Institute For Artificial Intelligence
Faculty of Mathematics and Computer Science

Master Thesis

Scene-graph-oriented Visual Scene Understanding for Robot Manipulation Tasks

A Deep-Learning- and Virtual-Reality-based Approach

Franklin Kenghagho Kenfack

Matriculation No. 3036422

15. April 2019

Examiner: Prof. Michael Beetz Ph.D.
Examiner: Prof. Thomas Schneider Ph.D.
Supervisor: Prof. Michael Beetz Ph.D.
Advisor: Msc. Ferenc Bálint Benczédi
Advisor: Msc. Feroz Ahmed Siddiky

Franklin Kenghagho Kenfack

Scene-graph-oriented Visual Scene Understanding for Robot Manipulation Tasks

A Deep-Learning- and Virtual-Reality-based Approach

Master Thesis, Institute For Artificial Intelligence
Faculty of Mathematics and Computer Science

University Of Bremen, April 2019

Declaration of copyright

Hereby I declare that my Master's Thesis was written without external support and that I did not use any other sources and auxiliary means than those quoted and cited. All statements which are literally or analogously taken from other publications have been identified as quotations or citations.

Bremen, 15. April 2019

Franklin Kenghagho Kenfack

Acknowledgements

I would first like to thank my thesis advisors Msc. Ferenc Bálint Benczédi and Msc. Feroz Ahmed Siddiky of the Institute for Artificial Intelligence at the University of Bremen. The door to Prof. Michael Beetz's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to thank the experts who were involved in the validation survey for this research project: Prof.Dr. Michael Beetz, Prof.Dr. Thomas Schneider, Msc. Ferenc Bálint Benczédi and Msc. Feroz Ahmed Siddiky. Without their passionate participation and input, the validation survey could not have been successfully conducted.

I would also like to acknowledge Msc. Feroz Ahmed Siddiky and Msc. Ferenc Bálint Benczédi of the Institute for Artificial Intelligence at University of Bremen as the second readers of this thesis, and I am gratefully indebted to their for their very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my family, formost my father Mathias Kemda, my mother Marie Songong Kemda and my brothers namely Claude Kemda, Joel Kemda, Olivia Kemda, Bleriot Kemda, Ange Kemda, Manuela Kemda and Horlane Kemda without forgetting other important relatives precisely Hermann Tsamo and Desire Nanfack as well as my friends Leo Kenmoe and Fabrice Awana for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you very much.

Bremen, 15. April 2019

Franklin Kenghagho Kenfack

Abstract

Interest in robots that can competently perform human-scale daily manipulation activities in human-centered environments, such as cooking in an ordinary kitchen, has been significantly growing. However, to complete such tasks, robots should not only separatedly exhibit standard visual perception capabilities such as captioning, detection, localization or recognition, but they should also demonstrate some cognitive vision, in which all these capabilities including considerable reasoning and attention are integrated together so that a deep understanding of the cluttered, noisy and dynamic scene can be reached [214, 215, 202].

Unfortunately, state-of-the-art Computer Vision systems still show some weaknesses in addressing this problem. While traditional Computer Vision systems suffer from intrinsic modeling limitations in practice, revolutionary Deep-Learning-based systems more than being not appropriate for Robotics applications, are victims to the Big Rich Data crisis [204, 164, 143, 107, 33, 82, 12, 113, 207].

This thesis shows that the underlying perception problem of visual scene understanding, mentioned above, can be solved by relying on the computational power of Deep-Learning-based computational models and leveraging accurately designed photo-realistic virtual worlds for their training, escaping then the Big Rich Data crisis. Given a single RGB or RGBD image of the robot scene, the proposed deep model, known as RobotVQA, outputs a scene graph, in which each node corresponds to a well- and fix-formatted dense description of an object in the scene and each directed edge represents a spatial relationship between the objects encoded by the nodes. Each object's description encompasses its fundamental visual properties namely the object's Category, Color, Material, Shape, Openability, Segmentation, Bounding Box and $6D$-Pose, whereas spatial relationships namely the On-, In-, Left- and Front-relations provide very good insights into how the objects in the scene do and can interact with each other. Additionaly, experiments show that a single system inference over a scene image takes averagely $0.13s$ (seconds), $0.226s$ including the input loading phase and $5.5GB$ of memory to complete on a GPGPU@(NVIDIA GeForce GTX 1080 Ti).

Summarisingly, RobotVQA presents the following key features:

- Deep Learning: able of generalization and end-to-end learning [82, 12, 113].
- Virtual Reality: allows Big Rich Data [14].
- Structuredness: scene-graphs have a fix format and well-defined values [95].
- Completeness: scene graphs are sufficiently informative to address the robot visual question answering problem on manipulation tasks [194, 215, 202].
- Explicit multi-task Learning: unification allows good reasoning through an inter-subtask collaboration and prevents overfiting [163].
- Multimodality: can work either with RGB or RGBD images.
- Reasonable Complexity: ≈ 5 frames/s + $5.5GB$ memory (Image Loading + Inference).

Contents

Chapter 1

Introduction

This chapter aims at defining the problem addressed by this thesis. It starts with a short history that sets the context and motivations of the thesis and then offers a brief technical overview of the topic of concern, where key concepts are defined and related works reviewed. Finally, the research questions, the goals as well as the roadmap of the thesis are clarified.

1.1 Motivation

History

Robots are physical agents that are developed to automatically solve problems while perceiving their operating environments and acting on them [24]. Interest in Robotics Reasearch is at least twofold [91, 81]. From the scientific viewpoint, it aims at giving insights into human nature, whereas from an engineering viewpoint, it allows to build machines that can phisically as well as cognitively substitute humans typically in overloading tasks, risky operating domains and careful manipulations. Robots are of different categories that can be distinguished by the type of problems they solve, the type of environments they interact with and the way they interact with their environments [59, 24]. Manipulators or Manipulation robots, as shown by figure 1.1, are one of the most common robot types developed in the world for they play a major role in solving the above mentioned problems. In constrast to other robot types, manipulation robots are particularly designed and equipped with artificial arms to tackle manipulation tasks. Demands as well as scientific researches in manipulation robots have been growing and shifting from the industrial sectors to home environments. However, while industrial robots have been being designed for simple pure physical tasks (e.g. welding, transport, etc) in well-controlled and isolated environments, Household Robotics, commonly referred to as Service Robotics, has shown itself to be extremly challenging. That is, in contrast to industrial robots, household robots are called on the one hand to operate within human evironments, which are especially

cluttered, noisy and dynamic. On the other hand, household robots are required to deal with typical human-scale manipulation tasks such as supermarket's, restaurant's, kitchen's services, house cleaning and inspection, which are computationally very complex [99, 49, 108, 93].

Figure 1.1 On the left side, a set of fixed KUKA industrial robots are welding cars. On the right side, an active household humanoid robot PR2 at cooking in the Institute For Artificial Intelligence At University of Bremen (IAI-Bremen)'s Kitchen.

source: [39, 87]

Given that living beings, whether they arose through natural evolution or intelligent design [9, 208], have been being continuously shaped by the latter to successfully behave (i.e. problem solving) within their operating environments, researchers in the field of Artificial Intelligence (AI) have been focusing on understanding human higher-level cognition, perception and motorics as well as how to transfer them to Household Robotics. This research paradigm, known as embodied situated symbolic-connectionnist cognition, which does not merely lead to effective household robots but helps back to better understand human behavior, has led to a very complex robot architecture commonly known as AI humanoid [128, 30, 89, 52, 42, 17, 149, 187, 186]. Since AI humanoids' higher-level cognition, which mainly focuses on abstract symbolic reasoning, often called Good Old-Fashioned Artificial Intelligence (GOFAI), has known significant success in the past, it has been receiving only few attention from the community recent years. Contrarily, humanoids' perception and motorics have been under the focus of intensive researches [29]. However, given that AI humanoids' motorics is usually driven by their perception, the latter has shown to be one of the most, not to say the most, challenging problems in Household Robotics [205]. By humanoid perception, it is understood the ability of humanoid robots to sense their environments and interpret their sensations. Like humans, humanoids are usually expected to actively use their five senses to perceive their environments. Among those senses, vision, for at least two reasons, has shown itself to be the most necessary. The first reason is that sensing is end-functional and Robotics applications do not usually stronly and equally require the robot to demonstrate all the five senses to be effectively, as well as efficiently, operational. Navigation and maniputation tasks for instance do not strongly require taste and smell contrarily to

vision and touch. Secondly, it has been proven that the greatest part of human perception takes place through vision [168, 28, 30, 99, 212]. Unfortunately, actual state-of-the-art Robot Vision (RV) systems, to the best of investigations, still fail to properly support autonomous household robots on complex human-scale manipulation tasks. Fundamentally, as intensively argued by the community [99, 30, 186, 187, 96], this is certainly due to the fact that actual Computer Vision (CV) systems do not fit the paradigm of embodied situated symbolic-connectionnist cognition. This paradigm implies that computational mechanisms, such as artificial vision, should be encapsulated within the proper hardware (i.e. embodiement) that allows them to efficiently and effectively interact with their intended real operating environments (i.e. situatedness), while emulating the distributed parallel processing of the nervous systems (i.e. connectionism) to accomplish lower-level cognition (i.e. perception, motorics) and the intuitive symbolic processing of the invisible mind (i.e. symbolism) to complete higher-level cognition (e.g. reasoning, language, planning, memory, etc). It then logically follows that artificial perception systems should be embodied, situated and importantly connectionist. Failure to meet these requirements in the past resulted in a situation where traditional CV systems, based on shallow symbolic architectures, suffer from intrinsic modeling limitations in practice, and revolutionary Deep Learning (DL)-based vision systems, based on deep connectionist architectures, more than being actually not appropriate for Robotics applications, are victims to the Big Rich Data crisis [204, 164, 143, 107, 33, 82, 12, 113, 207].

Research Project EASE.

Everyday Activity in Science and Engineering (EASE) [22, 136, 18, 172, 47, 23] is the research project of the interdisciplinary research center EASE at the University of Bremen that investigates everyday activities from a scientific and engineering viewpoint, and whose a subcenter for Integrated Intelligent Systems (IIS) is hosted at the IAI-Bremen. Its core purpose consists in enabling humanoids to competently perform complex human-scale daily manipulation activities such as cooking in an ordinary kitchen or carrying out chemical operations within a laboratory.

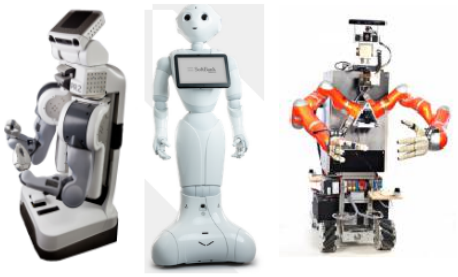


Figure 1.2 EASE Humanoids. On the leftmost side is PR2, at the center Pepper and Boxy on the rightmost side.

source: [86, 157]

Experiments at EASE are manifold however with a centered focus on cooking activities, certainly for their relevance to humankind and the computational challenges they raise. For this reason, an ordinary kitchen has been specially designed and built to frame research activities, where three humanoid robots are usually active namely the PR2 humanoid of Willow Garage, Pepper humanoid of Sofbank Robotics and Boxy humanoid of IAI-Bremen, as shown by figure 1.2. Open source softwares have been being developed notably KnowRob [195, 20], PRAC [134, 136, 135] to equip these robots with knowledge representation and reasoning capabilities, CRAM [21] for planning capabilities, GISKARD [58] for motorics capabilities, RoboSherlock [19] for vison capabilities, ROBCOG [70] for acquiring knowledge from virtual environments and finally OpenEASE [196, 48] for sharing knowledge. Unfortunately, as it will be seen later in this thesis, though RoboSherLock adopts a philosophy that aims at addressing the RV challenges mentioned above, its underlying working modules based on shallow architectures computationally seem to fail. This limitation of RoboSherlock and its improvement, that can be obviously approached by framing general research questions, are the very motivation of this thesis. Given that Deep Learning can yield powerful computational models but requiring Big Rich Data for training, usually really unreachable, this thesis, as illustrated in figure 1.3, aims at proposing an alternative to RoboSherlock called RobotVQA while preserving its philosophy, based on Deep Learning while leveraging virtual environments enabled by ROBCOG for training RobotVQA. Before accurately framing the research questions addressed and the goals aimed by this thesis, let briefly explore and define the related key concepts from the field of Robot Vision.

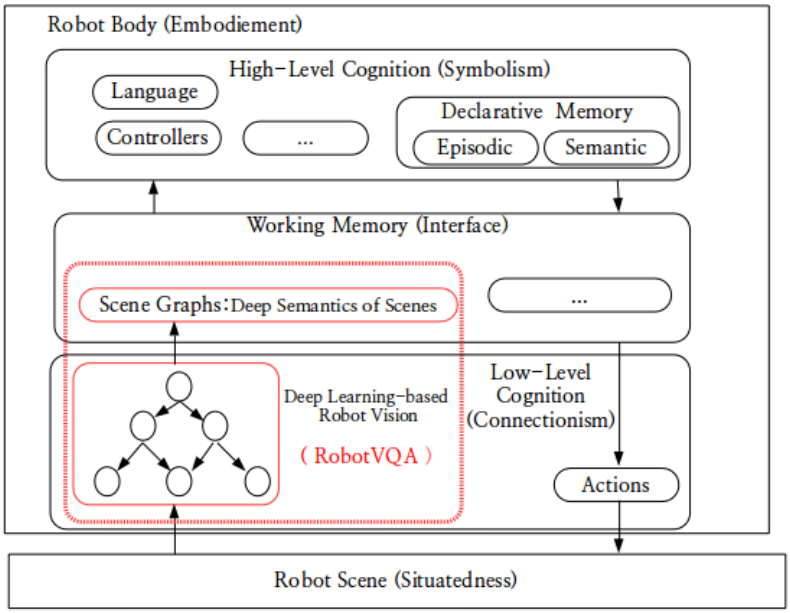


Figure 1.3 A SOAR-like general cognitive robot architecture with RobotVQA in the red-bordered frame.

1.2 Robot Vision

Robot vision has shown to be a major bottleneck to success in Household Robotics. This section aims at giving a broad overview of the field of Robot vision and particularly for the sake of completing manipulation tasks, which is the concern in this thesis.

1.2.1 Basics

To basically understand Robot Vision, it is necessary to examine it from a minimal set of perspectives namely its concept, the supporting hardware and the core approaches.

1.2.1.1 Concept

Goals

Computer Vision is the field of Artificial Intelligence that aims at developing computational artificial vision systems. On the one hand, from an engineering point of view, Computer Vision seeks to automate tasks that are importantly achieved by the visual system of living beings but importantly humans (humanoids). On the other hand, that is scientifically spoken, it seeks to answer questions about the computability of the biological visual system either from humans or animals. Summarisingly, Computer Vision tries to come out with computational mechanisms that can visually understand scenes. From the scene, artificial vision mechanisms extract information that a biological visual system is supposed to [191].

Standard Setup

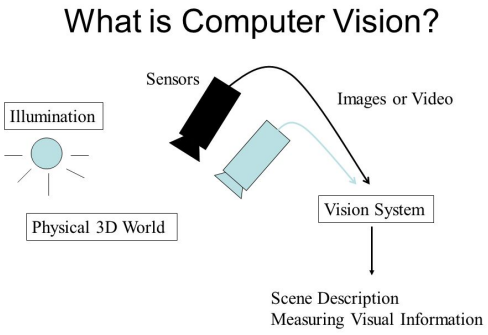


Figure 1.4 Standard Computer Vision system

source: [184]

As shown by figure 1.4, to operate, artificial vision mechanisms are first embodied within a computer system, which is in turn situated within an environment commonly referred to as scene. The computer system is then equipped with acquisition devices (i.e. artificial eyes) to sense the scene. Sensing the scene results in a raw representation of it, usually only partial depending on the viewpoint and viewfield of the acquisition sensor. When sensing occurs instantaneously, the resulting raw representation of the scene is called a scene image. A scene image usually contains information related to scene points such as color and depth; And the amount of scene points contained within an image is called the image resolution, which in turn determines the image size. In contrast to instantaneous sensing, continuous sensing yields a video; That is a sequence of images acquired over a chunk of time at a particular frequence. That chunk of time is called the video length and the acquisition frequence is known as the video frame rate, largely determined by the nature of the acquisition sensor and if the sensor works under multiple calibrations, then the adequate frame rate can be chosen to fit the application that the whole system is built for. The video size is then determined by its length, frame rate and frame size. Finally, the scene's raw representation is then passed to internal computational mechanisms (i.e. artificial visual cortex) for extracting the scene's semantics. This semantics is called an interpretation of the scene [191, 5, 38, 121, 31].

Robot Vision Vs Computer Vision

Robot Vision, as shown by figure 1.5, is certainly a subfield of Computer Vision, however with its own very specificities that substantially differ it from other areas of Computer Vision.

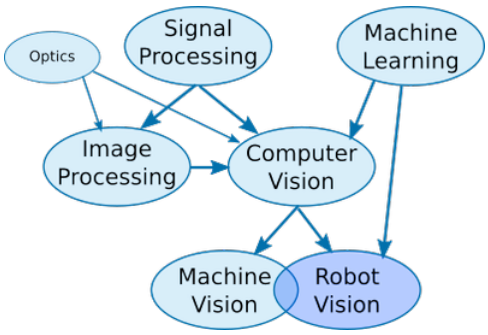


Figure 1.5 Artificial Vision's Family Tree

source: [50, 5]

That is, Computer Vision is not only dedicated to robots, which are extremly alive and dynamic, but rather also to soft computer agents such as vision-based surveillance systems, personal recognizers/identifiers, trackers, search engines, synthesizers/summarizers, etc which though important for humans essentially fail to meet Robotics applications' requirements. From a roboticist's viewpoint, the first major criticism of such soft vision systems is their very weak embodiedness and situatedness: these systems are significantly assisted on their inputs and outputs by humans.

Since they are usally soft, motionally static and environmentally isolated systems, their inputs are most often prepared by humans, for instance by ensuring adequate scene illumination, no blurring, perfect calibration, perfect positioning of the scene acquisition device, no corruption and sometimes scene arrangement. In the same way, their outputs are usually so vague, ambiguous and unspecific for robot action but quite acceptable and significant to humans who can easily exploit them because of their inherent nature to deal with ambiguity. One could argue that computational mechanisms could be built to disambiguate such vague and unspecific outputs, useless for efficient and effective robot action; However, this task (i.e. disambiguation of natural language) has shown itself to be very challenging and the resulting computational complexity would lead to the second, but quite related to the first, criticism of soft vision agents which is about realtimeness. Realtimeness [106] is the ability of a system to react as fast as possible to the stimuli from its environment in order to satisfy the constraints of the latter's dynamics. Most soft vision agents are not realtime and are not required to be. They quite often work offline and in batch mode on large-size images and videos, and then deliver an output report only after an unknown but long amount of time. And when working online, their passivity, isolation and assistedness pay for their non realtimeness: they do not undergo any substantial risk in case of tardive response. Following these remarks, there have been three major development mainstreams in the field of Computer Vision [38, 209].

Reactive Vision

The first mainstream is known as pure robot vision often called machine vision. In this paradigm, vision is embodied within robotics agents (usually very primitive) and situated within a real environment to directly guide in realtime the robot action. There is a strong coupling between action and vision, meaning that perception feedbacks are quasi-directly transcripted into motor commands in order to perform the desired action. Unfortunately, this architecture that tends to be reactive (i.e. Stimulus-Response (SR)) has shown to be extremly limited in addressing very complex tasks such as complex human-scale manipulation tasks which require considerable scene semantics from perception systems rather than just grasp trajectories, navigation directions, detections, etc. As it is for instance the case for the multinational company Amazon.com Inc, well-known for packet delivery as well as for its numerous logistics robots, SR-based robot vision has been most often successfully adopted but only in well-controlled and landmarked areas; That is in places where stimuli can be directly and easily interpreted. Such robots are just a stage above standard industrial robots as far as vision is concerned [121, 31, 209].

Deliberative Vision

The second mainstream concerns human-centered Computer Vision, commonly called Computer Vision. In this development paradigm, researches aim at addressing problems which are directly challenging for the human vision system. Among such vision systems are for example facial recognition (i.e. identification) and surveillance systems for security purposes or translation systems of visual content to text (vice-versa) for search, synthesis (e.g. creativity) and even medical assis-

tance (e.g. blindness) purposes. These heavy vision systems solve very complex visual problems by performing deep reasoning on visual contents to then come out with significantly meaningful information. Unfortunately, as mentioned earlier, though extremly meaningful to humans, these information are inexpoitable by robots and therefore meaningless to them. Attempts to equip robots with such vision systems have been doomed to failure. It has been proven for instance that the very prominent Viola-Jones facial detection algorithm is inherently inappropriate when embodied in robotic agents (i.e. bad illumination, viewfield and viewpoint) [209, 68, 67, 73].

Hybrid Vision

Finally, the last and recent development paradigm, which aims at embodied situated symbolic-connectionist cognition, tries to merge the advantages of pure robot vision (i.e. realtimeness, situatedness, embodiedness, structured scene represenation) and human-centered computer vision (i.e. cognition, complete scene representation) to yield vision systems that can allow robots to efficiently and effectively perform complex tasks in complex environments. As it will be seen later, a considerable forward step toward this goal has been Machine Learning (ML) and more recently Deep Machine Learning together with new more powerful DL-supporting computing hardwares. This thesis aims at such a vision system that can allow household humanoids to competently perform complex human-scale everyday activities [209].

1.2.1.2 Sensors

Before extracting information from the scene, the vision system needs to capture the scene; That is having a raw representation of the scene, as the human eye does. For this sake, artificial sensors have been being developed and each with its own specifity. Note that even animals' and humans' vision systems possess their own specificities that differ them from one another. Specificity concerns for instance the type of data that constitute the scene's raw representation and if it is for example the intensity of electromagnetic waves, as it is the case for many living beings, fine-grained specificity may concern the electromagnetic spectrum (e.g. visible). Following, as illustrated by figure 1.6, are the major sensors used in the field of Robot Vision.

RGB-Camera [165]

The RGB-camera is a passive sensor. It projects the white light emitted by the $3D$- or $2D$-scene onto a delimited rectangular $2D$-image plane called the camera sensor plane. The image plane is in fact spatially and uniformly discretisized into tiny rectangular receptive fields called pixel cells. Each pixel cell records the smallest unit of processable information whithin an image: this smallest unit of information is known as pixel. On recording the white light hitting each pixel cell, the white light is discretisized into three uniformly distributed spectra namely the Red, Green and Blue. Finally, the $3D$-vector encoding the respective intensity values of the Red, Green and Blue light is saved as the pixel value. It can also work actively when operating in

darker scene; In this case, the camera firstly lights the scene and then records the reflection. This camera is the classical image acquisition sensor for artificial vision systems, since it approximates the human eye at most and furthermore, it is considerably cheap. The lost of scene depth due to $2D$-projection constitutes its main disadvantage. Depth lost, though effortless recovered by the human vision system, is a major issue for artificial vision systems when operating with RGB-cameras. A technique known as stereo-vision and trying to combine two RGB-cameras in order to recover the scene depth while emulating biological binocular vision has only been difficultly successful. All the EASE's humanoids are equipped with a RGB-camera.

Figure 1.6 Example of sensors used in Robot Vision. (a) Ultrasonic sonar sensor provided by Arduino. (b) Radar sensor provided by Banner Engineering. (c) Three types of lidars provided by Velodyne. (d) RGB Camera provided by Raspberry Pi. (e) Stereo-vision sensor provided by Voltrium Systems. (f) RGBD Kinect camera provided by Microsoft provided by Microsoft.

source: [120]

LiDaR [173]

LiDaR stands for Light Detection and Ranging. It is essentially an active sensor. In contrast to a RGB-camera, a LiDaR sensor tracks the scene depth but ignores the properties of the electromagnetic waves emitted by the scene. The scene depth is computed as the distance travelled by the light emitted by the LiDar sensor towards the scene. This is achieved with knowledge of the speed and travelling time of the emitted light (i.e. time-of-flight calculation). The travelling time goes from when the light is emitted to when it is detected back after reflection by scene obstacles. This mechanism, though uncommon to the human vision system, is an effective hardware solution to the problem of scene depth recovery, faced by artificial vision systems. LiDaR sensors are commonly used for obstacle detection in the field of Robot Navigation.

RGBD-Camera [146]

The RGBD-camera unifies the RGB-Camera and the LiDaR sensor into a single sensor called

RGB- and Depth-camera. For its decreasing price in the market and ability to mimic the human eye (i.e. scene light+depth), RGBD-cameras are becoming the standard image acquisition sensors for artificial vision systems, after RGB-cameras. All the EASE's humanoids are equipped with a RGBD-camera.

Hyperspectral Camera [218]

The hyperspectral camera, as indicated by its name, works like a RGB-camera. However, it records the light emitted by the scene along the nearly full electromagnetic spectrum. It provides an extrem precision of the properties of the scene's light and can consequently allow direct retrieval of the scene's semantics. Unfortunately, this sensor is extremly expensive, data- and processing-burdening (i.e. curse of dimensionality). Certainly for these reasons, most, not to say all, robots including the EASE's humanoids are not equipped with such a camera. However, in areas, such as Geomatics, where there are no real alternatives, hyperspectral cameras receive great attention.

Thermographic Camera [15]

A thermographic camera operates quite similarly to most electromagnetism-based cameras cited above. However, it has been built to detect only the spectrum of infrared waves, ascribing to it consequently the specificity to operate in darker and especially hoter scene. For this quality, it has been mostly employed in military and surveillance applications, since they are usually required to operate in the night. And since most household humanoids are built to operate within almost illuminated indoor environments no matter whether cold or hot, this camera has not been really attractive to Household Robotics.

RaDaR [153]

RaDaR stands for Radio Detection and Ranging. It works like a LiDaR sensor, but instead of using white light as it is in LiDaR, radio waves are used. It is less precise than LiDar, but much more cheaper. It is commonly used in long range detection such as outdoor or aerial vision in drones, whereas LiDar is suitable for short range detection such as indoor vision in household robots. These specificities are due to the inability of white light and ability of radio waves to avoid distortions (e.g. clouds, walls) and undergo dispersions while travelling. For these reasons, RaDar has been obselete in Household Robotics and therefore absent in EASE's humanoids.

SoNaR [1]

SoNaR stands for Sound Navigation and Ranging. It works like LiDaR and RaDaR, however with the use of acoustic waves instead of electromagnetic waves. SoNaR is foremost an alternative where electromagnetism-based sensors are not very effective or quasi-useless such as in undersea vision. That is why it does not come into play when talking about Indoor Robotics and is not part of the EASE's humanoids' sensors.

Summarisingly, this brief analysis of the major sensors used in Robot Vision shows that the

RGBD-cameras, as already mounted on EASE's humanoids, constitute the most suitable candidates for image acquisation in a novel robot vision system provided by this thesis.

1.2.1.3 Core Approaches

As depicted ealier, the last stage of a typical robot vision system consists in interpretating the acquired raw representation of the scene. That is, extracting meaningful information from the images and videos of the scene.

