *IEEE CVPR*. 2015, pp. 567–576 (cit. on pp. 7, 49, 50, 90).
- [248] Shuran Song, Fisher Yu, Andy Zeng, et al. “Semantic Scene Completion from a Single Depth Image”. In: *arXiv preprint arXiv:1611.08974* (2016) (cit. on p. 45).
- [249] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958 (cit. on p. 37).
- [250] Michael Stark, Michael Goesele, and Bernt Schiele. “Back to the Future: Learning Shape Models from 3D CAD Data.” In: *BMVC*. Vol. 2. 2010, p. 5 (cit. on pp. 6, 45, 51).
- [251] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference On*. IEEE. 2012, pp. 573–580 (cit. on pp. 77, 79, 93).
- [252] Peter Sturm and Bill Triggs. “A Factorization Based Algorithm for Multi-Image Projective Structure and Motion”. In: *European Conference on Computer Vision*. Springer. 1996, pp. 709–720 (cit. on p. 77).

- [253] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. “Multi-View Convolutional Neural Networks for 3d Shape Recognition”. In: *IEEE ICCV*. 2015 (cit. on pp. 50, 51).
- [254] Hao Su, Charles R Qi, Yangyan Li, and Leonidas Guibas. “Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views”. In: *arXiv preprint arXiv:1505.05641* (2015) (cit. on pp. 5, 6, 51, 53).
- [255] Masashi Sugiyama, Makoto Yamada, and Marthinus Christoffel du Plessis. “Learning under nonstationarity: covariate shift and class-balance change”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 5.6 (2013), pp. 465–477 (cit. on pp. 38, 39).
- [256] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968 (cit. on pp. 37, 44).
- [257] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. “Multi-View to Novel View: Synthesizing Novel Views with Self-Learned Confidence”. In: *ECCV*. 2018, pp. 155–171 (cit. on p. 96).
- [258] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. “Implicit 3d orientation learning for 6d object detection from rgb images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 699–715 (cit. on p. 106).
- [259] Richard Szeliski. “Bayesian modeling of uncertainty in low-level vision”. In: *International Journal of Computer Vision* 5.3 (1990), pp. 271–301 (cit. on p. 18).
- [260] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010 (cit. on p. 16).
- [261] Yaniv Taigman, Adam Polyak, and Lior Wolf. “Unsupervised Cross-Domain Image Generation”. In: *arXiv preprint arXiv:1611.02200* (2016) (cit. on p. 104).
- [262] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. “Multi-View 3d Models from Single Images with a Convolutional Network”. In: *ECCV*. Springer. 2016, pp. 322–337 (cit. on pp. 79, 96).
- [263] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. “CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction”. In: *IEEE CVPR*. 2017 (cit. on p. 78).
- [264] Geoffrey R Taylor, Andrew J Chosak, and Paul C Brewer. “Ovvv: Using virtual worlds to design and evaluate surveillance systems”. In: *2007 IEEE conference on computer vision and pattern recognition*. IEEE. 2007, pp. 1–8 (cit. on p. 45).
- [265] Josh Tobin, Rachel Fong, Alex Ray, et al. “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World”. In: *IROS*. IEEE. 2017, pp. 23–30 (cit. on pp. 8, 39, 46, 102, 106, 113).
- [266] Michael Trembl, José Arjona-Medina, Thomas Unterthiner, et al. “Speeding up Semantic Segmentation for Autonomous Driving”. In: *MLITS, NIPS Workshop*. 2016 (cit. on p. 111).
- [267] Alan M. Turing. “Computing machinery and intelligence”. In: *Mind* 59.236 (1950), pp. 433–460 (cit. on pp. 14, 15).
- [268] Alan M Turing. “Intelligent machinery, a heretical theory”. In: *The Turing Test: Verbal Behavior as the Hallmark of Intelligence* 105 (1948) (cit. on p. 15).
- [269] Matthew Turk and Alex Pentland. “Eigenfaces for recognition”. In: *Journal of cognitive neuroscience* 3.1 (1991), pp. 71–86 (cit. on pp. 16, 17).
- [270] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. “Simultaneous Deep Transfer across Domains and Tasks”. In: *ICCV*. 2015, pp. 4068–4076 (cit. on p. 111).
- [271] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. “Adversarial discriminative domain adaptation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7167–7176 (cit. on p. 104).
- [272] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. “Deep Domain Confusion: Maximizing for Domain Invariance”. In: *arXiv preprint arXiv:1412.3474* (2014) (cit. on pp. 5, 6, 39).