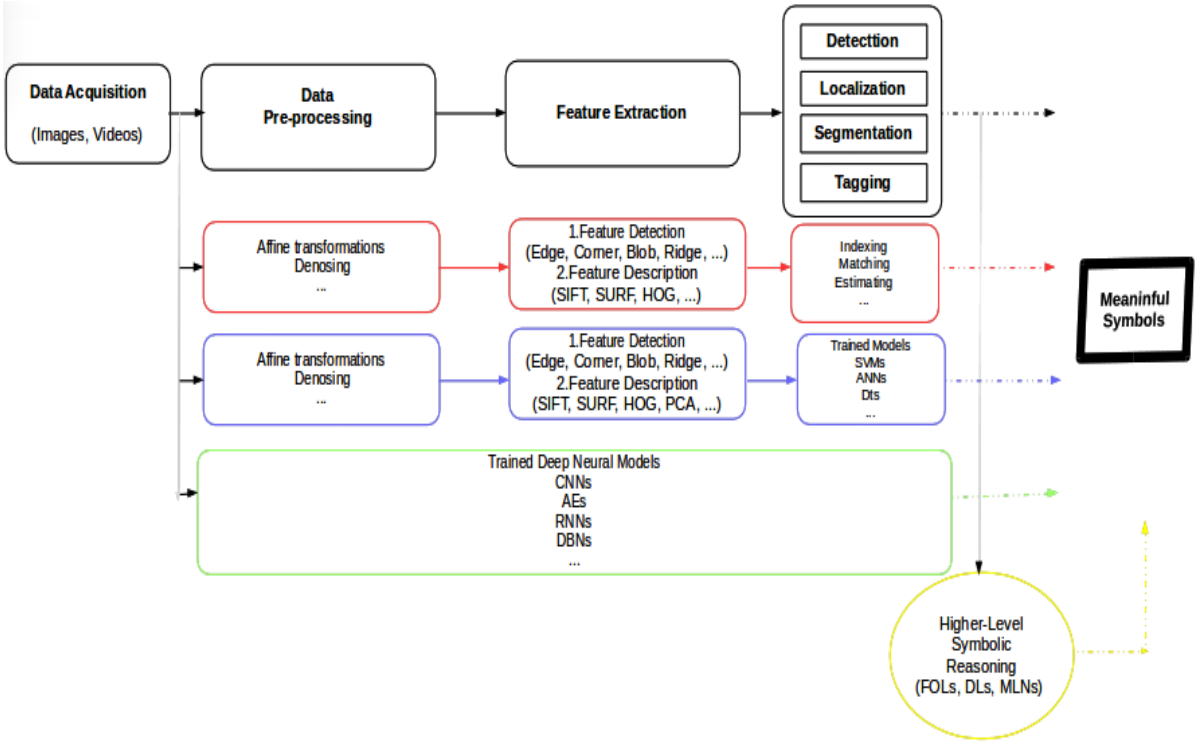


Figure 1.7 Standard Visual Processing Pipelines. The abstract processing pipeline is black-bordered. The old-school processing pipeline is red-bordered, the traditional Machine Learning processing pipeline is blue-bordered. The Deep Learning processing pipeline is green-bordered. The hybrid processing pipeline is yellow-bordered. Scale Invariant Feature Transform (SIFT), Speeded Up Robust Feature (SURF), Histogram of Oriented Gradient (HOG), Principal Component Analysis (PCA) are some hand-crafted features. Support Vector Machine(s) (SVM(s)), Decision Tree(s) (DT(s)), Artificial Neural Network(s) (ANN(s)) are some traditional Machine Learning models. Convolutional Neural Network(s) (CNN(s)), Reccurent Neural Network(s) (RNN(s)), Auto-Encoder(s) (AE(s)), Deep Belief Network(s) (DBN(s)) are some classical Deep Learning models. Description Logic(s) (DL(s)), First-Order Logic(s) (FOL(s)), Markov Logic Network(s) (MLN(s)) are some symbolic reasoning formalisms.

An important question would be the one to know what it means to be a *"meaningful"* information when processing such an image or video. That is, why is the raw representation of the scene itself not *"meaningful"*? This very interesting question would be revisited in the next section, but for now it is important to note that perception enables rational decision making. Since decision making is task-dependent, this logically suggests that *"meaningful"* in this context is task-dependent, implying that an extracted information is *"meaningful"* if and only if it can effectively be exploited by the robot to efficiently improve the accomplishment of the task which the vision system was needed for [122]. For instance, though the color of scene objects may be relevant for some tasks, they are nearly useless for robot navigation and that is why RGB-cameras are seldom used in such a context. Given that there are infinitely many possible visual tasks (i.e. problems), instead of doomly trying to provide a solution (i.e. algorithm) to each of them, researches in Computer Vision, as it has always been the tradition in any systematic investigation, have been identifying, as illustrated by Figure 1.7, building blocks of problems on the one side and building blocks of solutions on the other side.

Building Block Problems

Recognition Problem [94, 63]. Recognition has shown as the most fundamental block for building problems in Computer Vision (especially Robot Vision) and solutions to this building block is usually of an extrem relevance to other building blocks. Basically, recognition problems are questions about images but can also extrapolate from images to videos. The first question is *Detection*, which is broadly speaking about whether or not a particular feature appears on the input image (e.g. detection of defections, ...). *Image Tagging* is the second relevant question, that takes place after detection (implicitly or explitcitly) and is about what objects of interest as well as their properties are on the image. Objects are taken here in a broader sense and may therefore refer to concepts such as tools, humans, relations, regions, whereas properties may refer to categories, colors, shapes, etc. The following question is known as *Localization* and expresses the issue of pose estimation in Robot Vision. That is, where exactly the objects of interest on the image are so that the robot can attain and grasp them. The output of such a question would be the pose namely the position and orientation of each object of interest on the image. Finally, comes the last question as *Segmention* and precisely *Instance Segmentation*. That is about finding all the regions of the input image that are exactly part of a given object of interest. This is achieved by assigning each pixel to a single object of interest. These four questions that constitute the recognition problem in Computer Vision are essential in Robot Vision.

Secondary Building Blocks [94]. As noted before, Computer Vision is a parent of Robot Vision. Building blocks in the field of Computer Vision that are either not directly relevant to Robot Vision or just specialized blocks that can be tracked with the help of the recognition block but difficultly, are for instance *Motion Analysis*, which focuses on issues related to motion in videos such as tracking and surveillance. Another one is *Scene Reconstruction* which figures

out how a scene can be synthesized from a given set of related images acquired from the original scene. The last of these secondary building blocks of Computer Vision is *Image Restoration* which basically consists in denoising input images. Since input images are quite often noisy, this building block is implicitly bound to the beginning of every building block and commonly referred to as *data (i.e. image, video) preprocessing*.

Building Block Solutions

Old-School CV [132]. These are the oldest methods used in artificial vision systems to interpret images and videos. Their particularity is twofold. Firstly, they deeply rely on naive knowledge about the scene's geometry and physics to extract the scene's semantics. Besides it, they are completely procedural meaning that they are specified from the first instruction to the last instruction about how to interpret visual contents. Briefly spoken, the processing pipeline starts with a scene acquisition. The resulting image or video is then denoised and important features (e.g. edges, corners, etc) are extracted from it, reducing then the dimension of the input and clarifying its content. Finally, an operation such as indexing, matching, estimating or any combination is applied on the features to achieve the final function, which is either tagging, detection, segmentation, etc. This approach, though successful (i.e. fast and accurate) in well-controlled and isolated scenes (e.g. industries), has appeared to fail in slightly relaxed environments. This is due to the unsatisfaction of the criteria cited earlier namely embodiement, situatedness and connectionism. And this unsatisfaction itself results from the naive assumptions made earlier about the scene (i.e. idealization), ignoring therefore the real relations among the robot vision's task, its hardware and operating environment. For instance, to compute the pose of an object in this old school of CV, the scene has to be accurately landmarked. Even the estimation of the apparently most simple feature of scene objects which is color has shown to be very tricky (i.e. color constancy) when one goes beyond scene idealization.

Traditional Machine Learning [71]. This approach has extended the capabilities of the old-school CV a step further with the concept of learning. Instead of merely relying on naive assumptions due to scene idealization and coming out with hand-crafted symbolic procedures, traditional Machine Learning has tried to enforce the concept of situatedness in artificial vision systems by considering the real relation between the vision system's task and its operating environment, however loosely. The processing pipeline starts similarly as in the previous approach, however after the phase of feature extraction, instead of applying a deterministic hand-crafted symbolic procedure such as indexing, matching, estimating or any combination, a parameterized model (e.g. DT(s), SVM(s), K-Nearest Neighbours Classifier(s) (KNN(s))) is trained on a set of annotated data (e.g. labeled images, videos) to estimate the optimal parameters of the model. *Annotated data* means here that the training data are provided with some supervisory signals called labels, which inform the system about how good it performs. This parameter estimation is called learning and inference occurs only later when the model's optimal parameters are determined. During inference, an image or video is acquired and preprocessed, the features are

extracted and the parameterized model is applied on the features to achieve the final function (i.e. detection, segmentation, ...). Though traditional Machine Learning has improved the old-school CV, it has not challenged interesting problems of CV enough. The first remark related to this limitation is that learning occurs only partially and worse at the end of the processing pipeline. That is, feature extraction which is the most crucial phase of the processing pipeline is still hand-crafted and consequently leads to considerable information leakage. The second drawback of this approach is its weakness at effectively generalizing from learning due to the inability of the trained models to properly scale with big data. The reasons of this weakness at generalization constitutes the third drawback of this approach which is the shallowness of the underlying parameterized models in terms of parameters and approximation power. Most of these models, if not bounded to a particular class of problems (e.g. linear, quadratic, etc), quickly degenerate with the number of parameters (i.e. bias-variance problem). For its simplicity, low computational complexity and improvements over the old-school CV, traditional Machine Learning has been the standard approach in the field of Robot Vision.

Deep Learning [204, 143, 107, 12, 113, 65, 178, 177]. As indicated by the name, Deep Learning has not just pushed forward the strength of traditional Machine Learning to a great extend but has completely revolutionized CV. Empirical results on challenging CV problems have been so impressive that theoretical investigations of DL-based models have been being carried out. As results of such investigations, it comes out that DL-based models are programmed to success. Firstly, they strongly satisfy the criteria of embodiement, situatedness and connectionism by allowing an end-to-end learning through deep and intelligently structured artificial neuronal networks. Secondly, it has been proven that artificial neural networks are universal approximators, see Turing-complete. However, in contrast to most shallow models (e.g. DT(s), KNN(s)) which possess the property of universal approximation but only at a theoretical level, DL-based models dispose of a set of infrastructures that allow them to generalize in practice while learning on big data. Among these infrastructures are their own structure (i.e. connections, neurons, depth, activations), learning regularization techniques and their very flexible evolutionary training algorithm known as backpropagation. Another major strength of DL-based models is their great extensibility to new tasks. DL-based models can be easily extended or mutated to solve completely new or additional tasks while preserving their past knowledge thank to backpropagation and a technique called transfer learning. Finally, Deep Learning enables effective multi-tasking, yielding therefore to stronger unified models. To support the realization of DL-based models, specialized computing hardware architectures have been being developed, easing then connectionist computing also called parallel distributed computing. As far as the disadvantage of Deep Learning is concerned, the major one has been the astronomical rich and annoted set of data required for training DL-based models. Since the RV community is significantly and logically restricted compared to the CV community, DL-based robot vision has been significantly paralysed due to the lack of contribution in the formation of training datasets and typically rich ones for interesting problems.

Hybrid Methods [19]. The most intuitive hybrid approach adds a higher-level symbolic reasoning layer on top of the above standard approaches. while the standard layer deals with extraction of very complex patterns from noisy, unstructured and continuous data (i.e. images, videos) and then outputs meaningful symbols, the top layer conducts a higher-level symbolic reasoning on the outputted symbols to either fine-tune them or derive higher-level symbols and usually by integrating commonsense knowledge. Though this approach seems extremly promising at first glance, it effectively leads to embodied situated symbolic-connectionist vision if the standard or bottom layer is embodied, situated and connectionist. Moreover, the greatest vision task should take place at the bottom layer since symbolism is not well suited to learning and lower-level, noisy, unstructured and continuous visual contents.

1.2.2 Scene Understanding: "Robot Manipulation"

As seen in the previous section, visual scene understanding for a robot is about using its vision system to extract information that can be effectively exploited by the robot itself to efficiently improve the accomplishment of the task which vision was needed for. In this thesis, the focus is laid on human-scale manipulation tasks in household environments.

1.2.2.1 Robot Manipulation

The expression *"Robot manipulation tasks"* is vague enough to define at the first hearing what it means for the robot to manipulate. As indicated in the description of the project EASE, it is all about complex human-scale everyday manipulation activities.

Figure 1.8 EASE's Kitchen in the IAI-Bremen's Laboratory

source: [105]

Literally spoken, these are on the one hand activities that are commonly performed by human. And on the other hand, these activities require an intensive involvement of human hands. It then follows that robots are required to intensively make use of their hands to perform complex common human activities. However, human manipulation tasks go from very simple (a.k.a. atomic) tasks to very complex tasks. While atomic manipulation tasks would include actions such as grasping, picking up, turning, placing, lifting up, complex manipulation tasks would be regarded as goal-oriented long sequences of atomic tasks such as cooking, setting the table, cleaning up, carrying out chemical experiments or executing customer orders in supermarkets. The first remark one can point out is that completing such complex tasks do not merely require some robot vision and hands, but also considerable knowledge about which actions make up such activities, which objects are involved in, how to generate plans and how to turn those plans into actions. To prepare a pizza for instance, the robot should know the steps involved in, the ingredients, their prior locations and navigate around the room to fetch them. From this scenario, it is clear that commonsense knowledge, planning and navigation tasks, briefly spoken, do not belong to robot manipulation. This remark yields a much more narrow specification of what it means to the robot to visually understand the scene while manipulating. Additionally, given that this thesis was motivated by the project's EASE which, as already mentioned, majotarily focuses on fast-food cooking activities and disposes of an experimental kitchen, the robot's operating domain is reduced to that kitchen and its manipulation activities are restricted to cooking of fast-foods, for the sake of experimentation in this thesis. As shown by figure 1.8, the EASE's kitchen located in the IAI-Bremen's laboratory was specially designed to make it easy for the humanoid robots to navigate and have access to objects (e.g. no staircase, no keys for doors), abstracting consequently the above mentioned unnecessary navigation task interfering with the concrete manipulation task. Such a domain narrowing allows an effective evaluation of the robot cognition's performance on complex manipuation tasks.

1.2.2.2 Main Challenges

So far, it has been shown that Household Robotics and most importantly for the sake of manipulation is a great deal from an engineering as well as scientific point of view. It has been noted that the most effective way to achieve household robots that can competently perform everyday human-scale manipulation tasks was to go for humanoid robots enabled with an embodied situated symbolic-connectionist cognition. Moreover, it has been mentioned that, as far as the robot cognition is concerned, visual perception appears to be the key bottleneck and consequently deserves much more attention. Then, the fields of Robot Vision and Manipulation have been overviewed to a considerable extend, where their meanings were elucidated. Before reviewing works achieved so far in making household robots to visually understand their scenes for the sake of efficient and effective manipulation, it would be helpful to clarify the main challenges involved

in.

Scene's Semantic Representation. What symbols are meaninful while visually perceiving the scene for manipulation and how they should be formatted, is a central issue in addressing the Problem of Visual Scene Understanding for Robot Manipulation Tasks (PVSURMT), but also any computational problem [122]. That is, the scene's semantics (i.e. ontology) and the formalism (i.e. language) used to represent it are central to the underlying robot cognition (i.e. capabilities). An agent's semantic representation of the world, whether explitcit as in humans and most animals or implicit as in most insects, is the real connection between its environment, body and goals. Failure to provide the right scene ontology and formalism to describe it automatically and logically downgrades the agent's capabilities. To assess the relevance of such a scene's semantic representation, two criteria have been identified namely the criterium of *Completeness* and the one of *Structuredness*. While completeness indicates how informative the scene ontology is and can allow the robot to effectively and efficiently carry out manipulation tasks, structuredness is all about how accessible by the robot the ontology formalism is. A robot cannot for instance grasp an object if the pose of objects is not part of the scene ontology (i.e. incompleteness). On the other hand, it is quasi-useless to build a vision system that turns visual contents into natural-language texts since natural language, being an extremly ambiguous formalism, is challenging for pure computational artificial agents such as robots (i.e. unstructuredness) [134].

Uncertainty. It refers to the state of a system that faces a lack of knowledge while making decisions. Except in well-controlled and isolated environments (e.g. industries), vision in complex environments, such as indoor kitchens, is an inherently ill-defined and -posed problem. That is, given a visual content such as an image or video, there may exist, loosely spoken, several rational interpretations of it. In such circumstances, the agent's vision system must be sufficiently robust to deal with uncertainty in order to properly behave and consider the most rational interpretation, as humans and animals usually do. This ill-definedness and -posedness of vision has made Computer Vision hard and even AI-complete. Uncertainty in vision has many sources; The most natural and tricky of them being occlusions. Occlusion is a situation in which scene must be interpreted but only from a partial view of it. Remember that an image is only a partial view of the whole scene, and even in the case of multi-view visual contents such as videos, intra-occlusion, which occurs when scene objects hide each other, is unavoidable. To deal with uncertainty, humans usually rely on a set of tricks. The most naive and uncommon of them consists in coming closer to the objects, moving objects around, then having a multi-view representation of the scene in order to avoid any kind of occlusion. Since this trick considerably interrupts and slows down the agent activity's flow, it could only get little attention from human vision. Contrarily, it has been noticed that human vision deeply relies on scene context as well as knowledge acquired through learning from past experiences to deal with uncertainty. That is, even without touching a coffee mug, experience suggests that it is made up of ceramic material (learning). In the same way, a partially immerged utensil within a mug of coffee is more likely to be a spoon than a fork or knife (context). The second common source of uncertainty, which also results from the

sensor model and the scene model, concerns the content quality of acquired images. More than loosing the scene depth, RGB-cameras produce images, whose content qualitiy is significantly degraded by a number of scene-camera-related factors among which the scene illuminatedness (e.g. intensive sun, shadows, interference) and the camera's viewfield and pose. Once again, human vision overcomes such an uncertainty through context consideration and learning-based adaptation (e.g. color constancy). Finally, there is no a priori computation model for robots to perform vision tasks, leading consequently to probabilistic computation models. This whole analysis about uncertainty suggests that a good vision system should considerably rely on scene context and learning to deal with uncertainty when interpreting scenes. Furthermore, it is suggested to inform about the level of confidence of results, commonly called score, in order to allow uncertainty to be mastered all over the robot system, from perception through higher-level reasoning on percepts till action [19].

Realtimeness [106]. As mentioned before, Realtimeness is the ability of a system to react as fast as possible to stimuli from its environment in order to satisfy the constraints of the latter's dynamics. This definition stresses the point that though realtimess is a time-related concept, it is not absolute, but rather intimatedly linked to the dynamics of the agent environment. That is, no matter how fast or slow the agent is, it should just be available whenever sollicitated. When designing a system, realtimeness is a crucial performance criterium whenever the system sollicitations are of an extrem importance (e.g. manufacturing factories). Given that indoor environments such as kitchens are extremly dynamic and that kitchen activities such as cooking are time-constrained, a robot that operates under such conditions should demonstrate realtimeness. Since this thesis cares about visual scene understanding in human environments, a systematic way of assessing a robot vision system's realtimeness, as quite often done, consists in measuring the number of images processed by the robot vision system per time unit and then comparing this number to the one achieved by the human vision system. Though the human vision system seems to perceive pretty well today's videos playing at the standard frame rate of $25Hz$, the individual processing of each frame is illusive. In fact, some frames are regularly skipped and no effect is noticed due to the excellent continuity of the frames. This has been noticed by observing the behavior of the human vision system on videos made from random frames and playing at the same frame rate. Researches has shown that it is expected to operate at a frequency between 7 and $13Hz$ [197].

1.2.2.3 Semantic Scene Representation: "Scene Graph"

Representation of the scene's semantics has been noted as determinant and central for the robot coginition, with two evaluation criteria notably structuredness and completeness.

Structuredness

To achieve structuredness, the use of an unambiguous formalism or language to visually describe the scene is enough. Among the most common of them are First Order Logics (FOL(s)) and Description Logics (DL(s)). However, DL(s), for some reasons, have found much more attention in practice than FOL(s). The first reason is that FOL(s), though extremly expressive (i.e. ability to represent), are not decidable (i.e. superset of Turing-recognizable languages). That is, given a question formulated in such a language, it is not always possible to algorithmically answer the question. That is so bad for practical applications because any access to a world representation, formatted in FOL(s), would be framed as a question. To address this issue, restricted FOL(s) (i.e. subclasses of FOL(s)) have been derived and one class of them are DL(s), which are not just impressively expressive and decidable, but also quite natural (e.g. only concepts, binary relations and graphical/textual representation) [199].

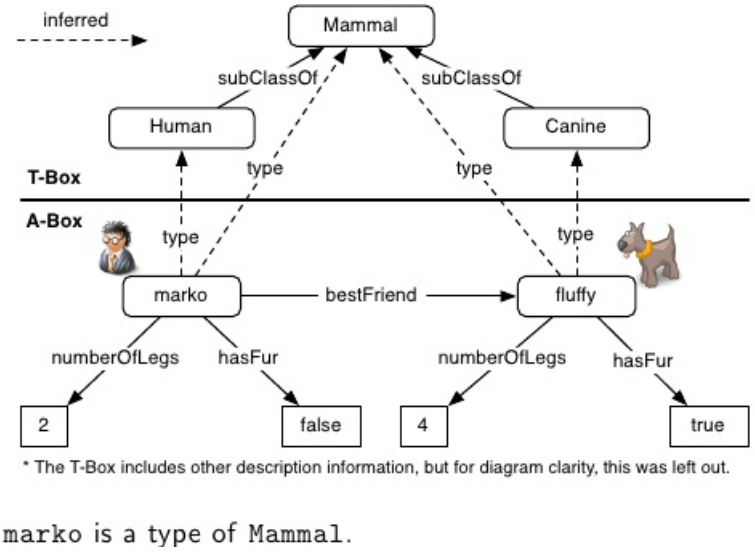


Figure 1.9 Example of T-Box and A-Box

source: [158]

To represent an agent's world using DL(s), there are two steps. The first step consists in writing a set of logical rules to express general knowledge (i.e. ontology) about the agent's world. Such a description also known as T-BOX (i.e. terminological box) has for goal to define the concepts (e.g. Human, Mother, Name) of the agent's world and the roles (i.e. binary relations, e.g. has-a, is-a) that link the concepts together. However, notice that such an ontology (i.e. abstract roles and concepts) remains very abstract since it is not grounded in reality. For instance, one may state that a Spoon has a Color, but what is concretely a Color? To ground the ontology in reality, concrete concepts (e.g. set of names, numbers and colors) as well as concrete roles, to connect abstract concepts to concrete ones, are introduced. Secondly, an agent's specific world, known as A-Box (i.e. assertional box) and made up of specific facts, should be built by

instantiating the T-BOX. The union of the T-Box and the A-Box would result in the agent's knowledge base. After building the agent's knowledge base, various questions (e.g. satisfiability of A-Boxes and T-Boxes, membership, selection, insertion, deletion) about the agent's world can be asked to the agent's inference engine. Analogically, T-Box's abstract and concrete domains respectively represent schemas and dictionaries in the context of relational database. In this same context, A-Boxes represent filled tables, and while the agent's inference engine acts as a DataBase Management System (DBMS), questions substitute queries [199, 118]. Figure 1.9 illustrates the modeling and exploitation of a world using DL(s).

While applying such a formalism to describe scenes in Computer Vision, the resulting knowledge graph, as shown in figure 1.9, is called a *Scene Graph*. Note however that untill this thesis formally does it, this concept has been only informally approached so far [214, 215, 95].

Completeness

Now that a formalism that can enable a structured representation of the scene's semantics is known, the next step will be to find what constitutes the content of such a representation. That is, how can one say on the basis of the content of the scene's semantics that the robot is able to perform manipulation tasks. From an image, there is an unexhaustive amount of information that can be extracted. Therefore, it is unthinkable to deal with a brute force approach as the traditional Turing Test might suggest [200]. And even if the amount of relevant information was exhaustive, brute force would severely penalize the system performance in terms of accuracy as well as computational complexity. Contrarily, only the relevant information for complex manipulation tasks should be considered. A powerful tool for tracking such information are Use Case(s((UC(s)) as specified by Unified Modeling Language (UML) [4]. While designing a system, instead of randomly listing a set of requirements, UC(s) attempt to come out with only relevant requirements by defining and parsing typical scenarios in which the designed system can participate. Note that even if there may exist infinitely many scenarios, the number of system functionalities is still finite and can therefore be tracked with a finite number of scenarios, selected based on the intended goals. A typical robot manipulation scenario for making basic instant coffee, as described in [206], would run as indicated by the following procedure 5.

Algorithm 1: Make Basic Instant Coffee

Data: Planner, Commonsense Knowledge, Navigator, Tools and Ingredients

Result: A cup of hot sweet milked coffee ready to drink

```
/* Phase 1: Heat up a cup of water                                          */
1 The robot detects a ceramic or glass mug ; // Visual detection and recognition of object and material
2 The robot frees the mug ;                  // Visual recognition of containment relations among objects
3 The robot picks up the mug ;               // Visual localization and segmentation of object
4 The robot looks for a free safe place ;    // Visual segmentation of the scene
5 The robot places the mug at the free place ;  // Visual segmentation of the scene, recognition of
  shapes and materials
```

```
6  The robot fetches a bottle of water ;                                    // See steps 1-3
7  The robot opens the bottle of water ;                                    // Steps 1, 3 (bottle cap)
8  The robot pours some water into the mug ;                                // Visual recogntion of color (fill level)
9  The robot detects and localizes the microwave ;                          // step 1-3
10 The robot opens the microwave oven ;                                     // Step 3 (open button)
11 The robot frees the microwave oven ;                                     // Step 2
12 The robot places the mug inside the microwave oven ;                     // Step 5
13 The robot starts the microwave and waits for some time ;                 // Step 3 (start button)
14 The robot fetches the mug from the oven ;                                // Steps 10, 3-5
   /* Phase 2: Add 1 to 2 teaspoons of instant coffee to the mug           */
15 The robot fetches the box of Coffee ;                                    // Steps 1-5
16 The robot fetches a steel spoon ;                                        // Steps 1-5
17 The robot fetches a new ceramic mug ;                                    // Steps 1-5 + episodic memory(filled mug)
18 The robot adds 1 to 2 spoons of coffee to new mug ;                      // As done in step 8
   /* Phase 3: Dissolve the coffee with a tablespoon of cold water        */
19 The robot picks the bottle of cold water ;                               // Step3
20 The robot opens the bottle ;                                             // Step 7
21 The robot picks the spoon up ;                                           // Step3
22 The robot fills the spoon with cold water ;                              // Step 8
23 The robot pours the spoon content into the new coffee mug ;              // Step 8, very difficult!
24 The robot uses the spoon to mix the the coffee with cold water ;  // Visual recognition of containment
   relations (spoon in mug before spoon agitation)
   /* Phase 4: Pour the hot water into the mug                             */
25 The robot picks the mug of hot water up ;                                // Step 3
26 The robot pours some hot water into the coffee mug ;                     // Step 8
   /* Phase 5: Mix in sugar or spices, if desired                          */
27 The robot fetches the box of sugar ;                                     // Step 6
28 The robot pours some sugar into the coffee mug ;                         // Step 8 or Steps 21-23
29 The robot mixes the content of the coffee mug ;                          // Step 24
   /* Phase 6: Add milk or cream if you're not a fan of black coffee       */
30 The robot fetches the box of milk ;                                      // Step 6
31 The robot pours some milk into the coffee mug ;                          // Step 8
32 The robot mixes the content of the coffee mug ;                          // Step 24
```

Figure 1.10 Robot on Making Basic Instant Coffee. This recipe is made up of of 6 phases proposed by [206]. Each phase consists of several steps and for each step, the robot vision system's requirements are commented beside. These steps as well as the robot vision system's requirements are greatly inspired by Robotics' constraints and human experience.

As shown by the above typical robot manipulation scenario, the robot visual system's mission in manipulation tasks would consist in the description of interesting scene objects as well as in the

description of spatial relationships among those objects. While object description (e.g. category, shape) informs the robot about the nature of scene objects and consequently how to deal with them, spatial relationships (e.g. on, in, left) among them provide it (i.e. robot) with sufficiently great insights into the objects' space occupancy and how to move them around the scene. As far as what exactly constitutes the object properties and the spatial relationships are concerned, note that though their nature is importantly suggested by the use cases, their very content results from human experience and Robotics constraints. For instance, the spatial relationships *In, On* are very crucial for the robot, in order to know, even if it exactly knows the pose of an object, whether or not the object can be directly grasped; That is whether or not there is something inside or on top of it. On the other hand, while the object category would inform the robot about what to do with the object (e.g. affordance), its pose as well as segmentation mask would give insights into how to grasp it. This objet description and spatial relations, which contain much more properties, will be detailly revisited in chapter 3.

Another very important point is that these information (i.e. object properties, spatial relationshps) can be extracted merely by observing scene images rather than scene videos. The choice for images as inputs further reinforces the system's realtimeness and enlarges its application domain (i.e. any image, any video). However, notice that one could argue that an analysis of object motion would not be possible with images. An objection to such an argument is firstly the fact that image processing in this context appears as a greedy approach to video processing. That is, such a greedy approach would still yield an analysis of object motion, since the proposed image analysis tracks the pose of each object. Furthermore, a computing mechanism can be provided to greedily integrate the results of analyses on past images into recent analyses: a kind of infinte impulse response on a sequence of images (i.e. video). This objection also rejects the argument that videos enforce coherency in the analysis much more than images. Finally, the robot vision system provides a confidence probability for each atomic information it returns, in order to allow further handling of uncertainty to take place from higher-level reasoning till actions.

While formalizing the whole in DL(s), the top concepts would be *Property* and *Object*. Then, the concept *Property* would be specialized (i.e. inheritance) through *is-a*-relations into the concepts *(Category, Pose, Segmentation Mask, etc)*. While a relation *has-a* would go from the concept *Object* to each specialized concept *(Category, Pose, Segmentation Mask, etc)*, the spatial relations *(is-on, is-in, is-left-of, etc)* would loop around the concept *Object*. Then, each specialized concept *(Category, Pose, Segmentation Mask, etc)* would be linked to a concrete domain *(set of categories, shapes, etc)* through concrete relations *(membership predicate)*. This construction would result in the T-Box of the robot's knowledge base. When a scene image is analyzed, new facts are inserted into the knowledge base's A-Box by instantiating the T-Box's rules with information out of the image analysis. Finally, each statement (i.e. predicate) in the A-Box as well as in the T-Box is assigned a truth propability p and each instance of the concept *Object* is provided with a timestamp t to note the instant at which it was inserted into the A-Box. Whereas the timestamp t allows to view the knowledge base not only as semantic but rather also as episodic,

the truth propability p enables a probabilistic reasoning on the knowledge base. Note that the strength of this semantic scene model is further testified by a very recent study that aims at framing the Visual Turing Test [60].

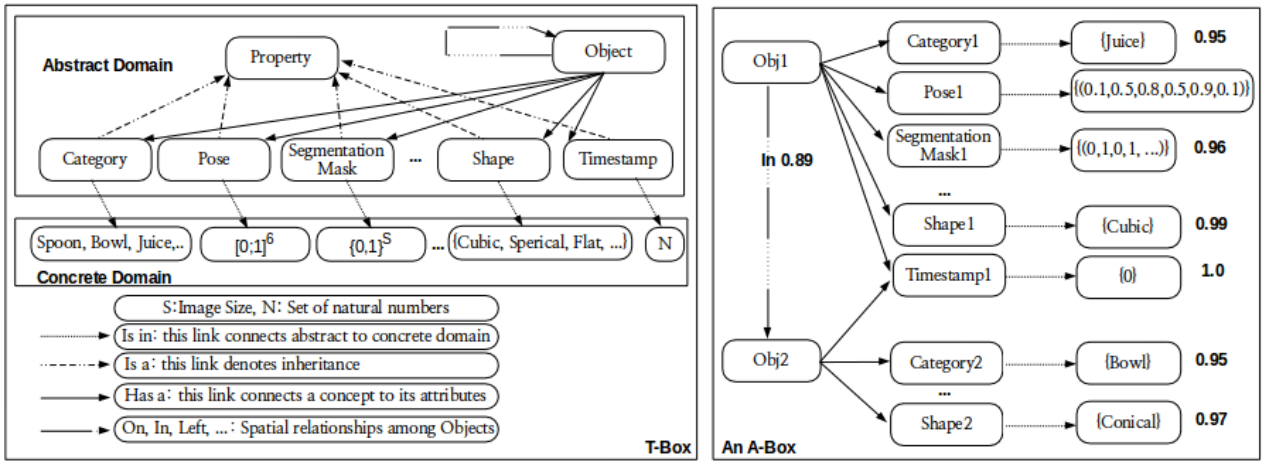


Figure 1.11 A typical scene graph constructed by the vision system of a robot (humanoid) to enable it to efficiently and effectively perform manipulation tasks (e.g. kitchen activities). The underlying formalism is *P-ALCNHr+(D)*, a decidable very expressive fragment of DL(s). Note that any statement is assigned a truth probability. Certain statements are implicitly assigned the truth probability of 1.

As far as the implementation of such an ontology is concerned, note that there are several fragments of DL(s) (e.g. *ALC, EL, ALCN, ALC(D)*) with different levels of expressiveness depending on the restrictions or extensions made upon the pure DL(s) *ALC*, which is decidable as stated in the previous section. While the most expressive DL(s) (e.g. *ALC(D)*) are undecidable, the least expressive ones (e.g. *ALC*, *EL*) more than being decidable, are more inference-efficient. The above ontology, proposed to semantically describe the robot scene, can be implemented with a decidable and very expressive fragment of DL(s) known as *P-ALCNHr+(D)* [118, 117, 69]. Figure 1.11 sketches the T-Box and an A-Box of such a robot scene's ontology with *P-ALCNHr+(D)*.

1.2.2.4 Related Works: "RoboSherLock"

After giving the motivations for developing household robots that can competently perform human-like manipulation tasks and the key role that visual perception plays in such robots, the challenges raised by the construction of such robots' vision systems were intensively discussed. In this section, past works, including RoboSherlock, achieved while addressing the PVSURMT are critically reviewed.

Traditional Approaches

[130], [104], [182], and [54] detect and localize objects to enhance autonomous robot manipulation. However, their approaches present severe limitations. They first consider color- and shape-idealized objects as well as landmarks in uncluttered scene. And based on this assumptions, they apply very shallow algorithms such as 3D-CAD-Model-based matching algorithms (e.g. point set registration) and color-based segmentation (i.e. structure- and texture-irrelevant). Moreover, the amount of information provided, with regard to the above scene graph, is considerably limited. On the other hand, [98] merely tracks the distal end of tool-like objects (e.g. Screwdrivers, Bottle oppeners, knives). In short, the distal ends of such tool-like objects are their extremity points (e.g. knife's cutting side, screwdriver's rotating head) that usually come in contact with the external world while working. Though this work does not really addresses the PVSURMT, it does study a feature of objects which might be extremly relevant to robots while performing manipulation tasks. Notice that this feature, namely the distal point of tool-like objects, has been captured several times (e.g. Spoon's head, bottle's head, bottle's cap) by the scenario 1.10 of *cooking basic instant coffee*, elaborated in the previous section. Furthermore, as indicated in the scenario 1.10, the authors reduce the problem of distal end estimation to the problem of detection and pose estimation. However, to solve these two problems, the authors firstly assume that the robot correctly holds the tool-like object and based on this assumption, they track the $2D$-position of the object's distal end based on the technique of optical flow (i.e. motion analysis). Relying on another assumption, that an object's distal end is the region that moves the more in the scene, they adopt the technique of stereo vision to recover the distal end's depth. The tracking, particularly of the distal point's orientation, is further supported by probabilistic models based on the robot hand's kinematics and dynamics. As results, the proposed model, though fast enough (15 frames/s on $3GHz$ pentium processor), and more than relying on weak assumptions, is based on approaches that have been earlier shown to be less efficient than DL-based approaches, especially combined with RGBD-cameras. Like [98], [37] does not actually address the PVSURMT, however it does focus on a crucial and almost untouched aspect of it: That is the recognition of object materials in the scene. This feature can further inform about the nature of the object, how careful to deal with, the weight and consequently the grasp force to apply when grasping and moving it. The work exploits traditional local visual features such as SIFT, Color SIFT, Linear Binary Pattern (LBP) and textons to recognize material by conducting a classification per pixel through a random forest. However, as also recognized by the authors, material is an extremly emergent property [56], which necessitates a processing of much more global features of the target rather than just examination of local features. As pointed out by the authors themselves, this strongly greedy approach (i.e. pixel classification on local features) might certainly be the cause of the lower accuracy of 34.5% shown by their random forest on the OpenSUrfaces dataset. For an improvement, the author consequently suggests an approach that jointly and globally examines the properties of objects in the scene together with their context: That is, a mug of coffee is more likely to be made out of ceramic. Additionally, due to their high

symbolicness, decision trees' complexity and accuracy degenerate on big high-dimensional data [4, 147, 160].

DL-based Approaches

In contrast to [98], [217] proposes a DL-based model to essentially detect and 2D-box-bound potential graspable or pressable areas in the scene. As key areas, count for instance handles of doors, windows, utensils, but also buttons on home items such as microwave ovens. Though such an operation should take place prior to and therefore relevant for actual manipulation (see scenario 1.10), it only provides very few information about how to perform higer-level manipulation tasks. Similarly to [217], [43] proposes multiple deep networks for task-oriented grasping. That is, given a set of objects in the scene and a specific task from a set of tasks (e.g. transport, handover, pour, open), a distinct deep network is trained to take the scene's depth map as input and localize the grasp regions of each object. Then, a matching algorithm, based on a large dictionary of gripper pose's prototypes, is run to estimate the gripper poses for each of the detected grasp regions. Though this work constitutes an important step toward robot vision for manipulation, it is considerably limited for enabling robots to perform higher-level manipulation tasks as it solely focuses on grasping. Furthermore, the strong coupling between higher-order outputs (e.g. gripper poses, grasp points) and lower-order inputs (e.g. scene's depth maps) through DL-based models makes the resulting system very inflexible. On the one hand, a very big and rich training dataset combining contexts (i.e. tasks, backgrounds, objects), grasp poses and grasp points will be needed. And on the other hand, more than requiring several deep algorithms, the knowledge gained by each of those algorithms will only very difficultly contribute to the resolution of other sub-problems related to the PVSURMT. To ease the collection of the training dataset, the authors rely on automatically annotated synthetic images, but by considering only depth images, firstly to avoid the difficulties resulting from modeling colorful scenes and secondly, the authors state that while depth maps are almost enough for addressing the underlying problem (i.e. grasp pose estimation), contrarily to color images, they reinforce generalization by abstracting color-based constraints. However, this inability to fully modeling the world could be the very reason for the restrictedness of the underlying problem when compared to the PVSURMT. Though [211] does not propose an explicit solution to holistic scene understanding (i.e. PVSURMT), it deeply investigates it. Firstly, it recognizes the fuzziness of the concept *scene understanding* and the dependance of its meaning on the underlying end-application (e.g manipulation). Moreover, the author suggests that a success toward visually understanding scenes necessitates the system to possess a very rich representation of the scene and insists on the prowess of data-driven approaches notably DL-based approaches to address this problem. However, the author warns about the big rich data needed to achieve a rich representation of the scene while relying on DL-based models. Finally, the author splits the problem of holistic visual scene understanding into two sub-problems: the spatial and semantic understanding of the scene. Whereas the second deals with object description, the former regards the spatial distribution of objects in the scene. Notice that this work further motivates the idea of scene graph as semantic representation of the

scene. Like [211], [72] points out the fact that scene understanding is a very important prerequisite for any indoor-agents such as robots and further recognizes the prowess of Deep Learning in Computer Vision, however deplores the big rich data needed to achieve it. And similarly to [43], it demonstrates the ability of DL-based models to perform well on scene-depth-driven tasks such as segmentation when they are trained on synthetic depth images. [110] introduces a DL-based approach to detect grasp regions on single patch-imaged objects. Though the system leverages the computation power of DL-based models, the extracted information and system's unsituatedness (i.e. patch-imaged objects) are significantly limited to advance vision-based robot manipulation in complex scenes.

Hybrid Approaches

[119] enhances visual scene understanding through human-robot dialogue. Firstly, the authors recognize the necessity of a deep visual scene understanding for robot manipulation tasks. Secondly, they even suggest a definition of the PVSURMT as knowing about separate objects in the scene. However, they just focus on instance detection and segmentation using shallow methods such as color-based clustering and stereovision-based scene's depth recovery which are not only noise-sensitive, but also texture- and shape-irrelevant. To deal with uncertainty while detecting and segmenting, the robot engages in a dialogue with a human to confirm its hypotheses. Beside the challenges posed by natural language processing in such a dialogue, the presence of a human assistant near the robot is required . As presented in [19], RoboSherlock is the leading cognitve robot vision system used in the project EASE. The system has been designed with a very strong philosophy that actually aims at a flexibly extensible, learnable and robust (e.g. uncertainty, representation) vision system. However, RoboSherlock is still a bit far from demonstrating an embodied situated connectionist cognition, which, as indicated earlier, is of great importance for robot vision systems. In RoboSherlock, flexible extensibility is achieved by breaking down the processing pipeline into well-defined phases namely and orderly image acquisition (Red,Blue,Red,Depth (RGBD)), object segmentation, object description and finally refinement of object description based on higher-level symbolic probabilistic reasoning. As far as robustness is concerned, it is achieved in many ways. Firstly, RoboSherlock aims at a structured and complete description (e.g. scene graph) of scene objects; However it does not consider some inportant information such as scene objects' materials and spatial relationships among them. Robustness in RoboSherlock also importantly means the subdivision of the whole perception task into well-defined subtasks as well as the definition of a specialized solver for each subtask. This strong specialization of individual solvers relies on the founded assumption that there is no single solver for all problems. Notice that the foundation of this assumption, as mentioned before, is the very consequence of the simplistic assumptions which those solvers rely on. For instance, while object segmentation is color-based, object pose estimation is reached through $3D$-CAD-model-based matching algorithms, object color estimation leverages color histograms and shape estimation restrictedly focuses on standard shapes (e.g. circle, flat surface). Another aspect of robustness in RoboSherlock is the refinement of the derived properties of scene

objects through a higher-level probabilistic reasoning that integrates commonsense knowledge. This refinement, which consists in producing consistent results, is particularly important because RoboSherlock, in the goal of maximizing its accuracy, allows many different solvers to operate on the same subtask (e.g. a specialized algorithm for each shape) and which in turn more likely produce different results (i.e. inconsistency). For probabilistically reasoning, RoboSherlock relies on the Markov Logic Network (MLN)-framework which itself combines a decidable fragment of probabilistic FOL(s) and Machine Learning. Notice however that this symbolic reasoning and learning that take place lastly is only very effective and efficient if the lower-level solvers and learners are good enough, which unfortunately seems not to be the case. Finally, [13] criticizes the shallowness of RoboSherlock's solvers and goes for DL-based models. It leverages Deep Learning to describe objects in the scene and then runs a higher-level symbolic reasoning for refining the results. Similarly to RoboSherlock, the goal of refinement is to provide consistency in the results delivered by the deep networks, but as done in [148], for the only sake of recognizing new objects without any training. For instance, the deep network, operating at the lowest level of the architecture, may detect a yellow scene object, filiform and eatable. However, not being able to determine the category of the object because it was not seen by the deep network before, the higher-level symbolic reasoner is called to leverage a very rich knowledge base (i.e. ontology) in order to reach the conclusion that an eatable filiform yellow object is more likely to be a banana. Though this work combines Deep Learning and higher-level symbolic reasoning, it misses some information extremly relevant to complex robot manipulation such as spatial relationships among scene objects, object pose and instance segmentation maks. Besides, the system, exactly as in [110], is unsituated since it can only operate on single patched-imaged objects (i.e. granted object localization, no occlusion). Lastly, for each object property (e.g. color, shape, material, affordance), a distinct deep network is trained to estimate it, increasing consequently the system's computational complexity and ignoring inter-task dependencies.

1.3 Problem & Goal Statement

Problem Statement

Robots that can perform complex human-like manipulation tasks in home environments have received very great attention in the fields of Robotics and AI (e.g. EASE), both from an engineering point of view when regarding the very broad range of applications it enables as well as from a scientific viewpoint since it might give insights into behaviors of biological entities. However, visual perception has been shown to be the key bottleneck of such robots due on the one hand to the crucial role it plays and on the other hand to the task- and environment-dependent challenges it raises. To address this problem, generally known as PVSURMT, careful investigations of biological agents (e.g. humans, animals) suggested to develop artificial vision

systems that demonstrate at least embodied situated connectionist cognition. In short, that is, for an agent's vision system to be successful, the agent's body (e.g. morphology, phisiology) that supports it, the environment in which the agent operates and the agent's goals should be considered. Furthermore, the agent's vision sytem should exhibits neural-network-like behaviors such as directly interpreting and learning from noisy, unstructured and continuous data (i.e. real world).

To better situate the state of the art reached while addressing the PVSURMT, the challenges raised by such an embodied-situated-connectionist-cognition-enabled vision system have been identified and solutions based on systematic investigations of current hardware technologies as well as theories in the field of Computer Vision have been rationally proposed. In conclusion, it has been noted that while dynamically capturing scene images with a RGB(D)-camera, the robot vision system may rely on DL-based models to learn how to infer from those noisy, unstructured and continuous data (i.e. scene images) a very rich semantic representation of the scene such as scene graph, that can allow the robot to effectively and efficiently perform complex manipulation tasks. However, intensive review of the literature indicates that weaknesses of state-of-the-art approaches at properly addressing the PVSURMT importantly rely on failure to meet this theoretical schema.

Given the failure of state-of-the-art approaches to implement the above promising theoretical schema, rationally derived to address the PVSURMT, this thesis aims at implementing it. However, DL-based computational models, especially to address the PVSURMT, require enormous rich annotated training data, which are practically spoken infeasible. Fortunately, as shown in the literature, synthetic data, automatically generated from virtual worlds, constitutes a good theoretical solution to this Big Data crisis. This being said, the research question that this thesis seeks to answer, is whether or not by relying on DL-based models and leveraging virtual worlds for training them, one can achieve an artificial vision system that takes as input RGB(D)-images of the robot scene and outputs a very rich semantic representation of the scene such as scene graph, that can allow robots to effectively and efficiently perform complex human-like manipulation tasks.

Goal Statement

Notice that the theoretical answer to the above research question is positive, since the question itself is derived from questioning the practicability of a theoretical solution scheme, rationally inferred, for addressing the PVSURMT. That is, this thesis is concerned with providing empirical evidences about the practicability of this theoretical solution scheme. To answer this question, a set of goals are set. Firstly, to be able to carry out empirical experiments, the proposed solution should be embodied and situated. Given that this thesis is motivated by EASE, the experimental environments are narrowed to the very rich kitchen of EASE and agents are assumed to be the EASE's humanoid robots, which are all equipped with RGBD-cameras. Then, the EASE's kitchen as well as robots are virtualized and a big rich annotated synthetic dataset for training

and testing the DL-based robot vision system to propose is automatically generated. To add real-world specificities to the collected synthetic data, the latter is augmented with a small set of annotated real data. Secondly, a DL-based model that takes a RGB(D)-image of the scene and infers in realtime from it the corresponding scene graph is designed and implemented. Finally, two types of experiments are carried out namely the test of computability and the one of knowledge transferability. The former aims at testing how good DL-based models can compute the PVSURMT. This is achieved by evaluating the performance of the system after training and testing it on synthetic data. As far as the second experiment is concerned, it aims at testing how good knowledge acquired by the robot vision system from the virtual world can be transferred into the real world. This is achieved by evaluating the performance of the system after training it on synthetic data and testing it on real data.

1.4 Roadmap

The presentation of this thesis has been organized in 6 chapters. The goal of this introductory chapter was to define the PVSURMT, explain why it deserves attention and clearly state the related questions the thesis aims at answering as well as the goals to achieve in order to reach an answer. As goals, the thesis seeks to design and implement a solution to the PVSURMT based on Deep Learning and Virtual Reality. In the second chapter, the theoretical background of DL-based Computer Vision as well as the state of the art is analyzed. The goal of this analysis is to identify key building blocks for the solution to propose. The third chapter focuses on the robot world virtualization as well as the collection of the data. The scene graph's structure and content determine the structure of collected data, which themselves determine the structure of the solution to propose to the PVSURMT. Then, the design as well as the implementation of the proposed solution (RobotVQA) to the PVSURMT is exposed in details in chapter 4. The following chapter namely the fifth deals with the solution testing, as described in the previous section, and then discusses the results. Finally, the last chapter concludes the thesis by summarizing it from the problem definition to the solution test and then presents some recommendations that could be integrated in future works for further improvements.

Chapter 2

Deep-Learning-based Computer Vision

2.1 Introduction

Earlier approaches to AI, commonly known as GOFAI, though successful on purely reasoning problems such as language generation and parsing, planning and search, have failed to capture important features of biological cognition such as perception, motorics, learning, embodiement, situatedness. In the field of perception, symbolic approaches were built based on very simplistic and basic assumptions from naive physics and geometry; These mechanisms completely failed when embodied and situated due to the unrealistic constraints they imposed on the real world. Furthermore, they lacked the ability to adapt to new stimuli [27, 133, 198]. In the late 1980s, a new paradigm called Machine Learning [198] was born to address these issues. Instead of building immutable symbolic procedure based on simplistic assumptions, parameterized models were built to determine (i.e. parameter estimation) themselves while learning from interactions with the real world. Among those models are Linear Transformations, Polynomials, ANN(s), DT(s), KNN(s), K-Means, etc. Unfortunately, though promising (e.g. speech recognition, vision, statistical logic), traditional Machine Learning still showed severe limitations. First, the very crucial phase of learning, which is feature extraction, is carried out by humans or manually. These human-engineered hand-crafted features, which simply rely on intuition and simplistic assumptions, as in the previous paradigm, are unavoidably biaised and lead to information loss in the whole learning pipeline. Another major flaw in traditional Machine Learning models is their inability to generalize due to their topology and learning principle. Decision trees (DT(s)), which are symbolic and tree-based, quickly degenerate in computational complexity and accuracy with increasing input dimension. Polynomials show strong overfitting and fluctuation whith increasing degree. Linear models get stuck on non-linear data distribution. Statistical relational learners which aim at combining probabilistic FOL(s) and Machine Learning become unmanageable with increasing logical knowledge base rules. Transfer learning is another aspect of cognition that is not allowed by traditional Machine Learning. Similarly to the concept of learning transfer in Psychology, transfer learning in Machine Learning refers to the ability of a system to exploit the

knowledge acquired from a domain A within a different but related domain B. In the late 2012, a new paradigm called Deep Learning arose and revolutionized AI and particularly the field of perception to an incredible level. While emulating the nervous system of biological agents, DL-based models have theoretically as well as empirically proved to be able to address most, not to say all, of the issues mentioned above. However, though Deep Learning has shown tremendous successes and breakthroughs in Computer Vision, it has not really yet been leveraged for robot vision systems, which are regarded as the key bottleneck of mobile robots.

This thesis aims at designing and implementing a DL-based vision system that can enable household humanoid robots to competently perform human-like manipulation tasks. For this reason, this chapter exposes the theoretical background of Deep Learning, related practical issues and past achievements in the field of Computer Vision. This analysis will serve as rationale for identifying the building blocks of a potential solution for household robot manipulators.

2.2 Fundamentals of Deep Learning

2.2.1 Computationalism Vs Connectionism

It has been mentioned several times before that the goal of AI and hence CV (e.g. RV) was twofold. On the one hand, it oughts to develop artificial systems that can solve complex problems well-known solved by biological agents and on the other hand to provide computational models of biological agents. However, it has been very controversial how biological cognition arises, giving therefore birth to two major schools of thought about the nature of human cognition namely the computationalism and connectionism schools. While the computationalists argue for a Turing-like information processing (i.e. discrete formal manipulation of symbols) in biological agents through the invisible mind, connectionists rely on empirical evidences that computation occurs in biological entities continuously, dynamically, parallely, distributedly, emergently and evolutionarily through the nervous system. Though it is not clear what the invisible mind actually is, there are indirect evidences (e.g. introspection, extrospection, intuition, theoretical arguments) that suggest that higher-level reasoning in biological agents in contrast to lower-level cognition (e.g. perception) occurs symbolically at a very mysterious-level domain, which is either non-physical or physically and subtly emergent. Supported by empirical results from both schools, this consensus has led to the adoption of hybrid cognitive systems where perception is handled by the connectionist part and higher-level reasoning by the symbolic part. Deep learners are purely connectionist models and consequently well suited for perception problems [187, 186, 27, 198].

2.2.2 Nervous System

Very essentially and abstractly, the nervous system appears as the visible information processor in biological agents and can be regarded as a complex network of cells commonly known as neurons, linked together by signal carriers called synapses. Both the neuron and the synapse constitute the core building blocks of the nervous system. The nervous system is made up of input neurons that continuously converts environmental stimuli (e.g. heat, light) into the appropriate internal signal (e.g. electrical signal). The converted signal is then simultaneously carried by multiple synapses to multiple internal neurons called hidden neurons. When carried by a synapse, the signal is either attenuated or amplified depending on the conductivity of the synapse. Then, depending on its state, each hidden neuron combines and transforms the signals from the incoming synapses and outputs a certain signal on all outgoing synapses: this step is called activation and constitutes the atomic level of information processing in the nervous system. The raw representation of the world initially held by input neurons are then processed into more meaningfull representations over and over while travelling accross the neural network untill the final representation is delivered by output neurons: ideally, these output neurons are motor cells that aim at exciting the muscles so that the latter produce the right motion. The nervous system is evolutionary in the sense that it can either regress or progress in its performance. While progression may be regarded as the development of new neurons, new synapses, adjustment of a neuron's dynamics or synapse's conductivity, regression can be assimilated to a synapse's or neuron's death. Whereas the progression of biological neural networks quite often results from learning and adaptation, failure to adaptation or learning is usually the cause of a regression.[159, 126]

2.2.3 Artificial Neural Networks

Perceptrons

This very essential and abstract description of neural networks in biological agents has led Frank Rosenblatt in 1958 to the development of the first artificial neural network known as the perceptron [162]. Note that his work was greatly inspired by the work of Warren McCulloch and Walter Pitts, who together in 1943 already proposed a prototype [125]. A perceptron is a very shallow artificial neural network that oughts to describe the local behavior of neural networks. That is, it describes inference (signal processing) and learning (evolution) around a neuron. Since such a theory on local behavior can be applied all over the neural network, it constitutes a building block for describing the behavior of any artificial neural networks. The figure 2.1 depicts on the top 3 interconnected natural neurons and an artificial neuron or perceptron at the bottom. As shown by the figure, a natural neuron consists of three parts namely the input gates called dendrites to receive signals from other neurons, a nucleous body to combine and transform all

the incoming signals into a single new outgoing signal and output gates called axons to forward the outgoing signal to other neurons.

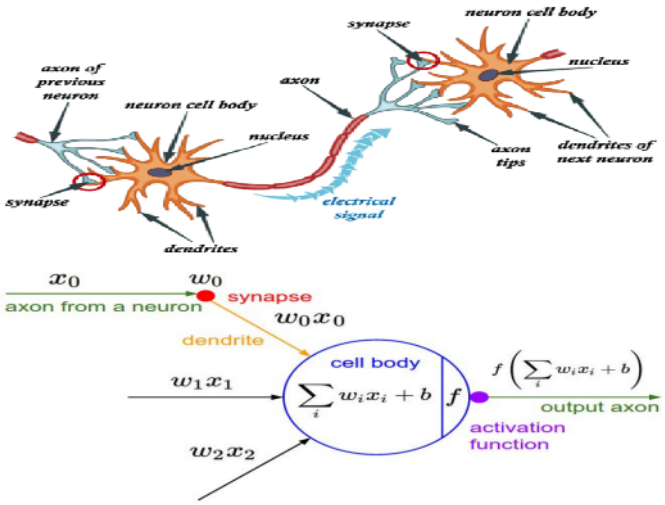


Figure 2.1 On the top of the figure, an interconnection of three natural neurons. At the bottom, an artficial neuron called a perceptron.

source: [180, 97]

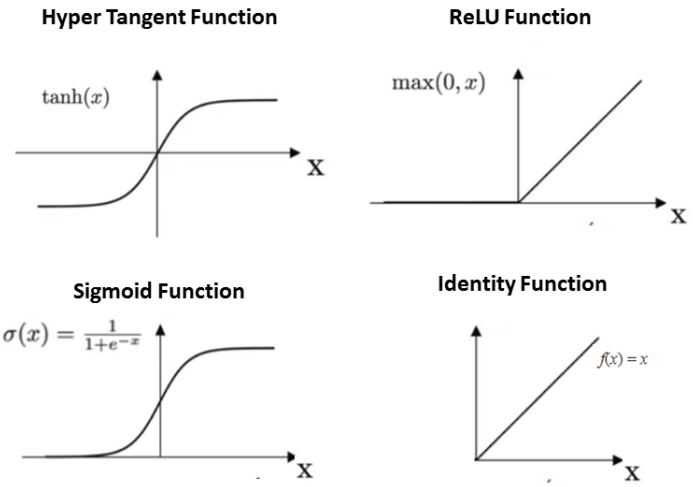


Figure 2.2 Major activation functions within perceptrons. $tanh(x)$ and $\sigma(x)$ aim at binary or bounded output signals. $f(x)$ merely forwards the input signals to output and $Relu(x)$ outputs unbounded primitive non-linear signals

source: [203]

These neurons are connected to each other through special links called synapses, whose conduc-

tivity, which can lead to signal attenuation or amplification, is entirely determined by learning and adaptation. In the perceptron of Frank Rosenblatt, a signal is represented by its a real-valued intensity I, a synapse conductivity is represented by a real number W, the influence of the synapse on the carried signal is represented by the convolution $W \times I$, the neuron is represented by a multidimensional real-valued function, whose the input vector represents the input signals, each dimension being representing a dendrite. The neuron combines the input signals by adding them together and then applies a (non)-linear transformation called activation function, whose output, considered as the neuron's output signal, is then duplicated and forwarded to other neurons. Figure 2.2 illustrates the major activation functions found in perceptrons and their respective main goals.