- [273] Vladimir Vapnik. *The nature of statistical learning theory*. 2013 (cit. on p. 18).
- [274] Vladimir N Vapnik and A Ya Chervonenkis. “On the uniform convergence of relative frequencies of events to their probabilities”. In: *Measures of complexity*. Springer, 2015, pp. 11–30 (cit. on pp. xix, xx, 36, 37).
- [275] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention Is All You Need”. In: *NIPS*. 2017, pp. 5998–6008 (cit. on pp. 88, 111).
- [276] Armando Vieira and Nuno Barradas. “A training algorithm for classification of high-dimensional data”. In: *Neurocomputing* 50 (2003), pp. 461–472 (cit. on p. 27).
- [277] Mei Wang and Weihong Deng. “Deep visual domain adaptation: A survey”. In: *Neurocomputing* 312 (2018), pp. 135–153 (cit. on pp. 18, 39).
- [278] Nanyang Wang, Yinda Zhang, Zhuwen Li, et al. “Pixel2mesh: Generating 3d mesh models from single rgb images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 52–67 (cit. on p. 131).
- [279] Peng Wang, Xiaohui Shen, Zhe Lin, et al. “Towards Unified Depth and Semantic Prediction from a Single Image”. In: *CVPR*. 2015, pp. 2800–2809 (cit. on pp. 107, 109).
- [280] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, et al. “High-resolution image synthesis and semantic manipulation with conditional gans”. In: *IEEE CVPR*. 2018 (cit. on pp. 98, 129).
- [281] Yan Wang, Jie Feng, Zhixiang Wu, Jun Wang, and Shih-Fu Chang. “From Low-Cost Depth Sensors to Cad: Cross-Domain 3d Shape Retrieval via Regression Tree Fields”. In: *European Conference on Computer Vision*. Springer, 2014, pp. 489–504 (cit. on p. 50).
- [282] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. “Image Quality Assessment: From Error Visibility to Structural Similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612 (cit. on p. 93).
- [283] Zhou Wang and Qiang Li. “Information Content Weighting for Perceptual Image Quality Assessment”. In: *IEEE Transactions on Image Processing* 20.5 (2011), pp. 1185–1198 (cit. on p. 93).
- [284] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. “Multiscale Structural Similarity for Image Quality Assessment”. In: *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. IEEE. 2003, pp. 1398–1402 (cit. on p. 93).
- [285] Georg Wiese, Dirk Weissenborn, and Mariana Neves. “Neural domain adaptation for biomedical question answering”. In: *arXiv preprint arXiv:1706.03610* (2017) (cit. on p. 31).
- [286] Paul Wohlhart and Vincent Lepetit. “Learning Descriptors for Object Recognition and 3d Pose Estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3109–3118 (cit. on pp. 6, 45, 50, 51, 66, 110, 112, 116, 118–121).
- [287] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal Component Analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52 (cit. on p. 17).
- [288] Steven Worley. “A Cellular Texture Basis Function”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM. 1996, pp. 291–294 (cit. on pp. 114, 117).
- [289] Changchang Wu. “Towards Linear-Time Incremental Structure from Motion”. In: *3D Vision-3DV 2013, 2013 International Conference On*. IEEE. 2013, pp. 127–134 (cit. on p. 77).
- [290] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. “Building generalizable agents with a realistic and rich 3D environment”. In: *arXiv preprint arXiv:1801.02209* (2018) (cit. on p. 45).
- [291] Zhirong Wu, Shuran Song, Aditya Khosla, et al. “3d Shapenets: A Deep Representation for Volumetric Shapes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1912–1920 (cit. on pp. 50, 51).
- [292] Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. “ViZDoom Competitions: Playing Doom from Pixels”. In: *IEEE Transactions on Games* (2018) (cit. on p. 95).
- [293] Yu Xiang, Wonhui Kim, Wei Chen, et al. “Objectnet3d: A large scale database for 3d object recognition”. In: *European conference on computer vision*. Springer. 2016, pp. 160–176 (cit. on p. 45).