As far as the perceptron evolution is concerned, while setting the conductivity W to 0 leads to synapse's death, the death of all synapses around a neuron, computationally spoken, leads to the neuron's death. Since synapto- and neuro-genesis (i.e. birth of new synapses and neurons) take place majoritarily at the birthhood [126], the number of of neurons within a perceptron is fixed at the design phase and all the neurons are usually fully connected. Like a biological neural network, The perceptron learns by suitably adjusting the conductivity W of its constitutive synapses so that the error between the target signal and the perceptron's output signal is minimized: this artificial learning theory is inspired by its biological counterpart well known as the Hebbian learning rule from Donald Hebb [79]. Multiple perceptron learning algorithms have been developed, going from the most simple rules such as the basic perceptron learning rule and the delta rule (i.e.only suitable for basic neural networks) to the most general and successful rule namely the back-propagation rule, commonly based on the technique of gradient descent when the error function is defferentiable as usual. In this thesis, the gradient-descent-based back-propagation learning rule is considered for its generality and renowned success [101]. Given the above perceptron 2.1, two of its neurons namely an input one $\mathcal{N}_i$ and an output one $\mathcal{N}_j$, as well as the conductivity W_{ij} of the synapse linking $\mathcal{N}_i$ to $\mathcal{N}_j$, the back-propagation learning rule $\mathcal{BLR}$ for updating W_{ij} is given by:

$$\mathcal{BLR}: \begin{cases} Out_j^{(n)} = \sigma_j(b_j + \sum_{k=1}^{N} W_{kj} \times Out_{kj}^{(n)}), & \text{Inference or prediction in } \mathcal{N}_j \\ \Delta_{ij}^{(n)} \leftarrow -\alpha \times \dfrac{\partial E[Targ_j^{(n)}, Out_j^{(n)}]}{\partial W_{ij}}(W_{ij}^{(n)}), & \text{Evaluation of inference error} \\ W_{ij}^{(n+1)} \leftarrow W_{ij}^{(n)} + \Delta_{ij}^{(n)}, & \text{Weight update} \end{cases} \tag{2.1}$$

where,

$$\begin{aligned} &N, && \text{is the number of inputs of } \mathcal{N}_j \\ &W_{ij}, && \text{is the weight variable of the synapse linking } \mathcal{N}_i \text{ to } \mathcal{N}_j \\ &W_{ij}^{(n)}, && \text{is the weight of the synapse linking } \mathcal{N}_i \text{ to } \mathcal{N}_j \text{ at time } t = n \in \mathbb{N} \\ &\sigma_j, && \text{denoted by } f \text{ in 2.1, is the activation function of } \mathcal{N}_j \end{aligned} \tag{2.2}$$

$$
\begin{aligned}
&\text{applied on the input signals by } \mathcal{N}_j \\
Out_j^{(n)}, &\quad \text{is the actual output signal of } \mathcal{N}_j \text{ at time } t = n \in \mathbb{N} \\
Targ_j^{(n)}, &\quad \text{is the expected output signal (target) at t=n of } \mathcal{N}_j \\
E, &\quad \text{the error function, defines the distance} \\
&\quad \text{between the expected output signal and the actual output signal} \\
\Delta_{ij}^{(n)}, &\quad \text{the algebraic conductivity gained} \\
&\quad \text{by the synapse linking } \mathcal{N}_i \text{ to } \mathcal{N}_j \text{ after learning at time } t = n \in \mathbb{N} \\
\alpha, &\quad \text{the learning rate} \\
b_j, &\quad \text{bias (constant neural input signal) for } \mathcal{N}_j
\end{aligned}
\tag{2.3}
$$

Summarisingly, this bio-inspired gradient-descent-based back-propagation learning rule states that the adjustment of the weight $W_{ij} = W_{ij}^{(n)}$ at time $t = n$ of the perceptron's synapse linking $\mathcal{N}_i$ to $\mathcal{N}_j$ is logically proportional to the gradient of the perceptron inference's error function with respect to variable W_{ij} at time $t = n$. The proportionality factor a.k.a. perceptron's learning rate α is at least twofold-interpretable. On the one hand, the learning rate defines how quickly the perceptron forgets past experiences to consider new experiences. On the other hand, this learning rate expresses how quickly the perceptron approaches its optimal synaptic weights. This being said, while smaller learning rates slow down the learning process but stabilize it, bigger learning rates accelerate the learning process but are more likely to cause learning instability. Given that the error function E is not always convex (i.e. unique global mimimum), then smaller learning rates are also very likely to lead to local minima.

Multi-Layer Perceptrons

As mentioned earlier in this section, the perceptron theory constitutes the foundation of modern theories of ANN(s) including Deep Learning. Note also that the perceptron as a computational model belongs to the family of traditional machine learning methods. After the perceptron, a much more sophisticated class of ANN(s) was developed known as Multi-Layer Perceptron(s) (MLP(s)) which were theoretically as well as empirically proved to be more powerful than the perceptron (Universal Approximation Theorem (UAT)) [82]. However, due to their shallowness (e.g. hundreds of neurons, human-engineered hand-crafted inputs), they remained enclosed in the class of traditional Machine Learning methods. As a naive response to the practical limitations of MLP(s), was the considerably growth of MLP(s) in terms of number of neurons and synapses. This practice merely founded on the rational idea that the neural network's size is crucial for success, has been severely punished by practical issues and doomed to failure [7, 62]. Among the major practical issues are:

The exponential growth of computational complexity: also known as the problem of combinatorial explosion, due to the brute force approach consisting in naively and fully connecting

neurons to each other, this has been a major practical issue in dealing with very large MLP(s). Though biological nervous systems present such connections at birthhood, the underlying machinery has shown to be powerful enough to support it and quickly yield adequade adjustments (i.e. learning) of connections.

The unrealistically astronomical amount of training data: also known as the curse of dimensionality and closely related to the first problem, it implies that the mathematical function underlying such a deep network is so parameterized that any attempt to train it with any dataset of realistic size will be doomed to failure. That is, instead of generalizing (i.e. learning) its behaviour from the training data, the system will focus on specializing (i.e. retention or overfitting) its behaviour on the training data.

The unstable learning: in contrast to stable learning, which tends to converge and lead to improved behaviors, unstable learning manifests in three ways. The first manifestation is stagnation due to the well-known problem of vanishing gradient, which occurs when after computing the gradient-based error resulting from a deep computation (i.e. inference), propagating it back (i.e. training), as shown by the back-propagation learning rule [2.1], to adjust the synaptic weights fails. That is, the gradient-based error keeps decreasing as it gets backpropagated untill it reaches 0. At this state, the gradient-based error can no more contribute to the adjustment of remaining weights. As a consequence, the base weights (i.e. earlier synapses), which participate in the earlier and crucial stages (e.g. feature extraction) of the processing pipeline, remain unlearned. Default to the vanishing gradient problem, is the exploding gradient problem, where the gradient-based error resulting from the system inference keeps increasing, even infinitely, and causing the degeneration of synaptic weights during back-propagation, which in turn maintains the divergent behavior. Finally, unstable learning can manifest as a divergent but alternating behavior. In this case, instead of growing infinitely bigger and bigger, the training error shows an alternating behavior, usually caused by the huge behavioral difference between adjacent neurons (e.g. internal covariate shift).

Deep Learners

Deep Learning [62, 171] is a set of purely and bio-inspired mathematical theories that have been being developed to bring very complex artificial neural networks alive, as well as the development of specialized hardware architectures to support the physical realizations of those complex models, responding then to the first problem of exponential computational complexity. As another response to this problem, Deep Learning integrates founded prior knowledge, as well as functional, hierarchical and logical aspects in the design of deep networks, avoiding consequently naive brute-force-based fully connections. Notice that this design philosophy directly addresses the second problem namely the curse of dimensionality. As far as the problem of instability in learning is concerned, very important theories such as the theory of residual connections (i.e. ResNet) [77] and greedy learning theory [46] have been proposed to address the vanishing gradient problem. To solve the exploding gradient problem, special regularization theories such

gradient clipping [142], decay of synaptic weights and adequate activation functions have been developed [66]. Adaptative learning rate and the recently developed bach normalization theory [90] aim specifically at avoiding the problem of unstable (alternating) learning. Contrarily to popular conceptions about Deep Learning, notice that deep learners go beyond merely big MLP(s). Figure 2.3 illustrates the hierarchical relation among perceptrons, MLP(s) and deep learners.

2.3 Deep Learning Typology

Depending on how they interact with the environment during learning and why they learn, deep learners can be classified into 5 main categories notably supervised, unsupervised, semi-supervised, reinforced and evolutionary deep learners. While developing a deep learner, its training's design and goals are essential and intimately linked [171].

Supervised Deep Learning

In supervised learning [154, 64], the goal of the system consists in learning a function $\mathcal{F}$ given a restriction $\mathcal{D}$ of the target $\mathcal{F}$. $\mathcal{D}$ is called the training dataset. The system should end at the end of the training with a hypthesis $\mathcal{H}$ of the target $\mathcal{F}$. $\mathcal{H}$ is called a predictive approximation of $\mathcal{F}$ and is usually obtained by minimizing the error to $\mathcal{F}$ on the training dataset $\mathcal{D}$. When the range $\mathcal{B}$ of $\mathcal{F}$ is intrinsically continuous (e.g. orientation, position), the deep learner is called a regressor and a classifier otherwise (e.g. materials, colors, shapes, number of occurences). Formally,

$$
\begin{aligned}
&\mathcal{F} \subseteq \mathcal{A} \times \mathcal{B}, && \mathcal{A} \text{ is the domain and } \mathcal{B} \text{ the range of } \mathcal{F} \\
&\mathcal{D} = \{(p_1, l_1), ..., (p_n, l_n)\} \subseteq \mathcal{F}, && \text{is the training dataset} \\
&\mathcal{P} = \{p_1, ..., p_n\}, && \text{training set (images) from robot environment} \\
&\mathcal{L} = \{l_1, ..., l_n\}, && \text{the set of labels associated with above points} \\
&E, n, && \text{error function and size of training dataset} \\
&\mathcal{H}, && \text{a parameterized deep neural network} \\
&\mathcal{W} = (w_1, ..., w_m) \in \mathbb{R}^m, && \text{synaptic weights in and parameters of } \mathcal{H} \\
&b = (b_1, ..., b_k) \in \mathbb{R}^k, && \text{neural biases in and parameters of } \mathcal{H} \\
&(\mathcal{W}_h, b_h) = \underset{(\mathcal{W},b)}{\arg\min}(\sum_{i=1}^{n} E(\mathcal{H}(p_i), \mathcal{F}(p_i))), && \mathcal{F}(p_i) = l_i, \text{ learning goal: minimizes error}
\end{aligned}
\tag{2.4}
$$

Note that the target $\mathcal{F}$ and the hypothesis $\mathcal{H}$, such as defined above [2.4], are purely functional and deterministic. However, both of them can be probabilistically [181] interpreted and yield the following more general formulation:

$$
\begin{aligned}
&\mathcal{H} \equiv Pr(\mathcal{B}|\mathcal{A};(\mathcal{W},b)), && \text{a parameterized deep neural net to approximate}\\
& && \text{conditional probability distribution of } \mathcal{B} \text{ given } \mathcal{A}\\
&(\mathcal{W}_h,b_h) = \arg\max_{(\mathcal{W},b)}(Pr(\mathcal{L}|\mathcal{P};(\mathcal{W},b))), && \text{learning goal: maximizes conditional likelihood}\\
&Pr, && \text{denotes the probability function}\\
&\forall x \in \mathcal{A},\ \mathcal{F}(x) = \arg\max_{y}(Pr(y|x)), && y \in \mathcal{B}\\
&\forall x \in \mathcal{A},\ \mathcal{H}(x) = \arg\max_{y}(Pr(y|x;(\mathcal{W}_h,b_h))), && y \in \mathcal{B}
\end{aligned}
\tag{2.5}
$$

While $\mathcal{P}$ is acquired by the agent (e.g. robot camera) from its environment, the associated set of labels $\mathcal{L}$ usually results from a manual annotation and its nature is determined by the goal of the vision system, for instance the name of the scene, the list of objects, the pose of the robot camera, the presence of defects, etc. The more the training data points, the more burdensome the manual annotation is. Since any complex problem $\mathcal{F}$ by definition can probabilistically be regarded as a high-entropy large data source (i.e. all possible annotated images), the training dataset $\mathcal{D}$ should be big and rich enough as well to be able to capture it and yield a valuable predictive approximation $\mathcal{H}$ of $\mathcal{F}$ [111]. Supervised learning either in traditional or deep Machine Learning has been the most dominant form of learning, and certainly because it is very natural (i.e. human learning), explicit (i.e. proper supervisor) and practical (i.e. end-goals). However, interesting problems (i.e. complex) are still pending due to the lack of annotated training data. Fortunately, virtual reality has been promising with automatically generated and annotated synthetic data. As far as the supervised learning rule is concerned, the most common approach consists, as mentioned earlier [2.1], in applying the gradient-descent-based back-propagation learning rule around each of the deep network $\mathcal{H}$'s neuron to back propagate the gradient-based error from the ouput neurons to the input neurons of $\mathcal{H}$.

Unsupervised Deep Learning

Contrarily to supervised deep learners, unsupervised deep learners [64] train themselves without any supervision such as labels from annotation. While learning in an unsupervised manner, the system's main goal usually consists in modeling the probability distribution over the data points of a given domain $\mathcal{F}$ from a restriction D of $\mathcal{F}$; That is, it tries to maximize the likelihood of the synaptic weights, rather than the conditional likelihood as done by supervised learners. Formally,

$$
\begin{aligned}
&\mathcal{F}, && \text{robot scene's all possible images}\\
&\mathcal{P} = \{p_1, ..., p_n\} \subseteq \mathcal{F}, && \text{training set (images) from robot scene}\\
&\mathcal{H} \equiv Pr(\mathcal{I} \in \mathcal{F}; (\mathcal{W}, b)), && \text{a parameterized deep neural net to approximate}\\
& && \text{the probability distribution over } \mathcal{F}\\
&(\mathcal{W}_h, b_h) = \underset{(\mathcal{W},b)}{\arg\max}(Pr(\mathcal{P}; (\mathcal{W}, b))), && \text{learning goal: maximizes the likelihood}\\
&Pr, && \text{denotes the probability function}\\
&\forall x \in \mathcal{F},\ \mathcal{H}(x) \approx \Pr(x)), && \mathcal{H} \text{ approximatively infers } Pr(\mathcal{I} \in \mathcal{F})
\end{aligned}
\tag{2.6}
$$

Notice that the definition of unsupervised learning is only meaningful in the probabilistic interpretation (i.e. richer). Unlike supervised deep learning that aims at estimating a posterior probability distribution (e.g. a feature y given the others x), unsupervised deep learning estimates a prior propability distribution (e.g. all the features x). For these reasons, supervised deep learners are qualified as discriminative models, whereas unsupervised ones are known as generative models. This property makes unsupervised learning useful for a range of applications such as data compression, where the model tries to come out with a compact representation of data (e.g. dimension reduction, clusters). Another important application of unsupervised learning is data generation, which has received great attention in the field of creativity modeling (e.g. Art) and security (e.g. adversarial attacks and defences). Unsupervised deep learning is the second most dominant form of deep learning and in constrast to supervised deep learning which is end-goal-oriented, it quite often takes place (e.g. as data compression) in earlier stages of a supervised learning.

Semi-Supervised Deep Learning

Semi-supervised learners aims at the same goal as supervised learners. But in contrast to supervised learners, semi-supervised learners [129, 139] do not just rely on a set of annotated training data $\mathcal{LD}$, but go beyond it and integrates unlabeled data $\mathcal{UD}$ too, yielding a mixed training dataset $\mathcal{D} = \mathcal{UD} \cup \mathcal{LD}$. Notice that the approach aims at freeing hunans from the burden of manual annotation of big data. Generally, an unsupervised learning takes place earlier in the training phase and creates clusters centered around labeled data points $p_i \in \mathcal{LD}$. Each unlabeled data point $q_j \in \mathcal{UD}$ is then assigned the label l_k of the cluster center $p_i \in \mathcal{LD}$, it is associated with. At the end of this process, all the data points $p_i \in \mathcal{UD}$ will be annotated as well and the normal supervised learning will start. Note that this is only the general idea and may lead to different implementations (e.g. self- and co-training). Moreover, it is important to indicate that though this computing scheme seems very promising, it remains unwidely used for the main reason that it is not clear how to come out with a secure clustering script (e.g. no strong priors).

Deep Reinforcement Learning

Reinforcement learners [167], inspired by Behaviorism [123], share the same goal with supervised learners; That is, they learn a function $\mathcal{F}$ under some external supervision. However, due to the nature of the target $\mathcal{F}$, they act very differently while learning. First of all, the target $\mathcal{F}$ is only difficultly definable in terms of input-to-output correspondance. Unlike a supervised model which may be taught to recognize objects while presenting images together with labels to it, a typical reinforced learner is expected to learn how to drive a car, to play a game or how to walk, which requires it to dynamically plan and reach long-term goals. Notice that though these problems (e.g. walking, driving) might be functionally formulated, a practical formulation would be infeasible. This is due, on the one hand, to the fact that the world, being just partially observable at a time, prevents the system from inferring, based on a single observation, the plan to reach a long-term goal (e.g. driving). One could argue here in favor of supervised learning while dynamically training on annotated time series such as videos, however the annotation would be extremly tedious: there are too much possible sequences of actions to the same goal, causing on the other hand reinforced learners to behave as optimizers rather than functional mechanisms such as supervised learners. Since any optimization problem can be reduced to a functional one, proponents of supervised learning would go for such a reduction. Unfortunately, such a practice would drastically increase the complexity (i.e. even near randomness) of the resulting functional problem and making it very unsuitable for supervised learners. Secondly, to enable effective optimization and avoid burdensome big data annotations, instead of providing reinforced learners with training data points (e.g. videos) together with explicit labels, the supervisory signals are very implicit and known as reward and punishiment. That is, while interacting with its environment, the system has to dynamically discover by itself a plan that maximizes its reward. Whereas bad plan actions are punished but not corrected, good plan actions are rewarded. Such an approach makes the supervision very flexible (i.e. effortless but efficient and interactive). To deal with uncertainty, the problem is probabilistically interpreted and the core probabilistic framework to intuitively define the behavior of reinforced learners within their environment is the POMDP (Partially Observable Markov Decision Process). Formally,

$$
\begin{aligned}
&\mathcal{S}, && \text{the set of world states (e.g. images)}\\
&\mathcal{A}, && \text{the set of possible actions (e.g. moving left)}\\
&r \in \mathbb{R}, && \text{the supervisory signal (i.e. punishment/reward)}\\
&\mathcal{A}m \subseteq \mathcal{A} \times \mathcal{S}, && \text{the motion model is granted (i.e. robot effectors)}\\
&\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R}, && \text{the reward function is granted (i.e. from human annotation)}\\
&\mathcal{F} \subseteq \mathcal{S} \times \mathcal{A}, && \text{the action policy}\\
&\mathcal{P} = \{s_1, ..., s_n, ...\}, && \text{a sequence of world states (e.g. images) observed by the robot}\\
&\mathcal{L} = \{a_1, ..., a_n, ...\}, && \text{the sequence of actions associated with the above states}
\end{aligned}
\tag{2.7}
$$

$$
\begin{aligned}
&\mathcal{RE} = \{r_1, ..., r_n, ...\}, && \text{supervisory signals associated with the above actions}\\
&\forall t \in \mathbb{N}^*,\ \mathcal{A}m(a_t) = s_{t+1}, && \text{new world state after action effect}\\
&\mathcal{H}, && \text{a parameterized deep neural network to approximate } \mathcal{F}\\
&\mathcal{W} = (w_1, ..., w_m) \in \mathbb{R}^m, && \text{synaptic weights in and parameters of } \mathcal{H}\\
&b = (b_1, ..., b_k) \in \mathbb{R}^k, && \text{neural biases in and parameters of } \mathcal{H}\\
&\forall t \in \mathbb{N}^*,\ \mathcal{H}(s_t) = a_t, && \text{infer the most likely action under a given world state}\\
&\forall t \in \mathbb{N}^*,\ r_t = \mathcal{R}(s_t, a_t, s_{t+1}), && \text{reward received at time } t\\
&\forall t \in \mathbb{N}^*,\ \mathcal{G}_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}, && \text{weighted cumulative reward from begin to end}\\
&\gamma \in [0; 1], && \text{discount factor: models the influence on future experience}\\
&(\mathcal{W}_h, b_h) = \underset{(\mathcal{W},b)}{\arg\max}(\mathbb{E}[\mathcal{G}_1]), && \text{learning goal: maximizes expected cumulative reward}\\
&\forall s \in \mathcal{S},\ \mathcal{H}(s) \approx \mathcal{F}(s)), && \text{approximative inference}
\end{aligned}
\tag{2.8}
$$

In the above formulation, relations (e.g. $\mathcal{H}$, $\mathcal{F}$, $\mathcal{A}m$) are functionally interpreted and can be translated as well into a probabilistic formalism, by simply replacing functions by argmax of probabilities as done in 2.5. To better understand this formulation, let consider the training of an autonomous car. At the begining, the car is placed in its operational environment $\mathcal{S}$. Then, the state of the world s_1 is captured with a camera and the resulting image is passed to a deep neural network $\mathcal{H}$ (i.e. action policy). The latter infers the most likely action a_1 (i.e. motor commands to realize a_1), which is subsequently realized to yield a new world state s_2. Then, the supervisor (i.e. human) observes the consequences (i.e. s_2) of the action a_1 and determines a step reward r_1. The weighted cumulative reward $\mathcal{G}_1$ is updated and the system inference's error is computed based on reward $\mathcal{G}_1$. After the adjustment of its parameters, the network $\mathcal{H}$ infers a new action a_2 and the whole process is iterated untill a good performance is reached. However, note that for a much more smooth learning, the step rewards can be accumulated for a long time before the network parameters are adjusted.

Notice in the above scenario that the robot's (car) vision system is implicitly integrated in the deep neural network $\mathcal{H}$: this is the mainstream approach in applying reinforcement learning for robots. That is, the main task is not perception but generally action-based goals such as driving, manipulating, walking, etc. Perception is only implicit. However, as mentioned several times before, such an architecture is a bit too reactive to address higher-level reasoning-oriented and multi-level tasks such as cooking in human environments, which requires deliberative actions. Besides, relying on such an architecture for such complex tasks make it very unflexible (e.g. no reusability of components nor adaptation of the whole system). Finally, notice that though

learning supervision is flexible enough, it is quite difficult to figure out how to come out with a good amount of training data.

Deep Evolutionary Learning

Deep evolutionary learning [185, 174] is a very recent development and more similar to reinforcement learning. That is, it oughts to approximate under loose supervision a function $\mathcal{F}$. However, deep evolutionary learners are constructed in a manner very similar to how biological agents hardly evolve over time (i.e. darwinism). Unlike previous learning methodologies where the learners had a predefined topology (i.e. strong priors), an initial population $\mathcal{P}$ of hypotheses $\mathcal{H}$ is generated based on the system environment laws (often randomly). Over time, the population evolves through rejection $\mathcal{R}$ and selection $\mathcal{S}$ of individual hypotheses based on their survival values v (i.e. supervisory signal) on the one hand, and on the other hand through production of new individuals by combination (i.e. crossover) $\mathcal{C}$ or mutation $\mathcal{M}$ of selected individuals. The system stops learning when the population becomes stable. Note additionally that except the selection operator, all the other ones are random to a great extend.

It is was shown earlier in this chapter, that the evolution of the biological nervous systems could be reduced to neuro-plasticity, which is the adjustment of synaptic weights and may be neurons' behaviors (e.g. bias, activation). Since this conclusion was typically reached, while trying to come out with efficient computational theories of neural networks, an attempt to come back to a much more general theory of neural evolution may be an indicator of subtle weakness. First of all, review of the literuature indicates that the methology of deep evolutionary learning has only been validated on small-scale problems (e.g. CIFAR). Note that recently, several other learning approaches such as extrem machine learning [85] and polynomial regression [36] have tried unsucessfully to defeat deep learning solely by showing performances on small-scale problems. However, deep examination of these methologies reveal that they cannot be practical for interesting problems. As far as deep evolutionary learning is concerned, the literature further indicates that the formats of individual hypotheses $\mathcal{F}$ constituting the population $\mathcal{P}$ are well-known classical deep learning architectures (e.g. CNNs 2.4) built upon strong priors to address specific problems. In other words, deep evolutionary learners end up with a classical topology (e.g. no unusual connections). A truly neural darwinism, as this methodology may claim, would start with individual neurons, which would evolve to a population of complete deep nets and end with a best solution. Such an approach will unevitably undergo the Darwinism burden [41]: billions of years to yield interesting individuals. Furthermore, the learning complexity is more likely to degenerate due on the one hand to the considerable degree of randomness introduced in the learning process and on the other hand to the size of individual hypotheses (very big for interesting).

2.4 Classical Deep Learning Architectures

In the past sections, it has been sufficiently indicated that Deep Learning was about deriving theories to bring deep neural networks alive. It was particularly insisted on the fact that the topology of deep neural networks is central to success, since it encodes the greatest part of the model's innate knowledge (e.g. priors, logics). Moreover, though the network parameters encode acquired knowledge, this acquisition importantly depends on the network topology (i.e. innate knowledge). In this section, the classical topologies of deep learning models are presented. However, note that a deep learning model can be derived from the combination of multiple classical models.

Convolutional Neural Networks

Convolutional Neural Networks (CNN(s)) [140, 166, 6] are the most dominant and successful classical deep learning models, whose design was greatly inspired by the visual cortex of animals, making it therefore suitable for CV problems $\mathcal{F}$.

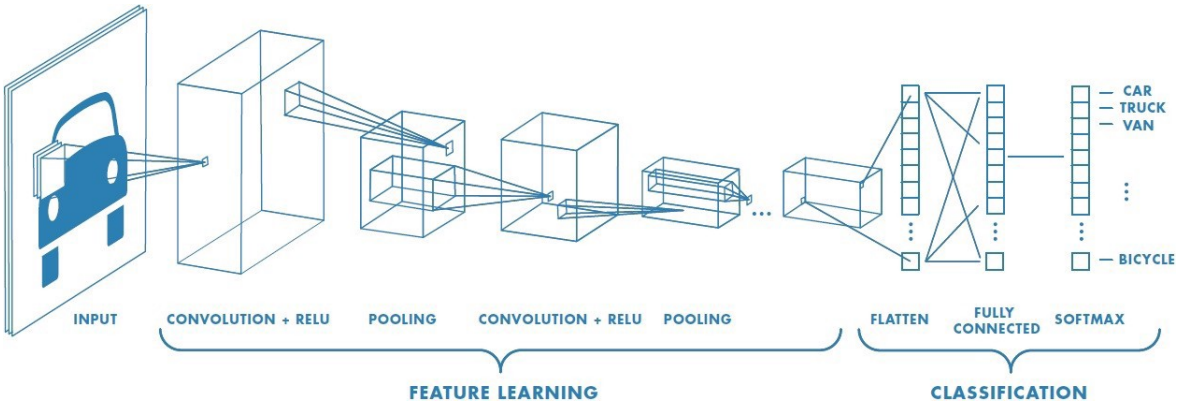


Figure 2.3 Architecture of an earlier typical CNN(s)-based model. The most right extremity of the model is a MLP(s) (fully connected layers), which itself is made up of perceptrons (small squarres). The figure further shows how complex a deep model is when compared to perceptrons and MLP(s) in size, structure and activation function. Today's deep models are far more complex. The input image is followed by a feature extraction (convolution+pooling), features which are summarized by a fully-connected-layer block and finally passed to the softmax layers for classification

source: [124]

CNN(s) are primarily supervised learners, nevertheless they can also be adapted and trained otherly. they operate on fix-length visual inputs such as images $\mathcal{I}$. To process variable-length sequences such as videos, the video's images are processed one after the other but without exloiting the sequentiality of those images. CNN(s) are made up of four major building blocks namely the input-layer block $\mathcal{IB}$, the convolution-layer block $\mathcal{CONVB}$, the fully-connected-layer

block $\mathcal{FCB}$ and the output-layer or softmax-layer block $\mathcal{SFTB}$, which can be composed in any other for any amount of times starting and ending respectively with the input-layer and output-layer blocks to yield a CNN(s): e.g. $CNN(s)(\mathcal{I}) = \mathcal{SFTB}(\mathcal{FCB}(\mathcal{CONVB}(\mathcal{IB}(\mathcal{I})))) = Pr(\mathcal{O}|\mathcal{I})$, where $\mathcal{O}$ (e.g. objects, categories, poses) is the network output and Pr a probability function. With a relatively massive convolution-layer block the previous example constitutes the classical organization of blocks within a CNN(s). Figure 2.3 illustrates the typical topology of a CNN(s). More explicitly, one distinguishes within a CNN(s) a:

Network Layer. Within a neural network (i.e. directed graph), a layer is a set of neurons (i.e. nodes) which are neither directly nor indirectly connected by any path (sequence of synapses), and most importantly operate at the same network depth (e.g. input, hidden, output).

Network Layer Block. This is a sequence of layer, in which each layer is connected to the next layer.

Input-Layer Block. This is the first block of layers within a neural network and whose neurons sense and encode the input image $\mathcal{I}$.

Convolution-Layer Block. This is the essence (i.e. $\approx 4\%$ of parameters, strong priors) and the most dense (i.e. $\approx 95\%$ of neurons) layer block of CNN(s)'s blocks. It captures important features of the biological visual cortex. That is, it produces a semantically very deep hierarchy of feature maps from the input image and constitutes the feature extractor of the whole model. Whereas its earlier layers output less semantic features (e.g. raw pixels, edges, corners, ..), the last layers come out with more semantic features (e.g. faces, categories, relations). Furthermore, CNN(s) are invariant to certain transformations (e.g. local deformations, global translations, scaling) of the input image $\mathcal{I}$ and can be easily taught (e.g. training data augmentation, constraint neural connections) to go beyond (e.g. rotation). Additonally, CNN(s) are not computationally complex and more likely to learn fastly and smoothly (i.e. strong priors, few parameters, highly parallelizable). Essentially, CNN(s) acquire these properties through a very deep alternated sequence of local processing operations on the input image called convolutions and dimension reduction operations called pooling. Convolution (i.e. local processing) denotes a particular type of synaptic wiring among the neurons of two layers, where the neurons of the first layers are only locally grouped to yield the input of a neuron of the second layer on the one hand. On the other hand, the synaptic weights of a groupement are shared by all other groupements. Pooling (i.e. dimension reduction) differs from convolution in the sense that groupements are usually mutually exclusive and the synaptic weights in groupements are all either set to 1 or 0 (i.e. not learnable) depending on the input signals. Whereas pooling layers mainly act as selectors (i.e. multiplexing based on e.g. maximum, minimum, average), convolution layers act as feature summarizers (e.g. lower-semantic features to higher-semantic feature). Moreover, the weights shared by a pooling or convolution layer's neural groupements is commonly referred to as the layer's kernel and the size of this groupment is commonly called the kernel size. The bigger the kernel, the more global the processing and the smaller the kernel, the more local the processing.

Fully-Connected-Layer Block. This block is made up of a few ($\approx$ 5% of neurons) fully connected layers (MLP(s)) and acts as a global feature summarizer ($\approx$ 96% of synaptic weights). In a MLP(s), given two consecutives layers, all the neurons of the first layers are connected to each neuron of the second layer. Since the convolution block (i.e. feature extractor) outputs a very high-dimension array of hierarchical local features from the input image, there is a need to reduce this dimension and may be to produce a much more robust (fully connected) global features: This is achieved by the fully-connected-layer block.

Softmax-Layer Block. This block is the last block of a CNN(s), usually made up of a single layer fully connected to the last layer of the previous block. The softmax layer outputs the conditional probability distribution $P(\mathcal{O}|\mathcal{I})$ of the network's output $\mathcal{O}$ given the network's input $\mathcal{I}$. This is achieved through logistic-like activation functions. However, note that this block is only useful for classification problems. For regression problems, where the regressed value is usually probabilistically expressed in terms of gaussian distributions, the mean and sometimes the standard deviation are outputted by a fully-connected-layer block or but rarely a convolution-layer block.

Reccurent Neural Networks

Reccurent neural network (RNN(s)) [34] is a direct answer to the problem of length-variable and sequential inputs encountered with CNN(s). This problem is solved by integrating an implicit memory mechanism in RNN(s) that allows them to track and leverage the sequentiality of the input images (e.g. videos). However, RNN(s) operate on sequences of image features $\mathcal{S} = \{F_1, ..., F_n\}$ not images $\mathcal{S} = \{\mathcal{I}_1, ..., \mathcal{I}_n\}$. That is, they do not endorse the feature extraction from images. For this reason, the feature extractor of CNN(s) is usually aggregated to RNN(s) to support the extraction of features, constituting then the complete classical topology of RNN(s).

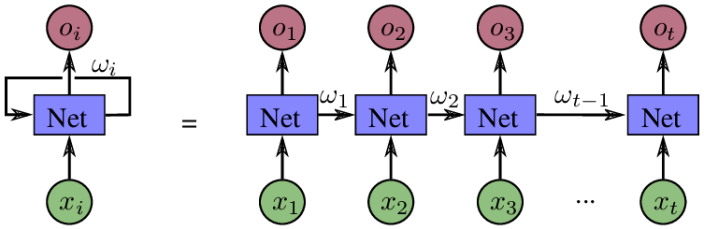


Figure 2.4 A classical architecture of RNN(s). *Net* is deep neural network, $\mathcal{M} = \omega$ and $\mathcal{I} = x$.

source: [192]

Notice that it is this ability, to allow deep networks to process sequential inputs, that makes RNN(s) a classical deep learning architecture. To track the sequentiality in the input, the

RNN(s) maintain a special memory vector $\mathcal{M}$ which incrementally integrates information about each incoming element F of the input sequence $\mathcal{S}$ in order to inform the processing of the input sequence's future elements about the past elements. Formally, $(\mathcal{O}_t, \mathcal{M}_{t+1}) = RNNs(F_t, \mathcal{M}_t)$, where $t \geq 0$ is the timestamp. The process is then iterated over the whole input sequence. Figure 2.4 illustrates the topology of a classical RNN(s).

It has been stated earlier in this chapter that MLP(s) were already universal approximators however for continuous functions on compact sets [82]. RNN(s) were proven to be Turing-complete by Siegelmann et al. [178]. Recently, a differentiable Turing neural machine was proposed by Graves et al. (Google's Project Deep Mind) [64] to approximate any Turing-complete machine. Differentiation is important because it allows the use of back-propagation to train Turing-complete RNN(s).

Auto-Encoders

Auto-Encoders (AE(s)) [127] can be seen as supervised and unsupervised learners at the same time. On the one hand, AE(s) are supervised learners in the sense that they learn the identity function $\mathcal{F} : \mathcal{S} \to \mathcal{S}$ and defined by $\mathcal{F}(\mathcal{I}) = \mathcal{I}, \forall \mathcal{I} \in \mathcal{S}$, where $\mathcal{S}$ is the set of inputs (e.g. Images). On the other hand, AE(s) can be regarded as unsupervised learners in the sense that they do not support any explitcit supervision (i.e. labels): This is the primarily interpretation of AE(s). Structurally, AEs are butterfly-shaped (i.e. inner layers shorter than outer layers) deep neural networks, whose layers are essentially fully connected layers. Notice that the core function of AE(s), which is computing an identity function, as well as their topology conveys them a very great ability: data compression. Figure 2.5 illustrates the compression mechanism of AE(s).

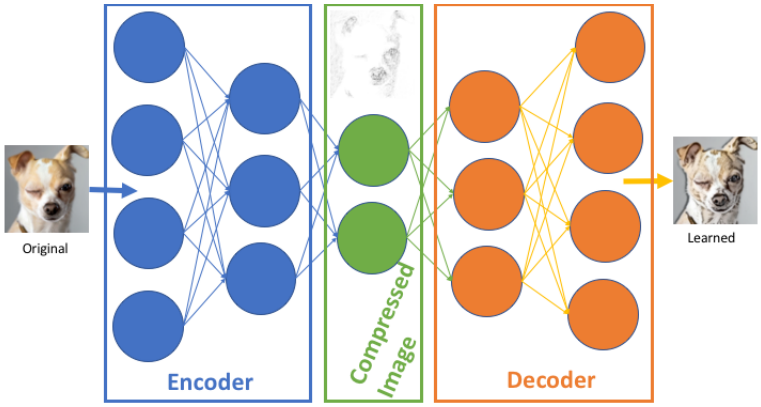


Figure 2.5 A classical architecture of AE(s). The blue deep net E_θ compresses the high-dimensional image $\mathcal{I}$ into a low-dimension vector $\mathcal{C}$ (green): encoder. And the orange deep net $\mathcal{D}_\phi$ decompresses the low-dimensional vector $\mathcal{C}$ into the high-dimensional vector $\mathcal{I}$: decoder.

source: [8]

Note that though AE(s) (i.e. $AEs(\mathcal{I}) = (\mathcal{D}_\phi \circ E_\theta)(\mathcal{I}) = \mathcal{I}$, $E_\theta(\mathcal{I}) = \mathcal{C}$, $\mathcal{D}_\phi(\mathcal{C}) = \mathcal{I}$, $|\mathcal{C}| << |\mathcal{I}|$) are primar1ily designed to extract essential features from the data, they enable simultaneously other interesting applications such as data denoising and data generation. In data denoising, a noisy image $\mathcal{I}_n$ is priorily derived from a clean image $\mathcal{I}_c$ and the AE(s) is then trained to learn a function that reproduces $\mathcal{I}_c$ from $\mathcal{I}_n$: $AEs(\mathcal{I}_n) = (\mathcal{D}_\phi \circ E_\theta)(\mathcal{I}_n) = \mathcal{I}_c$. In the case of data generation [45], the AE(s) is typically trying to learn the prior probability distribution $Pr(\mathcal{I}; \theta, \phi)$ of the data variable $\mathcal{I}$. That is, while the encoder E_θ learns the conditional probability distribution $Pr(\mathcal{C}|\mathcal{I}; \theta) = \mathcal{N}(0, 1)$ of the compression code $\mathcal{C}$ given the input image $\mathcal{I}$, the decoder $\mathcal{D}_\phi$ learns the posterior $Pr(\mathcal{I}|\mathcal{C}; \phi)$. However, note that such an application requires a special type of stochastic AE(s) known as variational AE(s) [45], developed otherly in the next section. To generate a very likely image $\mathcal{I}$, the latent variable $\mathcal{C}$ is sampled with respect to the normal distribution $\mathcal{N}(0, 1)$ and then passed to E_θ.

As mentioned in the previous sections [109], unsupervised learners such as AE(s) typically act as feature extractor in earlier stages of a supervised learning. However, since feature extraction in CNN(s) is perfomed in a discriminative manner (i.e. based on data and model's goal), AE(s) are usually outperformed since they solely rely on the data. Moreover, the training of AE(s) is not obvious since they are more likely to overfit and encounter back-propagation-related problems (e.g. gradient vanishing and exploding). In Computer Vision, convolution and pooling layers are usually used rather than fully connected layers in order to reduce the computational comlexity and avoid back-propagation-related problems.

Deep Belief Networks

Deep Belief Networks (DBN(s)) [26, 25] are the least used of all classical deep architectures and generalize AE(s). Fundamentally, DBN(s) are very deep multi-layer perceptrons (fully connected layers) which are built by consistently stacking a very big number of Restricted Boltzmann Machine(s) (RBM(s)). A RBM(s) is a two-layer perceptron with undirected synaptic links, meaning that the signal can travel in both directions along a synapse. RBM(s) are unsupervised generative models and can be seen as a type of fully stochastic splitted variational AE(s). As the input image $\mathcal{I}_0$ is forward propagated, the signal reaches the latent layer (i.e. innermost layer in AE(s)) as $Pr(\mathcal{O}|\mathcal{I}; \theta)$ (i.e. conditional probability distribution of the output signal $\mathcal{O}$ given the input signal $\mathcal{I}$ under $RBM(s)_\theta$). Then, a random output $\mathcal{O}_0$ is sampled (Gibb's sampling) with respect to $Pr(\mathcal{O}|\mathcal{I}; \theta)$ and backward propagated. The signal travels backward and reaches the input layer as $Pr(\mathcal{I}|\mathcal{O}; \theta)$ (i.e. conditional probability distribution of the input signal $\mathcal{O}$ given the output signal $\mathcal{I}$ under $RBM(s)_\theta$), then a random input $\mathcal{I}_1$ is gibb-sampled with respect to $Pr(\mathcal{I}|\mathcal{O}; \theta)$ and forward propagated again. After gibb-sampling a random output signal $\mathcal{O}_1$ from the resulting output's conditional probability distribution and repeating the above process for the k^{th} time, the training error is computed as $E =< \mathcal{I}_0, \mathcal{O}_0 > - < \mathcal{I}_k, \mathcal{O}_k >$, where $<, >$ designates the outer product. This error is then back-propagated for updating the model parameters: This learning rule is known as gradient-ascent-based contrastive-divergence-k, which fundamentally

does not differ from gradient-descent-based backpropagation spoken so far (i.e. only the abscence of explicit targets differs). Ideally, k should approximates infinity (i.e. $k \approx \infty$); However, it has been empirically shown that the learning process already easily converges for $k \geq 1$. To ease the sampling process, input and output neurons conventionally hold binary values (i.e. $\in \{0, 1\}$): Notice that RBM(s) actually generalize variational AE(s). Building a RBM(s) with different-size layers gives it additionally the ability to act as a compressor, where the shortest layer holds the compressed data (i.e. code). Furthermore, like AE(s), RBM(s) can be unsupervisedly trained, acting as parameters initializer, and then trained supervisedly to learn an end-goal-oriented function. Unfortunately, the binarization of input and output neurons of RBM(s) make their outermost layers intrinsically high-dimensional and causing them to become impratical when applied on high-dimensional data such as images (i.e. curse of dimensionality). Moreover, their shallowness speaks against their computational power (i.e. only 2 layers), compared to other deep models discussed so far. DBN(s) address this second issue by significantly increasing the number of layers.

Given a N-layer DBN(s), the layers i and $i + 1$ constitute a RBM(s), where i goes from 1 to $N - 1$. Since the training of DBN(s) is exposed to very big issues such as overfitting and back-propagation-related problems, it starts with the training of distinct RBM(s) with the gradient-ascent-based contrastive divergence-k and ends with a fine-tuning training of the whole DBN(s) with the gradient-descent-based backpropagation learning rule. Figure 2.6 illustrates a typical DBN(s) as a stack of RBM(s).

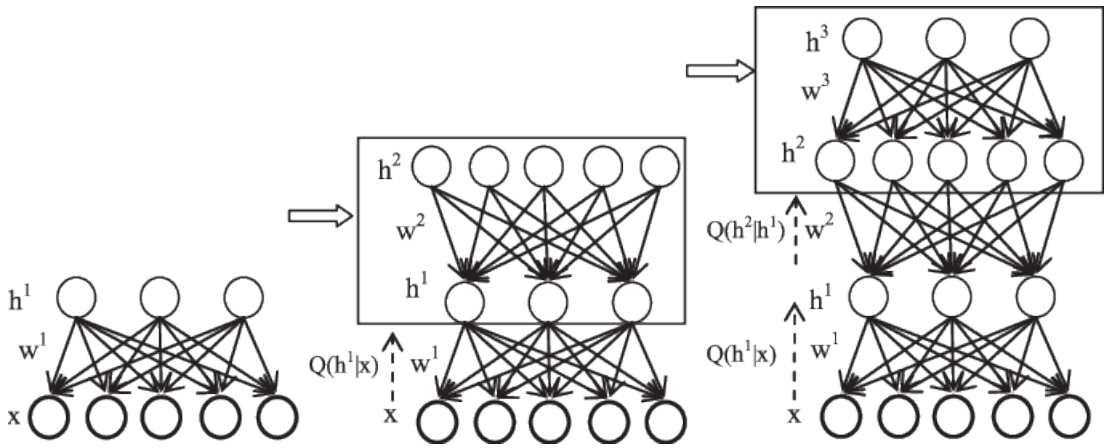


Figure 2.6 A classical architecture of DBN(s) at the most right side of the image. Building blocks such as RBM(s) are progressively stacked from the the most left side of the image to the most right side in order to yield deeper DBN(s). In the most left RBM(s), x designates the randomly sampled input, h is the randomly sampled output, Q is the conditional probability distribution of the output given the input and w are the synaptic weights.

source: [216]

DBN(s) can be regarded both structurally as well as functionally as deep RBM(s). Consequently, they can be applied likewise. However, unlike RBM(s), DBN(s) act deeply due to their deep fully connected layers. Unfortunately, this depth makes DBN(s) computationally very complex and besides worse than RBM(s) as far as the curse of dimensionality is concerned. For these reasons, DBN(s) have been receiving only very few attention from the CV community.

2.5 State of the Art

So far, it has been shown why Deep Learning is necessary, what it is able of, how to build deep learning models and how to train them. In this section, remarkable achievements in Computer Vision based on Deep Learning are reviewed. Since this thesis aims at a DL-based model that can provide deep semantics (i.e. scene graph) of robot scenes and for the sake of supporting humanoid robots on complex manipulation tasks, this section has been splitted into four parts based on the structure of scene graphs such as defined in the introductory chapter 1.2.2.3.

Image Tagging

As defined in the introductory chapter, image tagging [57] consists in globally analyzing a given image and then coming out with a summary label (e.g. color, name, material, shape) quite often also called scene gist [138]. There are hundreds of deep models around image tagging in the literature, however all of them can be reduced to a few established models, tested and approved on challenging benchmark datasets such as ImageNet[2012-2017] [92], MS COCO [115] and PASCAL VOC [53].

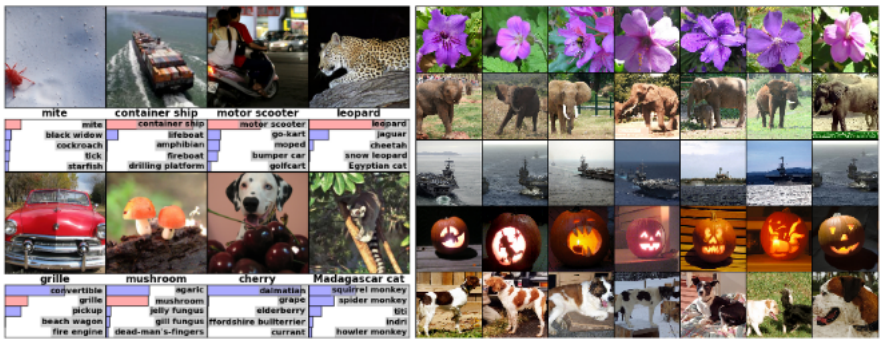


Figure 2.7 The right image shows typical images of the ImageNet dataset, while the left image illustrates the classification of some of those images: the red bar indicates the actual image tag and the length of a bar is proportional to the propability (score or confidence) of the associated predicted image tag.

source: [102]

As illustrated in figure 2.7, ImageNet is an annotated set of more than 14 millions of images with high entropy and containing more than 20.000 different labels. Note that the ImageNet dataset is progressily augmented over time and this is why a year is usually associated with the dataset name (e.g. ImageNet2012).

In 2012, a classical CNN(s) called AlexNet [103] was proposed for image tagging and won the challenge on ImageNet[2011-2012] with an error rate of 15.3%. In 2013, a modified version of AlexNet known as ZFNet [220, 219] and derived majoritarily from tuning the hyper-parameters of AlexNet, yielded a tagging improvement, lowering the error rate to 14.8% on ImageNet2012. In 2014, Google [189] introduced a new concept known as inception, which is about going widely (i.e. parallel computational multi-branches) rather than deeply (i.e. sequential stacked computational branches) while designing the CNN(s), in order to increase the model's computation power (i.e. parallel multi-way search) while preventing depth-related problems such as overfitting, gradient vanishing and exploding, as well as fixed-network-configuration-related problems such as failure to handle scale-variable inputs. The resulting architecture is commonly called GoogleNet (a.k.a. Inception) and showed an error rate of 6.7% on ImageNet2014, that is near the human performance, actually known as 5.1%. Besides GoogleNet in 2014, was VGGNet [179], just deeper with smaller convolution kernels and twice more parameterized than AlexNet, with a closer performance of 7.3%, which is also widely applied as building block in deep models. Notice that while GoogleNet searches more deeply and widely than AlexNet, VGGNet just does it deeply. In 2015, a new architecture called ResNet [78] has dominated the field of image tagging and showed a performance of 3.57% on the ImageNet2015. The key contribution of ResNet is their residual connections (a.k.a. shortcuts) between non-consecutive layers that allows them go very deeply in the computation while preventing depth-related problems. Combining the concepts of inception (e.g. invariance to input scale) and residual connections, namely Inception-ResNet [188], have led afterwards to a new record showing an error rate of 3.05% on ImageNet2012. In 2016, a slightly modified variant of ResNet [213] called ResNext was designed that showed an error rate of 3.03% on ImageNet2016. ResNext introduced the concept of cardinality, which consists in splitting a residual block (i.e. portion of the deep net which is shortcut by a residual connection) into several parallel and structurally identical residual blocks allowing to gain the advantage of inception as well as residual connections. Notice that ResNext is merely an extension of Inception-ResNet with more expressive parallel computational branches. Finally, SE-ResNet [83], which stands for Exitation and Squeezed ResNet, is another variant of ResNet proposed in 2017 that won the ImageNet2017 challenge for having shown an error rate of 2.25%. A SE-ResNet block looks like a normal ResNet block except that the residual block undergoes a further cleaning operation that enforces its discriminative power. This cleaning operation is achieved in two steps: the first step known as squeezing aims at summarizing the $[W \times H \times C]$-dimensional residual block RB_0 into a $[C]$-dimensional residual block RB_1 by applying a max-reduce operation known as global max pooling along the C-dimension of the residual block RB_0. Then, the residual block RB_1 is applied a non-linear transformation that allows a flexible interaction among the C different dimensions in

RB_1 and projects it onto a lower dimensional (i.e. $C' \leq C$) space in order to force the retention of essentially relevant information per dimension (i.e. bottleneck). The C'-dimensional residual block RB_2 is subsequently projected back onto the $[0;1]^C$ space as RB_3, representing the intrinsic discrimination factor of each of the C dimensions in RB_0. Finally, the retention of the essential from each information unit in RB_0 along the C-dimension yields a new much more discriminative $[W \times H \times C]$-dimensional residual block RB such that $RB[i,j,k] = RB_i[i,j,k] \times RB_c[k]$.

Summarizingly, this brief review of the state-of-the-art deep models for image tagging reveals that supervised learning and convolutional neural networks (CNN(s)) have proven to be successful in this task. Furthermore, special building blocks have been designed to improve the performance of these models over time. However, a closer look at these building blocks shows that they fundamentally rely on the concept of residual connections (i.e. ResNet). Figure 2.8 illustrates ResNet-based building blocks of deep models for image tagging.

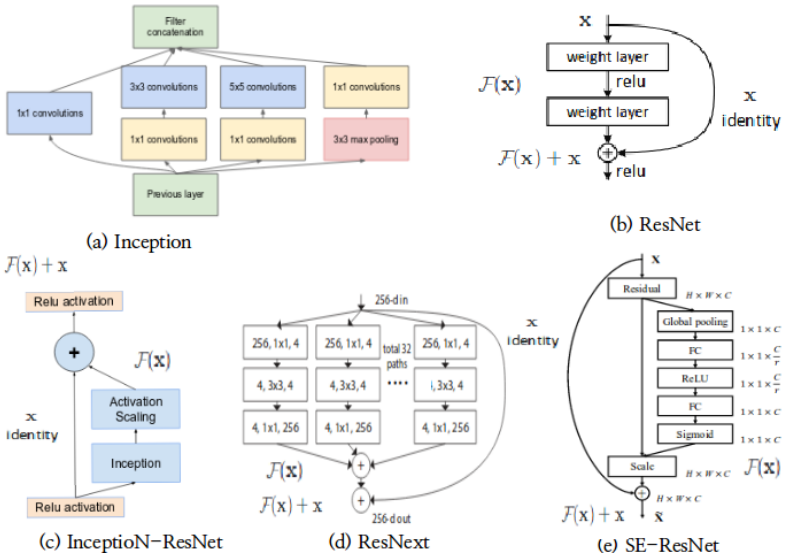


Figure 2.8 ResNet-based building blocks of deep models for image tagging. (a) shows the inception building block incorporated in GoogleNet and how it goes widely with parallel branches rather than deeply. (b) shows the fundamental ResNet building block with the associated residual block as $\mathcal{F}(x)$ and the residual connection as Identity x, which are then combined later as $\mathcal{F}(x) + x$. This residual connection is then illustrated on other ResNet-based building blocks (c), (d) and (e). (e) also illustrates the squeeze (global pooling) and the exitation ($Scale \circ Sigmoid \circ Fc \circ Relu \circ Fc(x)$) operations in the SE-ResNet. The leftmost branch (convolution-1D) in the inception building block (a) can be regarded as a loose residual connection.

source: [189, 78, 188, 213, 83]

Object Detection: Localization and Tagging

Notice that image classification as described above is merely about providing global information (i.e. tag) about the image. Moreover, the input images are expected to convey a global information: They are quite often centered around an object (e.g. cat, ship, glass). This is the reason why ImageNet usually contains small images of size 228×228. However, in complex scenes such as extremly dynamic and cluttered human environments, a household robot would need fine-grained information about each region of the scene in order to effectively interact with it. However, note that not all regions of the scene are relevant. A region is quite often relevant if it contains an object of interest, which itself depends on the system goals. A region containing food or kitchen stuffs is very likely to be interesting for a kitchen robot. A step to achieve this effective interaction with the scene consists in acquiring large (e.g. 640×640) images of the scene, identifying the regions of interest in the scene and tagging them as shown in the previous section: this localization followed by a tagging operation is called object detection [190]. Localization is the key component of object detectors. Figure 2.9 illustrates an object detection where an object is localized through a bounding-box.

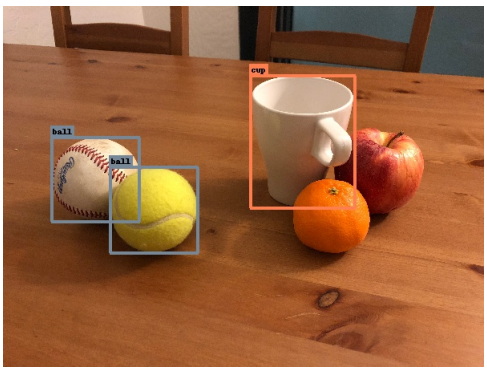


Figure 2.9 Detection of a cup and tennis balls on a table. The orange as well as the apple are misdetected either because they are not interesting or due to the detector weakness.

source: [61]

There are three major object localizers in the literature namely You Only Look Once (YOLO), Single Shot MultiBox Detection (SSD) and Faster-RCNN. The fastest and least accurate of them is YOLO. Given an input image $\mathcal{I}$, YOLO [155] splits $\mathcal{I}$ into $S \times S$ regular regions. Each region $\mathcal{R}_i$ is succeptible to be the center of an object $\mathcal{O}_k$ and each rectangle $\mathcal{B}_l$ centered at $\mathcal{R}_i$ is succeptible to be the bounding box of $\mathcal{O}_k$. However, the number of object candidates centered at $\mathcal{R}_i$ is limited to $B = 2$. Furthermore, YOLO does not allow two objects $\mathcal{O}_1$ and $\mathcal{O}_2$ of different classes to share the same center $\mathcal{R}$. The confidence of this localization is defined by the product of the probability that $\mathcal{O}_i$ is an object and the expected fraction of the union area, between the predicted box $\mathcal{B}_l$ and the expected box bounding $\mathcal{B}_e$ of the object $\mathcal{O}_i$, common to $\mathcal{B}_e$ and $\mathcal{B}_l$: formally $Pr(\mathcal{O}_i) \times IOU(\mathcal{B}_l)$. Notice that this confidence is maximal (i.e. 1) iff $Pr(\mathcal{O}_i)$ is 1 (i.e. certain of an object) and $IOU(\mathcal{B}_l)$ is 1 (i.e. perfect alignment between $\mathcal{B}_e$ and $\mathcal{B}_l$), where IOU

designates the intersection-over-union operator. YOLO regresses a $4D$-vectors, each of them holding the coordinates of a bounding box, and is essentially a CNN(s) inspired by GoogleNet to support classification. Though YOLO is extremly fast, it is useless for cluttered environments such as kitchen with small objects due to the constraints of candidate number and non-sharing of center between objects of different classes. A direct answer to these limitations of YOLO is SSD. Unlike YOLO, SSD [116] does not go for fixed-size regions in order to compute bounding boxes, but rather discretizes the space of bounding boxes ($\mathbb{R}^4$) into a finite set of various-size bounding boxes called anchor boxes. Then, a multitude of anchor boxes are densely spread all over the image and the confidence of each bounding box is computed through a classification. Since the bounding box associated with an object may not fit it very well, an adjustment is regressed from the local features enclosed by the bounding box. SSD is built upon VGGNet, however with an adaptation of the last layers of the convolutional block to maintain the feature maps at different scales: this enforces scale-invariance. Faster R-CNN [156] runs in a similar manner as SSD, however outperforms SSD primarily thank to its powerful ResNet-based feature extractor (i.e. basenet). Meanwhile, it significantly slows down more than SSD as the number of bounding box proposals increases. While the generation of bounding box proposals in SSD is wired and performed parallely (i.e. fast, highly parameterized, less accurate), Faster R-CNN handles this operation incrementally (i.e. slow, parameters sharing, more accurate).

Summarizingly, this review reveals the impact of CNN(s) notably ResNet-based deep nets and Region Proposal Network(s) (RPN(s)) on the detection of objects on images. The schemes of these object detectors are demonstrated in figure 2.10.