- [294] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. “Sun database: Large-scale scene recognition from abbey to zoo”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 3485–3492 (cit. on p. 54).
- [295] Chang Xu, Tongliang Liu, Dacheng Tao, and Chao Xu. “Local rademacher complexity for multi-label learning”. In: *IEEE Transactions on Image Processing* 25.3 (2016), pp. 1495–1507 (cit. on p. 36).
- [296] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. “PAD-Net: Multi-Tasks Guided Prediction-and-Distillation Network for Simultaneous Depth Estimation and Scene Parsing”. In: *arXiv preprint arXiv:1805.04409* (2018) (cit. on pp. 81, 107, 109–111, 122, 124).
- [297] Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. “Weakly-Supervised Disentangling with Recurrent Transformations for 3d View Synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 1099–1107 (cit. on p. 79).
- [298] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. “Evaluating bag-of-visual-words representations in scene classification”. In: *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM. 2007, pp. 197–206 (cit. on p. 18).
- [299] Ting Yu, Tony Jan, Simeon Simoff, and John Debenham. “Incorporating prior domain knowledge into inductive machine learning”. In: *Unpublished doctoral dissertation Computer Sciences* (2007) (cit. on pp. 30, 44).
- [300] Sergey Zakharov, Wadim Kehl, Benjamin Planche, Andreas Hutter, and Slobodan Ilic. “3D Object Instance Recognition & Pose Estimation Using Triplet Loss with Dynamic Margin”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 552–559 (cit. on pp. 6, 52, 53, 66, 110, 112, 118–121).
- [301] Sergey Zakharov, Benjamin Planche, Ziyang Wu, et al. “Keep it Unreal: Bridging the Realism Gap for 2.5D Recognition with Geometry Priors Only”. In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 1–11 (cit. on pp. 111–113, 116, 119).
- [302] Jure Zbontar, Florian Knoll, Anuroop Sriram, et al. “fastmri: An open dataset and benchmarks for accelerated mri”. In: *arXiv preprint arXiv:1811.08839* (2018) (cit. on p. 44).
- [303] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. “Understanding deep learning requires rethinking generalization”. In: *CoRR* abs/1611.03530 (2016) (cit. on p. 37).
- [304] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. “Self-Attention Generative Adversarial Networks”. In: *arXiv preprint arXiv:1805.08318* (2018) (cit. on pp. 88, 89, 111, 122, 124).
- [305] Jingwei Zhang, Lei Tai, Joschka Boedecker, Wolfram Burgard, and Ming Liu. “Neural SLAM: Learning to explore with external memory”. In: *arXiv preprint arXiv:1706.09520* (2017) (cit. on pp. 77, 78, 96, 97).
- [306] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. “A Comprehensive Evaluation of Full Reference Image Quality Assessment Algorithms”. In: *Image Processing (ICIP), 2012 19th IEEE International Conference On*. IEEE. 2012, pp. 1477–1480 (cit. on p. 93).
- [307] Ruo Zhang, P.-S. Tsai, J.E. Cryer, and M. Shah. “Shape-from-Shading: A Survey”. In: *IEEE TPAMI* 21.8 (1999), pp. 690–706 (cit. on p. 49).
- [308] Fang Zhao, Jiashi Feng, Jian Zhao, Wenhan Yang, and Shuicheng Yan. “Robust Lstm-Autoencoders for Face de-Occlusion in the Wild”. In: *IEEE Transactions on Image Processing* 27.2 (2018), pp. 778–790 (cit. on p. 78).
- [309] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. “View Synthesis by Appearance Flow”. In: *ECCV*. Springer. 2016, pp. 286–301 (cit. on pp. 79, 96).
- [310] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *ICCV*. 2017 (cit. on pp. 111, 123).

Colophon

This thesis was typeset with L^AT_EX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at <http://cleanthesis.der-ric.de/>.