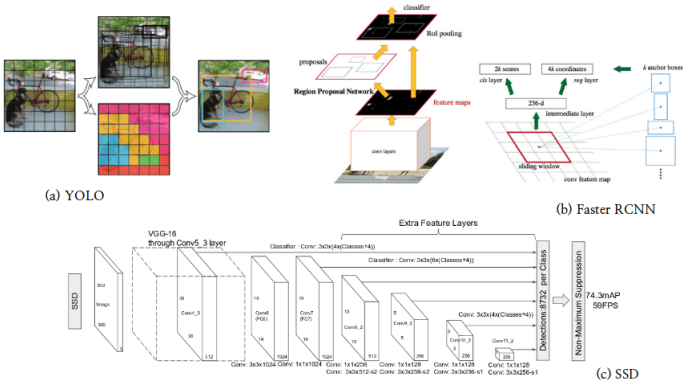


Figure 2.10 (a) is a brief demonstration of the YOLO pipeline (Grid+classification). (c) is a SSD with its VGGNet-based feature extractor, different-scale feature maps (scale-invariance) and the 8732 classifiers and regressors, each of them responsible for a particular bounding box (anchor box) at a particular position of the images. (b) is the Faster R-CNN with its ResNet-based feature extraction, RPN(s) for proposing k bounding boxes, pooling (RoI Pooling) the local features enclosed in each bounding box as a 256-D vector, then passing it to a classifier to outpout a $2k$-D vector (k boxes(class+confidence)) and to a regressor to output a $4k$-D vector (k boxes($4D$)).

source: [155, 116, 156]

Instance Segmentation

Though object detection enables a fine-grained tagging of images, the localization of interesting objects is still coarse for a primitive agent such as a robot to properly interact (e.g. grasping, bounding box misses object morphology) with them or understand how those object are spatially related to each other (e.g. many objects considerably share the same box). Instance segmentation [175] consists in detecting interesting objects in the scene as well as the exact portions of the scene that belong to each of them. Technically, this consists in associating each scene image pixel to a single scene object.

Though the literature suggests considerable novel work in the field of image segmentation based on Deep Learning, the majority of those works focuses on a different but closely related and less constrained problem known as the semantic segmentation (i.e. association scene image pixel to oject class(spoon)). That is, though semantic segmentation indicates which pixels are part of a spoon, it does not indicate which pixels make up a spoon. Note however that all this literature aims at solving some intrinsic problem related to image segmentation. Since 2017, Mask R-CNN [75], built upon the most accurate object detector Faster R-CNN, has established as the farmost accurate instance segmenter on top of DeepLab [35], SegNet [10], etc. After the bounding box proposals from Faster R-CNN, Mask R-CNN pools the features enclosed by each bounding box from the feature maps and from these features, produces based majoritarily on convolution a binary mask indicating the pixels in the bounding box which are part of the target object. The full architecture of Mask R-CNN is illustrated in figure 2.11.

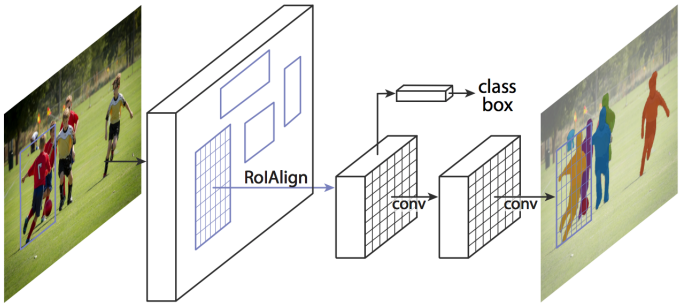


Figure 2.11 A typical topology of Mask R-CNN with its feature pooling mechanism (RoIAlign) and its massively convolution-based instance segmentation.

source: [75]

Object 6D-Pose

Though instance segmentation conveys relevant information (e.g. object morphology, space occupancy, $2D$-image position) for robot manipulation of objects, it remains insufficient. In practice, the robot would also need to know how far from it the object is, known as object depth lobject detection on table and lost during scene image acquisition through certain sensor models such

as RGB-cameras. Furthermore, it is also required the orientation of the object in the space. Providing a $3D$-position as well as $3D$-orientation of an object leads to its $6D$-pose [80].

Object 6D-pose based on Deep Learning is still embryonic. While the ones continue to rely on $3D$-CAD models of objects such as PoseCNN [210], the others still operate on isolated image patches avoiding fundamental issues such as multi-instance objects, occlusion and clutter in the scene. To the best knowledge of the literature, the most recent and robust approach is known as LieNet (a.k.a. Deep-6DPose) [44]. Essentially, LieNet extends Mask R-CNN with an additional pose branch to regress the $4D$-pose (i.e. depth+orientation) of each object. It regresses a $4D$-pose rather than a $6D$-pose because the other object position's coordinates (i.e. x,y) are directly inferred from the object's bounding box computed by Mask R-CNN. However, while some approaches interpret the object's $3D$-orientation either as $3D$-euler angles or $4D$-quaternions, LieNet regresses it via the Lie Algebra representation. As shown in figure 2.12, the feature extraction in LieNet is massively convolutional since it is built upon Mask R-CNN and the pose regression branch is massively made up of fully connected layers, due to the complexity of the pose estimation problem.

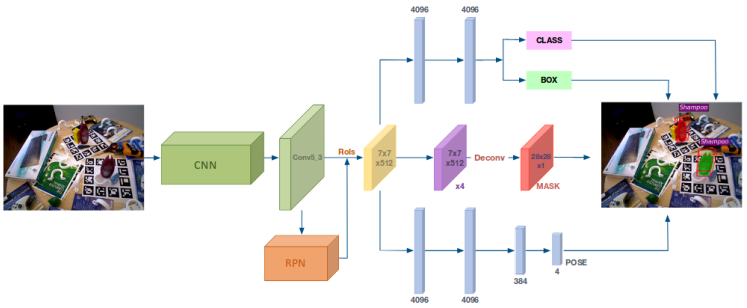


Figure 2.12 Architecture of LieNet

source: [44]

Spatial Relation

So far, it has been presented the state-of-the-art deep learning architectures that allows a manipulation robot to localize interesting objects in complex human-like scenes, to recognize and understand those objects (e.g. affordance, shape, material, etc) and to grasp them (e.g. $6D$-pose, segmentation mask). However, the robot still does not know how those objects are spatially related to each other, which is important for instance to know that if an object $\mathcal{O}_1$ is on top of another object $\mathcal{O}_2$, then to grasp $\mathcal{O}_2$, $\mathcal{O}_1$ must be graspped first and kept away. One could argue that the objects' segmentation masks as well as their poses are enough to derive these spatial relations, unfortunately this approach which seems theoretically plausible is naive in practice and even intrinsically misleading. Imagine two boxes of milk set near to each other on a table. Based on information related to each of those objects such as their poses or masks, the robot

will know that the first box is left to the second box. But if you rotate the image by $\frac{\pi}{2}$ (i.e. robot's camera or head inclination), the robot will claim that the second box is on top of the first box, which is wrong. Humans tackle these issues quite often by relying to a great extends on background or contextual rather than foreground information. This human inference is further enforced by naive physics which is itself acquired through learning over time.

In the CV community, understanding how scene objects interact with each other was of almost no interest till the famous problem of Visual Question Answering (VQA), which requires the system to answer questions about a given image and consequently to understand the relations among objects. Though there have been plenty deep learning frameworks aiming at expressing relations among scene objects, only very few of them have proposed an explicit logical mechanism for. Whereas explicit means that the proposed mechanism can be used as plugin in specific deep models, logical refers to the mechanism ability to capture the general algebra of relations. Particularly interesting among those mechanisms is the simple relational network $\mathcal{R}$ proposed by Google in their DeepMind's research project in 2017 [170]. $\mathcal{R}$ accepts a set of N objects $\mathcal{L} = \{\mathcal{O}_1, ..., \mathcal{O}_N\}$ ($\mathcal{O}_i \in \mathbb{R}^K$), certainly extracted from an image (e.g. RPN(s)). Notice that a set (i.e. $\{\bullet\}$) as input does not impose any constraint on the order of objects. Then, a binary relational operator $\mathcal{G}$ is applied on each of all the N^2 possible pairs of objects $(\mathcal{O}_i, \mathcal{O}_j)$ to discover the directed relation between both of them given a relation category Q also called the question (e.g. size, direction). This serialization of inputs empowers $\mathcal{G}$ both on its accuracy (e.g. parameters reduction and sharing) and practicability (e.g. no memory overflow). Afterwards, a summarization mechanism is proposed in case one would like to derive higher-order information. This summarization is achieved through the computation of a special summary vector $\mathcal{S}$ defined by $\mathcal{S} = \mathcal{F}(\sum_{i,j}^{K,K} \mathcal{G}(\mathcal{O}_i, \mathcal{O}_j, Q))$. Once again, summation allows to consider all pairwise relations however without imposing an order on how they should be inputted. This summation is followed by a non linear transformation $\mathcal{F}$ to allow a better interaction among pairwise relations. Since MLP(s) are universal approximators, Google-DeepMind suggests to consider both $\mathcal{G}$ and $\mathcal{F}$ as MLP(s). Figure 2.13 demonstrates $\mathcal{R}$.

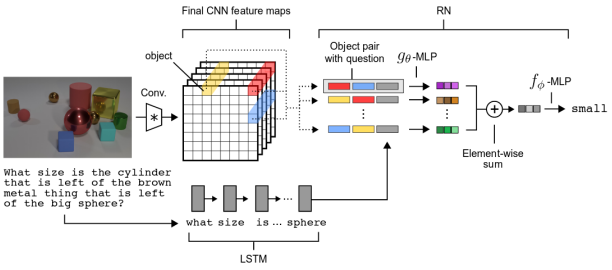
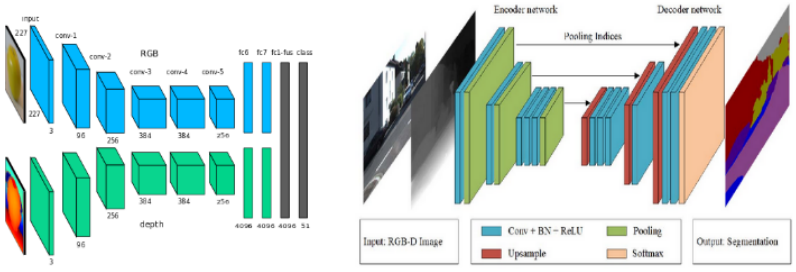


Figure 2.13 Simple Relational Network. Conv is the feature extractor (CNN(s)+RPN(s)), LSTM is a RNN(s) that converts the higher-order question Q (What is...?) from natural language to a vector, $\mathcal{G} = g_\theta$, $\mathcal{F} = f_\phi$ and $\mathcal{S} = small$.

source: [170]

Scene Depth Integration

Finally, this review of the most influential DL-based vision models closes with a discussion on how to efficiently integrate scene depth into such models. Notice that such models have proven on RGB-images without any cue on scene depth. The exploitation of depth information is left back to roboticists. To leverage scene depth in DL-based robot vision systems, there have been 3 major approaches. While some architectures combine the $[W \times H \times 1]$-depth image together with the $[W \times H \times 3]$-color image to yield a $[W \times H \times 4]$-unified input map [112], others maintain two parallel usually identical processing pipelines for both the depth and color image, then merge the results later [74]. Whereas the first approaches may prevent the full exploitation of learned parameters from pre-trained models and moreover cause some external and internal covariate shifts [90] (i.e. depth domain is unstable), the second approach systematically scales up the computational complexity of the resulting model by an important factor (e.g. 2). To avoid covariate shifts, depth maps are sometimes converted to color images notably as normal images (i.e. depth domain is stable) before integration into the models [51]. Normal images colorfully encode the scene depth but relatively rather than absolutely and primarily inform about the shapes of scene objects. These approaches are illustrated by figure 2.14.



(a) Multi−processing pipeline with colorful depth image.

(b) Unified input map with gray depth image

Figure 2.14 Approaches for integrating scene depth into deep models.

source: [51, 112]

2.6 Conclusion

In this chapter, the fundamentals as well as advanced theories of Deep Learning were presented, and followed by a review of top DL-based models for addressing the key problems of CV. It came out that Deep Learning aims at emulating the biological nervous system at a very fine-grained precision in order to tackle extremly complex and particularly perception-related problems, well known accomplished by biological agents. Like biological agents, DL-based models evolve over

time through learning from their environments, where an explicit external learning supervision has been noticed to be of a considerable importance in practice. Furthermore, though there exists a multitude of classical DL-based architectures, CNN(s) and RNN(s) appear to be crucial while addressing vision-related problems. Finally, an intensive review of the state-of-the-art DL-based vison models revealed that Mask R-CNN has by far outperformed on very large-scale datasets (e.g. ImageNet, MS COCO) in the domains of image classification, object detection, localization and segmentation. LieNet goes beyond by extending Mask-RCNN with an additional branch for computing 6D-poses of scene objects. As far as spatial relations among scene objects are concerned, Google-DeepMind has proposed a generic explicit logical mechanism for estimating direct and indirect relations among scene objects. Besides, it has been found advantageous to integrate the scene depth into deep models by combining depth-related maps together with color maps into a single unified input map rather than going for a model with multiple processing pipelines, each of which responsible for a particular scene map either color or depth.

Chapter 3

Dataset

3.1 Introduction

In the introductory chapter, a powerful formalism, called scene graph, that can allow a humanoid robot to perform complex human-scale manipulation activities such as cooking in cluttered, noisy and dynamic environments such as kitchens, was exposed. Basically, a scene graph is intended to provide a nearly complete and structured semantics of a robot scene from scene images. Furthermore, it was shown that a scene image analysis was more advantageous than a scene video analysis. Finally, DL-based models were found powerful enough to be able to accomplish such a perceptual task.

In the second chapter, the theories of Deep Learning were presented, where CNN(s) as well RNN(s) were shown to be crucial in computer vision. Whereas CNN(s) are experts in analyzing individual images, RNN(s) leverage the sequentiality of those individual images. However, as just mentioned in the previous paragraph, such an exploitation of the sequentiality of scene images was shown less advantageous. Moreover, given that these models are supervisedly trained and require big rich annotated data for their training, it would be globally more costly to leverage the sequentiality of images since it would require a training on image sequences (i.e. videos), which though extremly big are of very lower entropy compared to big sets of rich images, whose entropies are sgnificantly important. As shown in the introductory chapter, this exploitation of sequentiality in the images could be rather performed later at a symbolic level.

Since this thesis aims at developing a DL-based vision system that can produce scene-graph-oriented semantics of humanoid robot scenes in order to effectively and efficiently support these robots in accomplishing complex manipulation tasks in home environments, there is an unconditional need to evaluate such a model. In this chapter, a dataset, that can allow to teach, validate and test a deep model that accepts a scene image as input and return the corresponding scene graph as output, is built.

3.2 Big Rich Annotated Data

While biological agents take tremendous amount of time (e.g. years) to evolve through learning, by undergoing a very long matching process of extremly various environment stimuli to action responses, artifical agents such as deep models aim at this goal within a very small amount of time (e.g. days). As a necessary consequence, all the stimuli-response pairs encoutered by biological agents over years as they evolved, are needed at once: This is the big rich annotated dataset. In this appelation and in the context of visual perception, *data* commonly refer to images, *big* stands for the astronomical quantity of those images, *annotated* refers to the association of the right action response to each image. Finally, *rich* refers to the variability of features in those annotated images and is technically called entropy.

While training a deep model on a small amount of data is most likely to lead to overfitting (i.e. no learning but retention), doing it with a huge amount of data but low entropy will cause the system to effectively learn but within a very restricted domain. A huge amount of high-entropy training data is hence a sign of good representativeness of the artificial agent's infinite complex operating domain [183, 111].

Given that this thesis seeks to develop a DL-based vision system for manipulation robots in home environments, there is a need for training, evaluation and test big rich annotated datasets. However, building such datasets for all manipulation tasks in all home environments with such a dense annotation as specified by scene graphs, is simply unrealistic. For this reason, the evaluation of the vision model proposed by this thesis has been limited to cooking activities in the EASE's kitchen designed in the IAI-Bremen's laboratory. Even with this restriction, it is still unclear how one could come out with a huge amount of high-entropy and densely annotated training and testing data: i.e. given a scene image, it is not an easy task to provide the segmentation mask as well as the 6D-pose of each interesting object. A prompt solution to this problem would be to exploit publicly available datasets online for Robotics. However, this is a great mistake since the practice ignores the embodiedness and situatedness of the robots. Furthermore, the scene graph exposed in the introductory chapter is so rich that it is almost impossible to see a publicly available big dataset with such a dense annotation. Another common technique to produce big dataset consists in building a small dataset, then augmenting it through some annotation-invariant transformations such as slight modification of image pixels' values, image translation, rotation and flipping. However, notice that this approach presents two severe limitations. Though such a data augmentation significantly increases the size of the initial dataset, it does not really change its entropy. Secondly, this practice was very efficient before because the data annotation (i.e. segmentation masks, objects) could easily be recovered after augmentation; However, annotation recovery becomes a bit tricky with augmentation-sensitive labels such as object pose and spatial relations among objects, object material, color and shape. This big data crisis, preventing very motivated and curious scientists to push the borders of

DL-based AI, has finally forced them to think quite otherly: Virtual-reality-, but as well as augmented-reality-based synthetic datasets seem very promising [40, 3]. Figure 3.1 illustrates the major limitations of some publicly available kitchen-like datasets for robot vision systems.

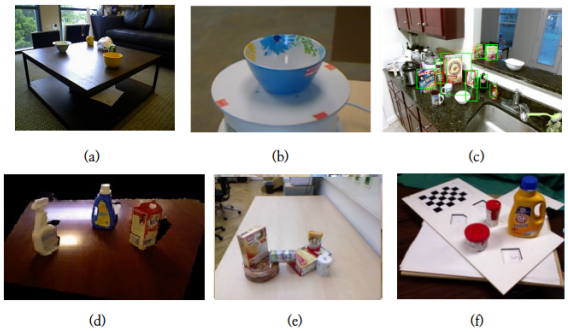


Figure 3.1 Limitations of publicly available kitchen-like datasets for robot vision systems. (a) very low entropy, no clutter, no occlusion, just for detections. (b) unsituatedness. (c) just detection. (d) no clutter, no oclusion, detection+pose. (e) very restricted domains + very few samples (≈ 111). (f) landmarked scene

source: [55]

3.3 Virtual Reality

In this section, the principles of virtual reality are elucidated and how virtual reality may contribute to the automatic generation of big rich annotated Data. Then, available frameworks for implementing such a data generator are presented.

3.3.1 Principle

Real world virtualization is quite often goal-oriented since the whole real world in its entireness cannot be described. Virtualization provides from a certain perspective a computer representation of the world that allows a goal (e.g. agent training, gaming) to be reached. After computationally modeling the real world with all its components and actors, the model is then simulated to achieve the intended goal. During the simulation, the real world usually interacts with the virtual world as it is the case in video games or pilot training, where humans control virtual actors by the means of input-output mechanisms such as hard joysticks or soft plugins (i.e. software).

Following the above principles, modeling the world for the sake of visual scene understanding for robot manipulation tasks, would require for instance a modeling of the robot with all necessary

sensors and actuators, a modeling of the objects of interest, a modeling of the required Physics such as optics and newtonian mechanics and finally a modeling of the mechanisms of interaction with the real world.

The major advantage of virtual reality is that, the resulting virtual world can be totally and effortless accessed as long as needed. That is why the field has received enormous attention from experimentation in Science and system design, testing, training and maintainance in Engineering, video game industries and personal training in academic institutions. In this thesis, virtualization is leveraged for training DL-based robot vision systems [131].

3.3.2 Frameworks

Among the numereous existing virtual reality engines to virtualize and simulate the real world, Unreal Engine (UE) [169, 141, 145, 201] has shown itself one of the best, due obviously to its significant contributions in the field of virtual reality. For this reason, this thesis adopts Unreal Engine version 4.16 as the framework for virtualizing the robot world. However, a mechanism is still needed for interacting with external agents such as data collection programs. Fortunately, there exists such a plugin specially designed to allow computer vision systems to interact with UE simulations. This plugin is known as UnrealCV (version 3.10) for Unreal Computer Vision [150]; That is, the visual scene is virtual. Whereas development in UE is done in the programming language C++, UnrealCV's frontends are developed in Python and backends in C++. This connection between UE and UnrealCV is demonstrated in figure 3.2.

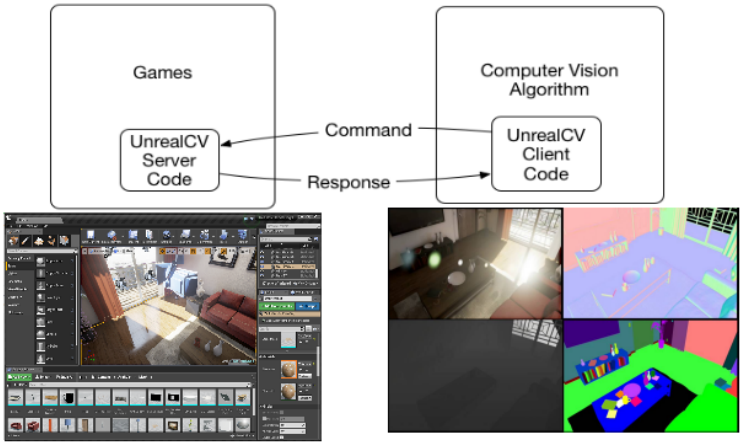


Figure 3.2 A connection from UnrealCV to a UE simulation to request scene images under multiple modes (color, depth, normal and instance segmentation mask).

source: [150]

3.4 Data Generation

This section describes the full process of the automatic dataset generation based on virtual reality.

3.4.1 Scene Graph

In the introductory chapter, a formalism, called scene graph, was defined for providing efficient and effective semantics of manipulation robots' scenes. The formalism was built upon the probabilistic decidable general description logic with concrete domains *P-ALCNHr+(D)*. Essentially, given an image of the scene, a scene graph shows all the interesting scene objects, their attributes and the spatial relationships with a degree of certainty. However, it was not clarified so far what attributes make up an object, what are their concrete values and how they can be spatially related to each other. While some of those information such as object category (i.e. Spoon) are purely revealed by the robot's operating domain (EASE's kitchen), others are purely derived from concrete robot manipulation scenari and human manipulation experience (i.e. object material, shape, spatial relations among objects).

Object Attributes

Object Category: this is the most fundamental property of an object, when performing tasks such as cooking, since it conveys information about the object's end utility. Its concrete domain $\mathcal{D}_{Ocat}$, obviously with regard to the EASE's kitchen, includes 23 values namely $\mathcal{D}_{Ocat}$={Cooktop, Tea, Juice, Plate, Mug, Bowl, Tray, Tomato, Ketchup, Salt, Muesli, Spoon, Spatula, Milk, Coffee, Knife, Cornflakes, Eggholder, Pancake, Cereal, Rice}.

Object Material: the material of an object is extremly relevant when it comes to understand how to deal with contacts with an objects. Would the robot put a steel cup inside the microwave oven, a plastic on top of the cooktop or cut some bread with a plastic knife? Knowing about the object material is very important in manipulation tasks. The concrete domains $\mathcal{D}_{Omat}$ of object materials is defined by $\mathcal{D}_{Omat}$={Plastic, Wood, Glass, Steel, Cartoon, Ceramic}.

Object Shape: this attribute plays a key role when it comes to know how to place an object on top of the others, how to place the other on top of or inside it, how to grasp it. A filiform object such as spoon cannot contain any other object and it is more difficult to place a plate on top of a bottle than on top of a bowl. Its concrete domain $\mathcal{D}_{Osha}$ is defined by $\mathcal{D}_{Osha}$={Cubic, Conical, Cylindrical, Filiform, Flat}.

Object Color: though object color is not directly relevant for the object manipulation, it typically acts as a contextual information for tracking. Imagine that the robot has 2 white glass mugs in front of it and pours some orange yogurt inside one of them, after fetching the spoon,

how will the robot localize the glass mug with yogurt? Certainly, through the mug's color. The concrete domain of object colors $\mathcal{D}_{Ocol}$ is defined by $\mathcal{D}_{Ocol}$={Red, Orange, Brown, Yellow, Green, Blue, White, Gray, Black, Violet, Pink}.

Object Openability: this is a binary value which indicates whether or not an object is openable or not, which is crucial while accessing the content of the object. Moreover, it also acts as a contextual information in the prediction of other properties. An openable cylindrical object is more likely to be a bottle of milk than a mug and prior to get the juice out of the bottle, unlike a mug of juice, the openable bottle must be opened. Its values are $\mathcal{D}_{Oopn}$={Openable, Non-Openable}.

Object Segmentation Mask: this is a generally important feature in Mobile Robotics. Essentially, it informs about the the object's space occupancy, roughly about the location and spatial interactions among objects. An object's mask is obtained by setting the values of all pixels belonging to the object to 1 and all other pixels' values to 0. Formally, its concrete domain $\mathcal{D}_{Omsk}$= $\{0,1\}^{(H,W)}$, the set of binary matrices of W columns and H rows, W and H being respectively the width and height of the input scene image.

Object 6D-pose: an object's $6D$-pose consisting of its $3D$-position and $3D$-orientation is essential for a robot to effiently grasp the object. Formally, its concrete domain is defined by $\mathcal{D}_{Opos}$= $[0,1]^6$. Notice that all the 6 dimensions of the pose have been normalized to $[0,1]$. Whereas the position's coordinates are normalized based on the size of the EASE's kitchen, the orientation's euler angles are reduced to $[0,2\pi]$ and then $[0,1]$.

Spatial Relations Among Objects: these are essential in understanding how scene objects interact with each other. Prior to use an object such as a bowl, its content must be eventually taken away. In principle, we distinguish almost 8 types of spatial relations namely $\mathcal{D}_{Orel}$={Left, Rigth, Over, Under, On, Front, Behind, In} between any pair of distinct objects in the scene. For N objects, the number of pairs being approximatedly $\mathcal{O}(N^2)$ leads directly to $\mathcal{O}(8 \times N^2)$ relations which become quickly intractable at every stage (i.e. annotation, training, evaluation) of the vision system development's pipeline. Based on many tricks such as the transitivity rule, this problem is addressed by defining an efficient format for describing spatial relationships which significantly reduces the complexity to $\mathcal{O}(N)$ in average. The following algorithm 2 illustrates this compression and demonstrates the completeness of the group of relations $\mathcal{D}^{min}_{Orel}$={on, in, left, front}.

Algorithm 2: Quasi-Lossless Compression of Spatial Relation Description

Data: set of N scene objects $\mathcal{S}_{obj} = \{O_1, ..., O_N\}$, set $\mathcal{R}_{Orel}$ of all possible pairwise spatial relations among objects of $\mathcal{S}_{obj}$, with $|\mathcal{R}_{Orel}|$=$\mathcal{O}(8 \times N^2)$, $\forall(O_i, r, O_j) \in \mathcal{R}_{Orel}$, $r \in \mathcal{D}_{Orel}$, $O_i, O_j \in \mathcal{S}_{obj}$
Result: minimal semantically quasi-complete set $\mathcal{R}^{min}_{Orel}$ of spatial relations among the N scene objects of $\mathcal{S}_{obj}$, with $|\mathcal{R}^{min}_{Orel}|$=$N$-1, $\forall(O_i, r, O_j) \in \mathcal{R}^{min}_{Orel}$, $r \in \mathcal{D}^{min}_{Orel}$, $O_i, O_j \in \mathcal{S}_{obj}$

1 $\mathcal{R}^{min}_{Orel} \leftarrow \mathcal{R}_{Orel}$; // initialize $\mathcal{R}^{min}_{Orel}$
2 $\forall(O_i, r) : (O_i, r, O_i) \in \mathcal{R}^{min}_{Orel} \implies \mathcal{R}^{min}_{Orel} \leftarrow \mathcal{R}^{min}_{Orel} \setminus \{(O_i, r, O_i)\}$; // no reflexivity

/* under- and over-relations not necessary, in- and on-relations enough */
3 $\forall(O_i, O_j, r) : (O_i, r, O_j) \in \mathcal{R}^{min}_{Orel} \wedge r \in \{over, under\} \implies \mathcal{R}^{min}_{Orel} \leftarrow \mathcal{R}^{min}_{Orel} \setminus \{(O_i, r, O_j)\}$
/* no redundance */
4 $\forall(O_i, O_j) : (O_i, right, O_j) \in \mathcal{R}^{min}_{Orel} \implies \mathcal{R}^{min}_{Orel} \leftarrow (\mathcal{R}^{min}_{Orel} \cup \{(O_j, left, O_i)\})) \setminus \{(O_i, right, O_j)\}$
5 $\forall(O_i, O_j) : (O_i, behind, O_j) \in \mathcal{R}^{min}_{Orel} \implies \mathcal{R}^{min}_{Orel} \leftarrow (\mathcal{R}^{min}_{Orel} \cup \{(O_j, front, O_i)\}) \setminus \{(O_i, behind, O_j)\}$
/* no shortcut, but transitivity */
6 $\mathcal{T} \leftarrow \emptyset$
7 $\forall(O_i, O_j, O_k, r) : (O_i, r, O_j) \in \mathcal{R}^{min}_{Orel} \wedge (O_j, r, O_k) \in \mathcal{R}^{min}_{Orel} \wedge r \in \{left, front\} \implies$
$\mathcal{T} \leftarrow \mathcal{T} \cup \{(O_i, r, O_k)\}$
8 $\forall(O_i, O_j, O_k, r, r', r") : (O_i, r', O_j) \in \mathcal{R}^{min}_{Orel} \wedge (O_j, r, O_k) \in \mathcal{R}^{min}_{Orel} \wedge (r', r", r) \in \{in, on\}^3 \implies$
$\mathcal{T} \leftarrow \mathcal{T} \cup \{(O_i, r", O_k)\}$
9 $\mathcal{R}^{min}_{Orel} \leftarrow \mathcal{R}^{min}_{Orel} \setminus \mathcal{T}$
/* no aerial left-, front-relations, $\vee^*$ denotes the logical exclusive or */
10 $\forall(O_i, O_j, O_k, r, r') : (O_i, r, O_j) \in \mathcal{R}^{min}_{Orel} \wedge r \in \{left, front\} \wedge r' \in \{in, on\} \wedge ((O_i, r', O_k) \in$
$\mathcal{R}^{min}_{Orel} \vee^* (O_j, r', O_k) \in \mathcal{R}^{min}_{Orel}) \implies \mathcal{R}^{min}_{Orel} \leftarrow \mathcal{R}^{min}_{Orel} \setminus \{(O_i, r, O_j)\}$
/* No cycle */
11 $\mathcal{T} \leftarrow \emptyset$
12 $\forall(O_i, O_j) : (O_i, left, O_j, O_k) \in \mathcal{R}^{min}_{Orel} \wedge (O_i, front, O_k) \in \mathcal{R}^{min}_{Orel} \wedge (O_j, front, O_k) \in \mathcal{R}^{min}_{Orel} \implies$
$\mathcal{T} \leftarrow \mathcal{T} \cup \{(O_i, front, O_k)\}$
13 $\forall(O_i, O_j, O_k) : (O_i, front, O_j) \in \mathcal{R}^{min}_{Orel} \wedge (O_i, front, O_k) \in \mathcal{R}^{min}_{Orel} \wedge (O_j, left, O_k) \in \mathcal{R}^{min}_{Orel},$
$\mathcal{T} \leftarrow \mathcal{T} \cup \{(O_i, front, O_j)\}$
14 $\forall(O_i, O_j, O_k, r) : (O_i, r, O_j) \in \mathcal{R}^{min}_{Orel} \wedge r \in \{in, on\} \wedge ((O_i, left, O_k) \in \mathcal{R}^{min}_{Orel} \vee (O_k, front, O_i) \in \mathcal{R}^{min}_{Orel})$
$\implies \mathcal{T} \leftarrow \mathcal{T} \cup \{(O_i, r, O_j)\}$
/* front is prioritary over left in case of double-relation in a pair of objects */
15 $\forall(O_i, O_j) : (O_i, front, O_j) \in \mathcal{R}^{min}_{Orel} \implies \mathcal{T} \leftarrow \mathcal{T} \cup \{(O_i, left, O_j), (O_j, left, O_i)\}$
16 $\mathcal{R}^{min}_{Orel} \leftarrow \mathcal{R}^{min}_{Orel} \setminus \mathcal{T}$
17 **return** $\mathcal{R}^{min}_{Orel}$

Given the complete graph of spatial relations among scene objects $\mathcal{R}_{Orel}$ of size $\mathcal{O}(8 \times N^2)$, this algorithm computes a minimal spanning tree $\mathcal{R}^{min}_{Orel}$ of size N-1 from $\mathcal{R}_{Orel}$. However, it prioritizes the front-relation over the left-relation between two objects. This priority for the front-relation can be explained by the fact that it better informs, in the sense that it tells the robot that it might not directly access an object. Furthermore, though it may seem to lead to information loss, it does not definitively, since the robot can still rely on object masks or $3D$-positions to recover the marginalized left-relation. However, such a recovery is not garanteed for all scene configurations: hence, the quasi-lossless rather than lossless compression. Figure 3.3 graphically illustrates the above compression scheme 2.

Additionally, since ANN(s) are not good enough at or require considerable effort for demonstrating higher-level symbolic reasoning as this strong compression scheme might demand, to achieve a good tradeoff between the deep model's prediction power and the required workload (e.g. annotations), the last two rules notably (g) and (h) (see figure 3.3) were relaxed while

collecting the dataset in the context of this thesis. This section ends with a concrete illustration, as demonstrated in figure 3.4, of a scene graph derived from a kitchen scene RGB(D)-image. On training, the robot vision system is taught to infer the scene graph S associated with a givien scene image I. While building the training dataset, S constitutes an annotation of I and $< I, S >$ is called an annotated scene image.

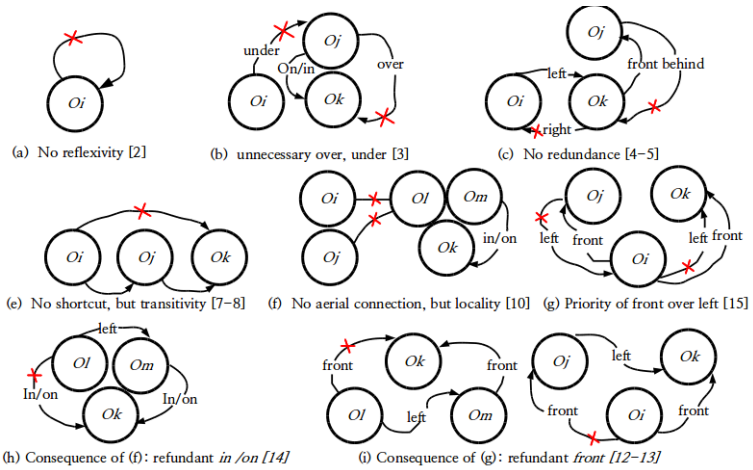


Figure 3.3 Rules for compressing a graph of $\mathcal{O}(8 \times N^2)$ spatial relations among N objects into a minimal graph of N spatial relation among those objects. Each rule is associated with the above algorithm's lines implementing it. The red crosses show destruction of useless relations through application of rules.

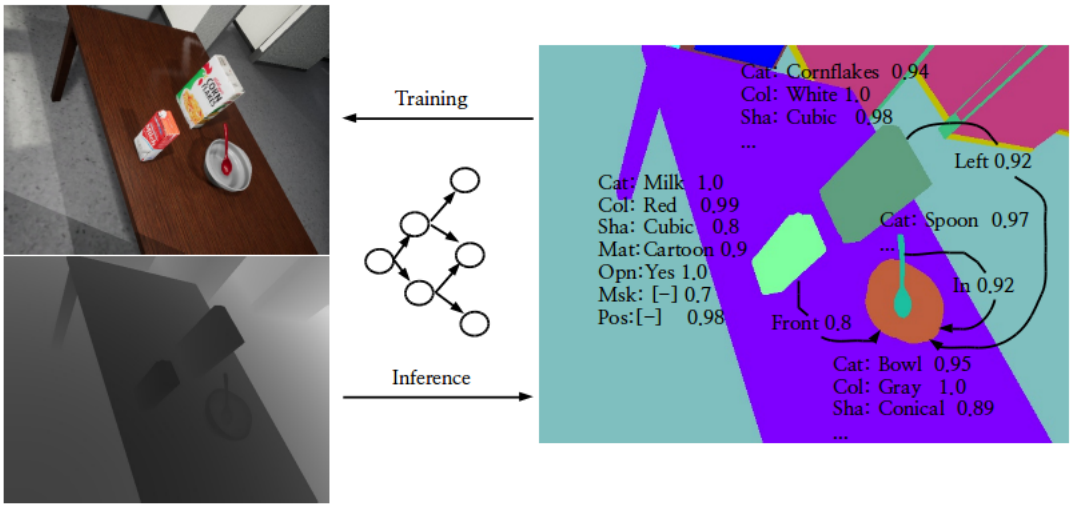


Figure 3.4 On the left side, an input scene RGB(D)-image and the associated scene graph on the right side.

3.4.2 Robot World Virtualization

This section exposes the virtualization of the robot world in UE, which obviously encompasses the EASE's kitchen, the robots as well as the interactions (i.e. I/O mechanisms) with the external real world. The virtualization of an object in UE basically follows two steps namely the definition of the object's physical properties and finally the definition of object's responses to stimuli either external (i.e. from real world) or internal (i.e. from virtual world) a.k.a. behaviors.

3.4.2.1 Kitchen Virtualization

Roughly speaking, the kitchen is made up of a building which itself consists of walls, ceilings, floors, doors, windows,lights, an equipment encompassing tables, fridges, sinks, ovens, washmachine and finally purely manipulable objects such as utensils (i.e. spoon, plate) and food items (i.e. Milk box, juice bottle).

Physical Properties

Object Mesh. This is the earliest stage of virtualization and consists in defining the topology and the size of the object. As far as the EASE's kitchen is concerned, the meshes of objects from walls to utensils were entirely covered in the past, precisely by [70] in the context of the EASE's subproject ROBCOG. Figure 3.5 demonstrates the meshing of the EASE's kitchen.

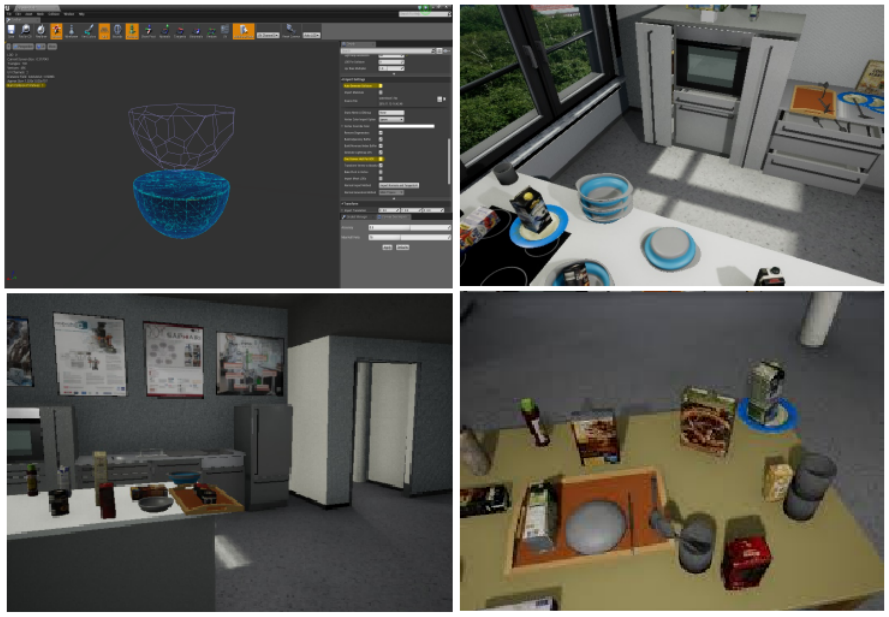


Figure 3.5 Meshing of the EASE's kitchen in UE 4.16.

Object Material & Texture. These are by far the most important visual properties of objects. they informs about the object category, material, color, segmentation mask, shape and pose. Unfortunately, as shown in the above figure 3.5, they were not really (e.g. low variability and realisticness) considered in the ROBCOG project. In the context of this thesis, as illustrated in figure 3.6 various photo-realistic materials and textures were developed to yield a very rich-featured kitchen.

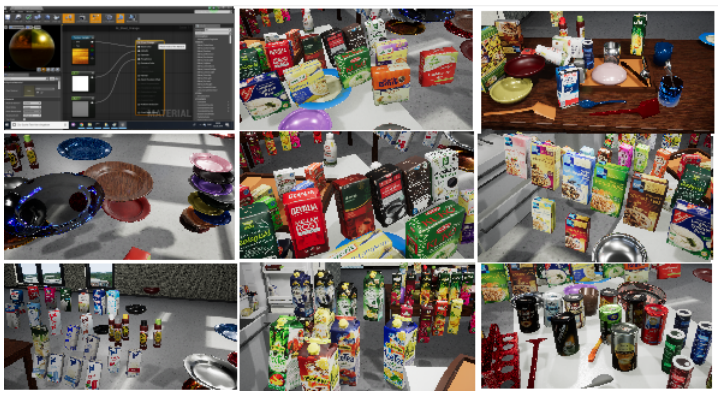


Figure 3.6 Various photo-realistic materials and textures for a very rich-featured kitchen.

While the object texture describes the spatial distribution of colors over the object surface, the material is concerned with the object surface behaviour faced to incident light, which is basically defined by the metallicity, the specularity, the opacity, the refraction index, roughness and emissivity of the object surface. Notice that this job is greatly achieved through the graphical API of UE, which constitutes a major positive feature of UE.

Mechanics-related properties. When an object is created in UE, it is automatically assigned a set of properties related to its mechanics (i.e. kinematics, statics, dynamics) which can then be set properly or even disabled to ascribe a particular mechanical behaviour to the object. Such mechanics-related properties encompass for instance the object mass, the object centre, the gravity force, the object collision property, position and orientation. Notice that properties such as position and orientation are subject to change during simulation.

Semantic static visual properties. One could argue that the object's mesh, material and texture are static visual properties. That is true, however, they are raw static visual properties and though they are easily semantically accessible to humans, it remains a challenge to machines, that this thesis aims at addressing. To avoid the chicken-egg dilema (i.e. epistemic circularity), the semantic static visual properties of each object are explicitly specified through an additional general-purpose property called Tag. An object's Tag is a record of its id-number, category name, color name, material name, shape name and openability status. The object's pose is internally generated by the UE simulator and can consequently be automatically tracked. As far as the spatial relationships among scene objects are concerned, their tracking relies on

fundamental properties of the external data collector, which is intensively exposed in the next sections. The following figure 3.7 demonstrates the configuration of objects'properties through the UE's Graphical User Interface (GUI).

Figure 3.7 Configuration of physical properties of virtual objects through the UE's GUI.

Object Behaviour

An object's basic behaviour may include its displacement (e.g. position-related), free fall (e.g. gravity-, mass-related), rotation (e.g. orientaion-related), deformation (e.g. scale-, splitting-, shearing-related), appearance mutation (e.g. material-, texture-related). These basic behaviours which are quite often triggered by either external or internal events and which more complex behaviours can be derived from, are automatically supported by UE. To accomplish such basic behaviours, the user-defined response program to the triggering event just needs to send the appropriate physical properties' values to the UE simulator.

3.4.2.2 Robot Virtualization

As shown with the kitchen, the robot virtualization would include the real robot's essential components such as important sensors, actuactors and controllers.

Robot Actuators and Sensors. To track the essential robot components, let consider the following abstraction. Notice that why manipulating, the humanoid robot does not observe its own body through its camera, but rather the manipulated objects. For this reason, the robot body except its camera can be made invisible. This is also advantageous in the sense that it abstracts the useless robot-specific morphology. However, one could argue that the robot often perceive its own arms. Though this argument is valid, notice that the robot arm does not bring any information into the scene graph and can therefore be regarded as a simple background object. This being said, the robot body can be reduced to its RGBD-camera. To move the

robot, the robot controllers simply move the camera along the three dimensions of the space and rotate it around the three main space axes. Notice moreover that this robot camera with such a high degree of freedom can be interpreted as a robot with an extremly higher degree of freedom. UE 4.16 offers RGBD-camera models with important camera-specific properties such as the aspect ratio, the resolution and the fields of view. For accomplishing basic manipulation tasks such as fetching, picking up and placing objects, the robot controllers directly act on the objects rather than on the invisible robot arms: The robot actuactors (i.e. legs and arms) are soft, while its sensors (i.e. RGBD-camera) are hard.

Robot Controllers (I/O mechanisms). Finally, the robot controllers consist of external programs for moving the robot around the kitchen, making it to manipulate scene objects and collecting scene images through the robot camera as well as scene graphs associated with those images. These external programs operate as client applications under the UnrealCV protocol exactly as Web browsers operate under the Hypher-Text Transfer Protocol (HTTP) protocol. They send set- and get-requests to the UE simulator through the UnrealCV protocol (i.e. UnrealCV client + server) in order to set and get the properties of virtual components. Whereas setting the properties of virtual components indirectly causes the virtual robot to act on them (e.g. object displacement), getting their properties appears as a visual perception process from the virtual robot (e.g. scene images, scene graphs). The UnrealCV's set- and get-commands take as arguments the virtual component's type (e.g. object or camera), the id-number of the component and then the name of the object property whose value is being set or got. Since the component's crucial property Tag was added in the context of this thesis, this thesis clones and extends the UnrealCV's source code from [151, 152] in order to handle the component's property Tag. Since the component property Tag is essentially a length-variable list of strings, each string being a pair (attribute, value), it generalizes the virtual perception of objects' semantic static visual properties in UE. Figure 3.8 illustrates the virtualization of the kitchen robot in UE.

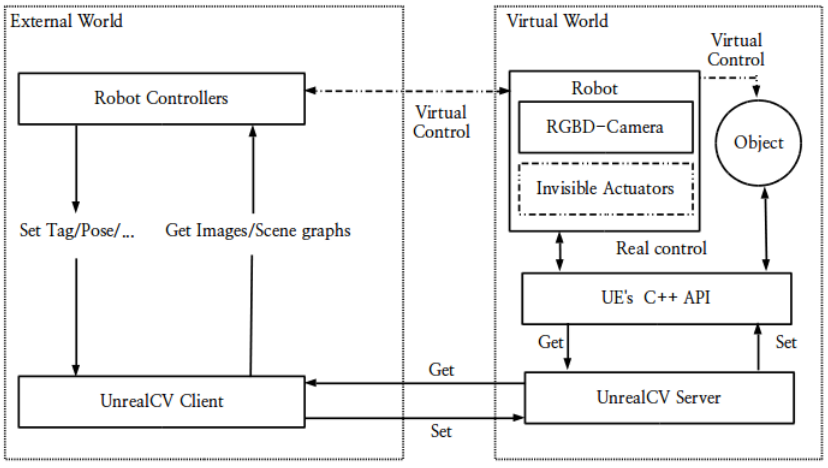


Figure 3.8 Virtualization of the kitchen robot in UE.

3.4.3 Simulation

So far, it has been shown how to virtualize the EASE kitchen as well as the robots inside it. Furthermore, it has been shown how the virtual robot can navigate inside the virtual kitchen, manipulate virtual objects and collect annotated scene images. However, it is still not clear how the virtual robot can perceive the spatial relations among virtual scene objects and moreover how the robot controllers flow. In this section, the algorithms that make up the robot controllers, responsible for the robot navigation, manipulation and collection of synthetic annotated images, are discussed. The simulation of the virtual world is then reduced to the execution of the robot controller,

3.4.3.1 Principle

The collection of synthetic annotated images from the EASE's virtual kitchen is achieved in many simulation episodes.

At the beginning of each episode $\mathcal{E}_j$, the kitchen is manually configured or decorated. This initial configuration $\mathcal{C}_0$ involves the setup of the scene backgroung (i.e. wall's, tables's materials and arrangement) $\mathcal{B}_j$, the placement of an initial set $\mathcal{S}_0$ of manipulable objects such as kitchen utensils in the scene and finally the establishment of the inital graph $\mathcal{G}_0$ of spatial relations among manipulable objects. Then, a 6-dimensional continous space $\mathcal{R}_j$ (i.e. $3D$-translation + $3D$-rotation) is defined, in which the virtual robot will navigate.

Afterwards, the robot navigation space $\mathcal{R}_j$ is discretized and for each robot pose $\mathcal{P}_i \in \mathcal{R}_j$, the robot controller gets the scene color- and depth-images $\mathcal{I}_i^{(c)}$, $\mathcal{I}_i^{(d)}$ as well as the scene graph $\mathcal{SG}_i = \{\mathcal{G}_i, ...\}$, in which each assertion has a confidence value of 1, and subsequently saves them to the hard disk. Note that each image is saved within a single image file and the scene graph within a json file. Then, the robot controller moves the robot to the next pose $\mathcal{P}_{i+1}$ and randomly changes the structure of the graph of spatial relations $\mathcal{G}_i$ to yield a new graph $\mathcal{G}_{i+1}$. Finally, for each object $\mathcal{O}_i \in \mathcal{S}_i$, the robot controller randomly changes its color, material, shape and openability, leading to a new set of objects $\mathcal{S}_{i+1}$. Notice that the poses of objects are modified while moving from $\mathcal{G}_i$ to $\mathcal{G}_{i+1}$. These changes lead to a new scene configuration $\mathcal{C}_{i+1}$.

In short, after every collection of an annotated image, the scene foreground is completely and randomly reconfigured in order to maximize the entropy in the collected dataset. Additionally, remember that the virtual robot only perceive a partial view of the scene $\mathcal{C}_i$, consequently of the set of object $\mathcal{S}_i$ and the graph $\mathcal{G}_i$. This allows to minimize the amount of episodes while maintaining the big size of data as well as their high entropy. As far as the scene background $\mathcal{B}_{j+1}$ and the robot navigation space $\mathcal{R}_{j+1}$ are concerned, they are changed at the beginning of the next episode $\mathcal{E}_{j+1}$.

In the context of this thesis, a dataset of around 71.000 annotated images has been collected through approximatedly 35 simulation episodes, each of which with averagely 2000 annotated images. While this dataset size is upperly bounded due to the restricted amount of time to accomplish this thesis, its lower bound is constrained by the domain representativeness rule which states that the minimal size of a statistically representative sample of an infinite population fairly distributed, with a confidence level of 99% and margin error of 0.485%, is around 71.000 [2, 144]. This level of confidence and margin error get significantly improved as the population turns to be unfairly distributed as it is the case with natural images. Finally, it is important to note that this approach, consisting of maximizing the dataset entropy while variating the scene theme was greatly inspired by the recent work of [11] entitled *Variations on theme*.

3.4.3.2 Algorithms

In this section, the above simulation principle is turned into algorithms, which are gradually presented from the most global routines to the most local ones. The following algorithm 5 globally presents the flow of the robot control during a single simulation episode.

Algorithm 3: Virtual Robot Control

Data: $\mathcal{R}$: is the discretized 6-dimensional robot navigation space for the simulation episode $\mathcal{E}$, $\forall(x, y, z, \theta_x, \theta_y, \theta_z) \in \mathcal{R}$, (x, y, z) and $(\theta_x, \theta_y, \theta_z)$ are respectively the $3D$-position and euler $3D$-orientation along the x, y and z axes of the virtual world's coordinates system.
$\mathcal{G}$: is the initial graph of spatial relations among scene objects for the simulation episode $\mathcal{E}$.
$UnrealCVClient$: is the UnrealCV protocol's client.

Result: Set of annotated images (color and depth images + scene graphs) of size $|\mathcal{R}|$.

```
/* get all interesting/foreground objects of the world and their respective mask colors */
VirtualWorldObjectIds ← UnrealCVClient.getAllObjectIds()
VirtualWorldObjectMaskColors ← ∅
for objId in VirtualWorldObjectIds do
    VirtualWorldObjectMaskColors.append([obj, UnrealCVClient.getObjectMaskColor(objId)])
end
```

```
/* move the robot all over R, manipulate objects, collect and save annotated images */
for (x, y, z, θx, θy, θz) in R do
    UnrealCVClient.setRobotCameraPose((x, y, z, θx, θy, θz)) ;          // move robot
    (I_rbg, I_mask, I_depth) ← UnrealCVClient.getRobotCameraImages() ;  // get actual scene images
    saveImages(I_rbg, I_mask, I_depth) ;                                // save acquired images
    /* Scene graph generation                                                          */
    JsonFile ← ∅ ;                                                      // Create json file
    VisibleObjectIds ← ∅ ;                                              // track visible object
    for [objId,maskCol] in VirtualWorldObjectMaskColors do
        if maskCol in I_mask then
            VisibleObjectIds.append(objId) ;                            // track visible object
            JsonObject ← ∅ ;                                            // Create json Object
            jsonObject.append(getObjectMask(I_mask,maskCol)) ;          // object mask
            jsonObject.append(UnrealCVClient.getObjectPose(objId)) ;    // object Pose
            jsonObject.append(UnrealCVClient.getObjectTag(objId)) ;     // object Tag
            jsonFile.append(jsonObject) ;                               // add object annotation
        end
    end
    G* ← updateSpatialRelation(G, visibeObjectIds) ;                    // update spatial relations
    jsonFile.append(G*) ;                                               // add spatial relations
    jsonFile.save() ;                                                   // save json file
    /* reconfigure scene foreground. nil is explained at 22                            */
    G ← reconfigureSceneForeground(G, nil, VirtualWorldObjectIds, UnrealCVClient)
end
```

In the above agorithm $\mathcal{I}_{rbg}$ is the actual $[480 \times 640 \times 3]$-RGB-image with byte values, $\mathcal{I}_{depth}$ is the actual $[480 \times 640]$-RGB-image with 32bits-float values and $\mathcal{I}_{mask}$ is the actual $[480 \times 640 \times 3]$-mask-image with byte values perceived by the robot camera. The resolution $[480 \times 640]$ has been chosen for its commonality in Robotics applications (i.e. essential view + low complexity). In the next section, the algorithm *updateSpatialRelation* 2 for deriving the partial graph of spatials relations $\mathcal{G}^*$ from the total graph $\mathcal{G}$ based on the robot camera view of the scene $VisibleObjectIds$ is presented.

Algorithm 4: updateSpatialRelation

Data: $\mathcal{G}$: total set of scene objects and spatial relations among them, $\forall(obj_1, r, obj_2) \in \mathcal{G}$, obj_1, obj_2 are scene objects and $r \in \{in, on, left, front\}$ a spatial relation.
$VisibleObjectIds$: is the set of actual partially observed scene objects.
Result: $\mathcal{G}^*$: the set of actual partially observed scene objects and the spatial relations among them.

```
G* ← G ;                                  // Initialize the partially observed scene
G⁻ ← ∅ ;                                  // Initialize the unobserved scene
```

```
for (obj_1, r, obj_2) ∈ G* do
    /* unobserved spatial relation                                                    */
    if obj_1 ∉ VisibleObjectIds ∨ obj_2 ∉ VisibleObjectIds then
        /* first object is unobserved                                                 */
        if obj_1 ∉ VisibleObjectIds then
            /* check all relations involving this object                              */
            for (obj_1^*, r^*, obj_2^*) ∈ G* do
                if obj_1 = obj_2^* ∧ r^* ∈ (in, on) ∧ obj_1^* ∈ VisibleObjectIds then
                    G* ← G*.append((obj_1^*, r, obj_2)) ;          // maintain in-, on-transitivity
                end
            end
        end
        /* second object is unobserved                                                */
        if obj_2 ∉ VisibleObjectIds then
            /* the unobserved relation is either left or front                        */
            if r ∉ {in, on} then
                /* check all relations involving this object                          */
                for (obj_1^*, r^*, obj_2^*) ∈ G* do
                    if obj_2 = obj_2^* ∧ r^* ∈ (in, on) ∧ obj_1^* ∈ VisibleObjectIds then
                        G* ← G*.append((obj_1, r, obj_1^*)) ;      // rethink aerial connections
                    end
                end
            end
        end
        G⁻ ← G⁻.append((obj_1, r, obj_2)) ;                        // track unobserved scene
    end
end
G* ← G* \ G⁻ ;                                                     // delete unobserved scene
return G* ;                                                        // return observed scene
```

Notice that due to the relaxations 17 applied on the compression scheme for reducing the complexity of spatial relations among scene objects, the above algorithm 2 that aims at deriving the observed scene $\mathcal{G}^*$ from the total observable scene $\mathcal{G}$ essentially focuses on two occlusion types. Whereas the first type regards the occlusion of containers (i.e. object that contains others) with reestablishment of aerial spatial relations (i.e. left, front) due to a loss of the locality property, the second type still concerns the occlusion of containers however with the reestablishment of inclusion relations (i.e. in, on) due to the respect of the transitivity property. In the next section, the algorithm $reconfigureSceneForeground$ 22 to reconfigure the scene foreground is finally presented.

Algorithm 5: reconfigureSceneForeground

Data: $\mathcal{G}$: total set of scene objects and spatial relations among them, $\forall(obj_1, r, obj_2) \in \mathcal{G}$, obj_1, obj_2 are scene objects and $r \in \{in, on, left, front\}$ a spatial relation.
$containerId$: is the id of the scene object containing the subscene to reconfigure. if its value is nil, then the whole scene $\mathcal{G}$ has to be reconfigure.
$VirtualWorldObjectIds$: the list of the ids of all the objects in the virtual world either present or not present in the scene
$UnrealCVClient$: is the UnrealCV protocol's client.

Result: $\mathcal{G}^+$: reconfigured scene.

$SceneObjectIds \leftarrow getSceneObjectIds(\mathcal{G})$; // get the ids of all scene objects
if $containerId = nil$ **then**
  /* get all the scene objects that are not contained by any other object. */
  $baseObjectIds \leftarrow getBaseObjectIds(\mathcal{G}, nil)$
end
else
  /* get all the scene objects directly contained by the scene object with id $containerId$. */
  $baseObjectIds \leftarrow getBaseObjectIds(\mathcal{G}, containerId)$
end
/* randomly permute the positions of base objects in the scene, each moving with the objects it contains */
$\mathcal{G} \leftarrow randomPermuteObjects(\mathcal{G}, UnrealCVClient, baseObjectIds)$
for $objId \in baseObjectIds$ **do**
  /* get the new pose of each base objects */
  $(x, y, z, \theta_x, \theta_y, \theta_z) \leftarrow UnrealCVClient.getObjectPose(objId)$
  /* fairly and randomly select an object $objId^*$ of the same type as objId according to the color, shape, material, openability and which is not already in the scene */
  $objId^* \leftarrow getRandomObjectId(UnrealCVClient, objId, SceneObjectIds, VirtualWorldObjectIds)$
  /* get the pose of the randomly selected object */
  $(x^*, y^*, z^*, \theta_x^*, \theta_y^*, \theta_z^*) \leftarrow UnrealCVClient.getObjectPose(objId*)$
  $UnrealCVClient.setObjectPose(objId^*, (x, y, z, \theta_x, \theta_y, \theta_z))$; // physical swap $objId$ and $objid^*$
  $UnrealCVClient.setObjectPose(objId, (x^*, y^*, z^*, \theta_x^*, \theta_y^*, \theta_z^*))$; // physical swap $objId$ and $objid^*$
  $objId \leftarrow objId^*$; // logical swap $objId$ and $objid^*$
  $\delta_z \leftarrow randomNumber()$; // randomly sample a real number
  $\theta_z \leftarrow \theta_z + \delta_z$; // also randomly rotate the object around the z-axis
  $UnrealCVClient.setObjectPose(objId, (x, y, z, \theta_x, \theta_y, \theta_z))$; // apply the rotation
  /* recursively reconfigure the subscene contained by $objId$ */
  $\mathcal{G} \leftarrow reconfigureSceneForeground(\mathcal{G}, objId, VirtualWorldObjectIds, UnrealCVClient)$
end
$\mathcal{G}^+ \leftarrow \mathcal{G}$; // new configuration of scene foreground
return $\mathcal{G}^+$; // return reconfigured scene

Foremost, notice that in the three algorithms 22, 5, 2 just presented above, some routines such as $randomNumber()$, $getRandomObjectId()$ and $getAllObjectIds()$ have not been explicitly

defined. This has been simply done for the sake of code lisibility since those routines, whose full definitions are much more susceptible to noise the essential ideas behind those algorithms rather than clarify them, are indeed very basic and essentially consist in linear searches of elements and whose internal workings can be directly inferred from their identifiers, parameters and comments about them. Additionally, the three algorithms appear to run in polynomial time and particularly quadratic given the depth of nested linear loops in the codes. Finally, notice also that after permuting the locations of base objects at line 8 of the above algorithm 22, their orientations θ_z around the z-axis are randomly adjusted. More than increasing the dataset entropy, contrarily to the random adjustment of other orientations namely θ_x, θ_y, this operation preverses the newtonian statics of scene objects. Figure 3.9 illustrates various aspects of the collected dataset.

Figure 3.9 A brief overview of the collected synthetic dataset. Various aspects such as occlusion, clutter, view angle, photo-realisticness, feature variability (entropy) and illumination are demonstrated.

3.4.4 Augmentation: Real Data Injecttion

Though the collected dataset is significantly rich and big, notice that the collected images lack of background noise such as chairs, cables, pictures, bags, people, dirty items, filled utensils (e.g. cup of juice), robot arms and paper sheets, which are not really specific to the target environment, however quite often occasionally present in the scene and constituting a challenge for vision systems. Since modeling such noise is an extremly challenging task in virtual reality, the collected synthetic dataset is augmented with some real images, well known for incorporating such background noise.

To achieve this data augmentation, approximatedly 610 real images were collected, around 110 of which from online publicly avalaible datasets and about 500 acquired by the RGB(D)-camera of a humanoid robot in the real EASE's kitchen, each of which annotated with object categories, colors, shapes, materials, openability, bounding boxes, segmentation masks and spatial relations among objects. However, note that due to the resulting work overload, those images were only partially annotated with respect to spatial relations and segmentation masks, but totally with respect to other features. The GUI-software *Labelme* [32] was used to annotate each of these 610 RGB-images with object Tags (i.e. category, color, shape, material, openability) as well as instance segmentation masks and bounding boxes. For tagging the images with spatial relations among objects, a special GUI-software *relationAnnotator* was developed in the context of this thesis in Python programming language. Notice that object $6D$-poses were ignored during annotation. On the one hand, this is due to the fact that it is very difficult to manually estimate them. On the other hand, they are not necessary for achieving the goal (i.e. robust background discrimination) requiring a data augmentation. Finally, note also that such a data augmentation further supports the Virtual Reality (VR) modeling in its weakness at accurately capturing features such as object materials (e.g. plastic). The following figure 3.10 presents the annotation of those real images with the softwares *LabelMe* and *relationAnnotator* as well as some of those images. One could argue that this real dataset is too small for Deep Learning, fortunately it can be significantly augmented in runtime by slightly distorting those images while training the vision system.

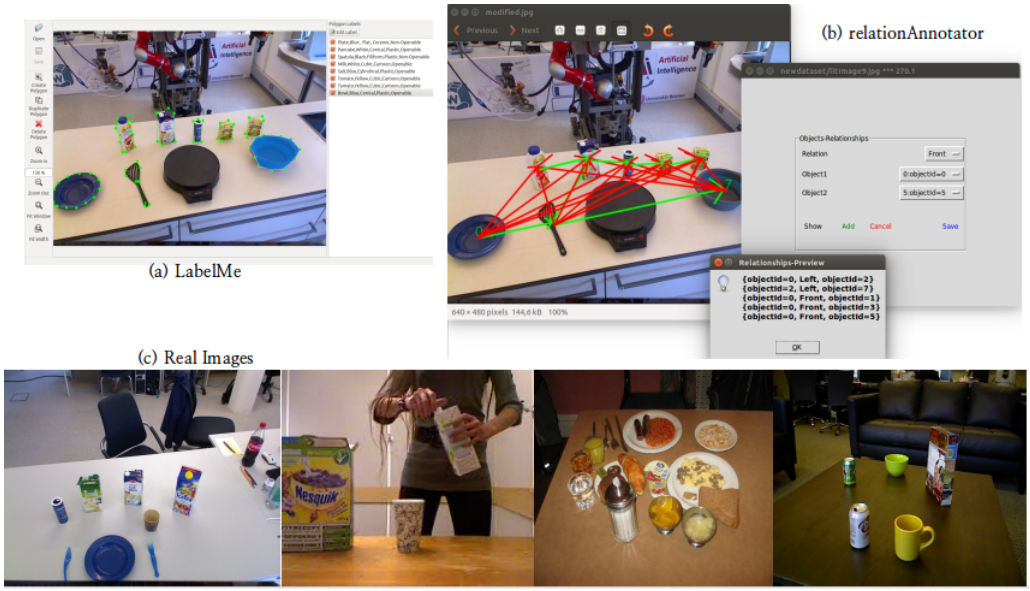


Figure 3.10 Augmentation of synthetic datasets with real datasets. (a) exposes *LabelMe*, (b) presents *relationAnnotator* and (c) illustrates some real images.

3.5 Conclusion

The aim of this chapter was to present the methodology used to acquire a big rich annotated dataset to train and test a DL-based vision system to effectively and efficiently support humanoid robots in the accomplishment of complex human-scale manipulation tasks in home environments, while providing a nearly complete and structured semantics of the scene called scene graph. Since the EASE research project was the very motivation of this thesis, its kitchen, located in the IAI-Bremen's laboratory and specially designed for research experiments, was chosen as the training and test domain. Given that it is nearly impossible to come out with a manually acquired and scene-graph-based annotated big rich set of training and test images from this kitchen, the latter was photo-realistically virtualized as well as the EASE's humanoid robot through the VR engine UE version 4.16 together with a self-adapted UE's plugin UnrealCV version 3.10, and then simulated, where the virtual EASE's robot automatically navigated in the virtual kitchen, manipulated objects and collected around 71.000 scene images, each of which annotated with a scene graph. However, due to the lack of enough background noise such as cables, dirt, chairs and people in the collected synthetic dataset, the latter was augmented with around 610 real RGB(D)-images, majoritarily acquired from the real EASE's kitchen and annotated partially through the software *LabelMe* and tagged with spatial relations through a self-developed software called *relationAnnotator*. This real dataset, which seems to be small (i.e. 610 images) can be significantly augmented in runtime by slightly distorting those real images while training the vision system.

Chapter 4

RobotVQA

See, deeply learn, fully Describe, Answer

4.1 Introduction

In the introductory chapter of this thesis, motivations for humanoid robots that can competently perform complex human-scale manipulation tasks were exposed as well as the power of DL-based computational models to address this problem formulated as the PVSURMT. Afterwards, the major theories of Deep Learning were reviewed in the second chapter, where it came out that supervised CNN(s) together with RNN(s) were the most appropriate frameworks to tackle the PVSURMT, however demanding enormous rich annotated training and test datasets. In the third chapter, Virtual Reality was leveraged to acquire such a dataset, which was subsequently augmented with real data in order to incorporate enough background noise, making therefore the system training much more robust. This chapter very detailly presents the DL-based model proposed in this thesis, called RobotVQA, to address the PVSURMT.

4.2 Tradeoffs

Scene graph has been proved to be an appropriate formalism to semantically represent visual scenes in the context of robot manipulation tasks and a big rich annotated dataset that can be used to train and test a DL-based computational model that aims at solving the PVSURMT while accepting scene images as inputs and outputting the corresponding scene graphs has been built. This being said, it remains the big and last question of how to build a DL-based mechanism to output scene graphs of inputted visual scene images. As a good research practice, this thesis does not aim at blindly reinventing the wheel, but rather at building upon the state-of-the-art

achievements in Deep Learning which are actually the condensed fruits of decades of researches. Since most of the top achievements in the state of the art present some advantages and inconvenients (e.g. scope, accuracy, complexity), it is then suggested that these past achievements be leveraged on the basis of the advantages they offer. Below are the main criteria considered in exploiting those works.

Scope and Transfer Learning In Deep Learning, one of the most stupidest attitude consists in completely building and training a model from scratch. This is due to the fact that DL-based models carry millions of parameters and consequently require a huge amount of data, time, dedicated techniques and hardware to be trained efficiently: this remains the tasks of famous DL-driven Companies such as Google LLC and Facebook Inc. Moreover, structuring deep neural networks, as mentioned earlier, is a very tedious and research task on its own. This has been proved while reviewing the literature in the field of DL-based CV and RV showing that almost all models proposed so far are reductible to a very small set of well-established deep frameworks which are themselves built the ones upon the others. is built in such a ways that at least its basenet, the most parameterized part of it, is not trained from scratch. This practice, that consists in building new deep models upon past deep models and using their learned sypnatic weights for initializing the formers That is, the model is known as transfer learning. Transfer learning significantly reduces the model design effort, implicitly augments the data, and speeds up the training of the new network. This transfer usually concerns the basenet, firstly because it is the least task-independent and deepest section of the model (i.e. feature extration). Secondly, due to its depth, the basenet is most of the time victim to the vanishing gradient problem and other depth-related problems. The closer the scopes of the old and the new model, the more efficient and effective the transfer learning.

Structuredness. The main reason why scene graph was adopted as an appropriate formalism for semantic scene representation was its ability to maintain structureness, which is essential for primitive agents such as robots. Unfortunately, a lot of the reviewed state-of-the-art solutions ignored it since they were not intentionally design for primitive reasoning agents such as robots but rather humans. However, a few of them show a great interest to the concept of scene graph.

Completeness. Scene graphs have been designed to provide as much information as possible about the scene which can effectively and efficiently support the robots in manipulation tasks. In the context of Deep Learning, such a completeness of the scene semantics implies multi-task learning. Notice that most of the reviewed models only focus on primitive or atomic CV problems such as object classification, segmentation and pose estimation. None of them provides a unified framework to address all of those problems at once. One could theoretically argues that those solutions could be grouped within a single package in order to address much more complex problems, however this is almost undoable practically. Firstly, some of the subproblems such as spatial relation, shape and material estimation are not clearly or addressed at all. Secondly, seen the spatio-temporal complexity of a single deep network, it is not reasonable to load a

single model for each required primitive task. And finally, each of these deep networks is trained on a different data domains and all of them cannot consequently be applied to a common data domain for inference. This thesis aims at summarizing the key principles of each of these reviewed solutions within a unified model and trains the unified framework on a single homogeneous dataset in a multi-tasking way.

Accuracy. Research is not only about solving new problems, but rather also about improving past solutions. The PVSURMT has been majoritarily addressed with traditional CV approaches, which have been shown to be significantly more inferior than DL-based approaches. On the other hand, some DL-based solutions that go deeper in addressing the PVSURMT are bottlenecked by limited building blocks (e.g. feature extractor). That is the case for instance in [13]. This thesis aims at incorporating more powerful building blocks to improve the overall accuracy.

Realtimeness. One of the most important criteria in Mobile Robotics applications is realtimeness. Realtimeness is the ability of a system to respond in realtime. Such a software should therefore garantee a certain response time delay. As far as the PVSURMT is concerned, the vision system should be fast enough to garantee a fast perception-guided robot manipulation. In the introductory chapter, it was shown that a visual perception speed's lower bound between 7 and $13Hz$ was acceptable.

4.3 Proposed Model

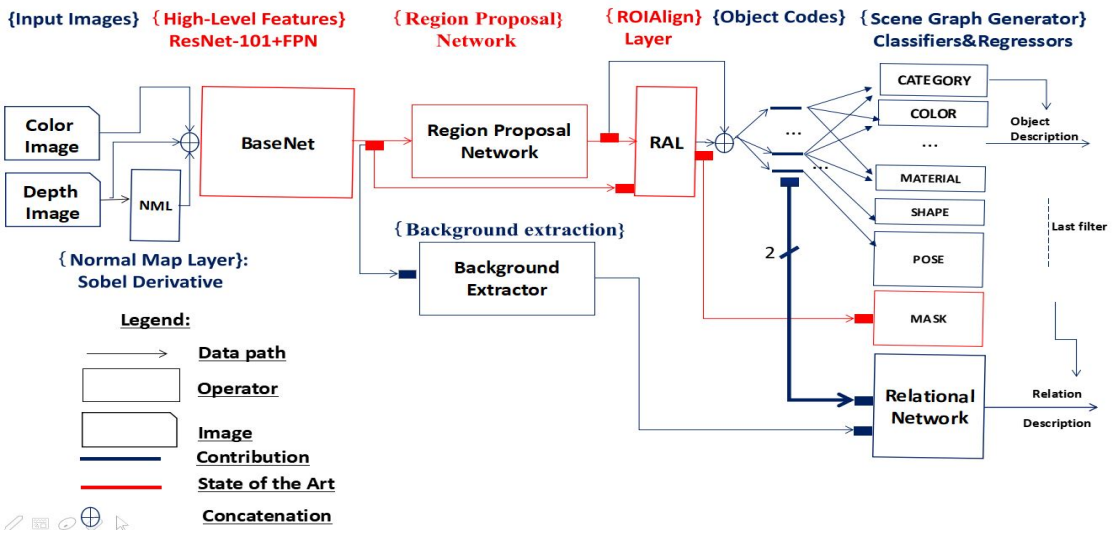


Figure 4.1 Architecture of RobotVQA

4.3.1 Inputs

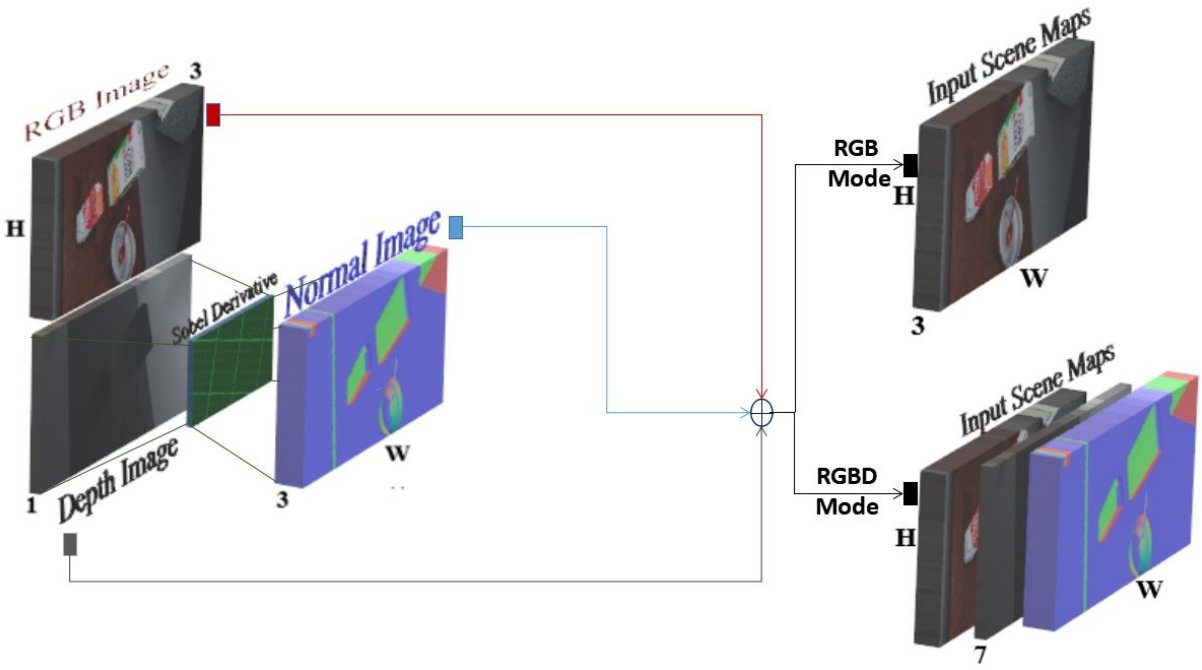


Figure 4.2 RobotVQA's Input Scene Maps

As shown by figures 4.1 and 4.2, RobotVQA is a multimodal system as it works for both RGB- and RGBD-scenes. This multimodality significantly increases the portability (a.k.a democratization) of the system. Additionally, providing a system for deep RGB-scene understanding would be a significant contribution in approaching the human vision, given that RGB-cameras are the most appropriate, not to say the perfect, model of the human eye. As discussed in the second chapter, instead of going for a multi-pipeline processing system, where each scene image mode owns a separate processing pipeline and yielding therefore to a more computationally complex system, the combination of all scene maps into a single multi-channel input scene image is adopted in this thesis. The input layer of RobotVQA consists of:

An RGB-Image. This image represents the fundamental raw representation of the scene. It approaches a raw representation of the scene when observed by the human eye. Mathematically, it corresponds to a $[W \times H \times 3]-$ dimensional byte-array, meaning that the pixels take their values in the interval $[0.0, 255.0]$. To avoid unstable update of RobotVQA's synaptic weights (i.e. one-way gradient), the pixel values are normalized by shifting and centralizing them around the mean pixel value, which itself is obtained by averaging all the values of pixels that constitute the images within the dataset built in chapter 3. Imagine that the pixel values are enclosed

within the interval $I = [250.0 - dv, 250.0 + dv]$ with a mean pixel value mpv of 250.0. Then, instead of naively projecting those values onto the space $[0.0; 255.0]$ (e.g. only positive), it would be preferable to shift, centralize them around mpv and only encode their deviations from the mean. These deviations as image pixel values could now be uniformly projected onto the much more confined space $[-dv, dv]$ (e.g. positive and negative). This operation is also crucial for efficiently dealing with symmetric neural activations such as the sigmoid and tanh functions in order to avoid neural saturation. As indicated in the third chapter, W and H are respectively fixed to 640 and 480 in this thesis. This resolution is on the one hand the standard for most robot cameras and Robotics datasets. On the other hand, it allows the image to be resized a couple of times through a division by 2 in order to build a pyramidal view of the scene (Feature Pyramidal Network (FPN)). This concept of FPN will be better explained when talking about the basenet.

A Depth-Image. Instead of measuring the light spectrum and intensity reflected by scene points, the depth image measures the distance from the camera center to scene points. Mathematically, it is defined as a $[W \times H]-$dimensional floating-array. Contrarily to the rgb-image pixel values which are natively 8-bit integers, depth-image pixel values are natively 32-bit floats for more precision, however are normalized to the interval $[0.0, 255.0]$ based on the max depth found in the dataset and then around the mean depth-pixel as done above with RGB-images. Note that the above description of the depth map is the most native of it. That is, it serves fundamentally at detecting obstacles as well as their locations (i.e. distance). However, by applying additional simple operations on it, it leads to subsequent maps able to deliver a broad range of higher-level semantics such as object shapes and inclusion relations among objects. One of those secondary depth-driven maps is the normal map. This is a colorful or RGB-version of the native depth-map, which provides a lot of cues about the shape of objects in the scene. The normal map is obtained by simply computing the gradient module at each pixel or point of the depth-map. This can be done by applying an appropriate convolution operation such as the Sobel operator. Since the basenet in CNN(s) are deeply convolution-based, one could argue that the deep model will automatically infer such an operator when given the native depth-map to estimate the shapes of objects. Though such a view is theoretically valid, practice strongly suggests to orientate as much as possible the deep model and as mentioned several times before, this practice is called network design and constitutes an entire research subfield on its own. For this reason, the normal map is first computed from the depth map, afterwards normalized as suggested previously and then aggregated to the latter to form a depth-driven $[W \times H \times 4]-$dimensional scene image.

A mode bit. This bit indicates whether the system should consider the depth image or not. When set to 0, the system works in RGB-mode otherwise in RGBD-mode. In RGB-mode, the input is solely represented by the $[W \times H \times 3]-$dimensional float-array from the RGB-image, whereas it is represented by the $[W \times H \times 7]-$dimensional array from a combination of the RGB-image's array and the depth-driven image's array.

4.3.2 Basenet

This is the core, not to say the essence, of any deep model in terms of functionalities, spatio-temporal complexity and parameterization complexity. As demonstrated by figure 4.3, it takes as input the image's floating $[W \times H \times C]-$array ($C \in \{3, 7\}$) and extracts a deep hierarchy of features from it. The hierarchy in this context essentially encodes the level of semantics or abstraction of the features. This deep hierarchy of features, represented as a set of 5 $[W'_i \times H'_i \times D']-$dimensional float-arrays ($D' = 256$, $W'_i = \frac{W}{2^{i+1}}$, $H'_i = \frac{H}{2^{i+1}}$, $i = 1, .., 5$), constitutes the output of the basenet called feature maps.

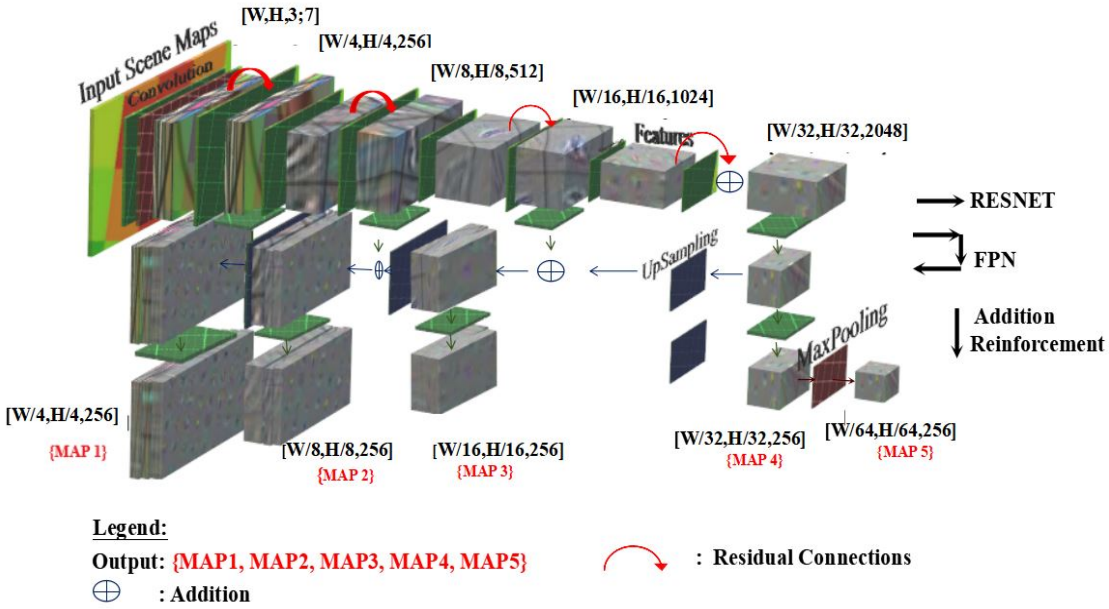


Figure 4.3 A Simplistic View of RobotVQA's Basenet

This thesis adopts the basenet proposed by the recent achievement Mask R-CNN on the one hand for its leadership in the state of the art in terms of accuracy (i.e. by far the most outperforming), scope (e.g. segmentation, classification, detection) and on the other hand for the quality of transfer learning it allows to this present work. As already mentioned in the second chapter, Mask RCNN's basenet is essentially a combination of two recent basenet's architectures namely the FPN [114, 161] and the Residual Network (ResNet). As far as ResNet is concerned, its main contribution consists in building a very deep networks while avoiding network-depth-related problems such as the well-known problem of vanishing gradient. This is performed by allowing very special inter-layer connections in the network called *shortcuts* or *residual connections*. In contrast to traditional inter-layer connections, shortcuts or residual connections allow non-consecutive layers to be directly connected. Such a direct connection between non-consecutive

layers allows the signal to still travel between them in case of saturation (e.g. overfitting, vanishing gradient) in the layers in-between (i.e. residual block). One of the two major goals of FPN is to preserve the spatial distribution of information while extracting the features. That is, $W' \approx W$, $H' \approx H$ and $\frac{W'}{H'} \approx \frac{W}{H}$. This preservation is so helpful for operations such as semantic (or instance) segmentation and estimation of spatial relations among objects. As demonstrated in figure 4.1, this is achieved through a sequence of downsampling convolution operations to achieve deep semantics and subsequently followed by a sequence of upsampling deconvolution operations to restore the spatial distribution of information, where additional shortcut (a.k.a. lateral) connections allow the convolution block to integrate information in the deconvolution block in order to support the upsampling operations since they are fundamentally information-lossy computational processes. Moreover, note that while doing this, the FPN also generates multi-scaled feature maps (i.e. pyramid of feature maps) that allow a scale-invariant processing (i.e. detection, segmentation) of scene images and objects. This handling of scale-variable input images is the second major advantage of the FPN.

4.3.3 Region Proposal Network

Complex visual tasks quite often require to extract information about specific regions or objects in the scene. These objects and regions are called regions (or objects) of interests. To extract those information, there have been however two broad approaches namely the bling processing and the region-based processing. While the first approach (e.g. YOLO) aims at globally processing the input image without any regional discrimination to output the required information, the second approach (e.g. SSD, Faster R-CNN) foremost localizes the regions (or objects) of interest. As for any blind, brute-force or non-informed procedure, the blind approach paies the burdensome price to perform badly when both the scene and the information to extract become complex (e.g. cluttered scene).

In the context of CV, such an informed differentiable ANN(s)-based procedure, called RPN(s), was first proposed by R-CNN, then improved by Fast R-CNN in term of running time and further by Faster R-CNN later and finally in term of accuracy by Mask R-CNN, to explicitly discover the regions of interest in the scene and subsequently process them separatedly in order to extract region-specific information or combine them to extract global information. Review of the literature indicates that Faster R-CNN significantly outperforms in terms of accuracy all non-R-CNN deep models designed for object localization in multi-object scenes. Though most compertitors of Faster R-CNN are considerably faster, carefull experiments on very simple General Purpose Graphical Processing Unit(s) (GPGPU(s)) suggest that Faster R-CNN can run at $5Hz$ while running on large frame (e.g. $[1024 \times 1024]$) containing up to hundred interesting objects. Furthermore, note that reducing the frame size as well as the number of target objects in the frame scales down the running time by an important factor (i.e. 2), since those metrics are

closely related to the RPN(s)'s processing time (i.e. number of region proposals). Fortunately, robot camera frames (e.g. [640 × 480]) as well as the number of interesting objects in robot scenes (e.g. 20) are by far smaller and the actual GPGPU(s) have significantly improved.

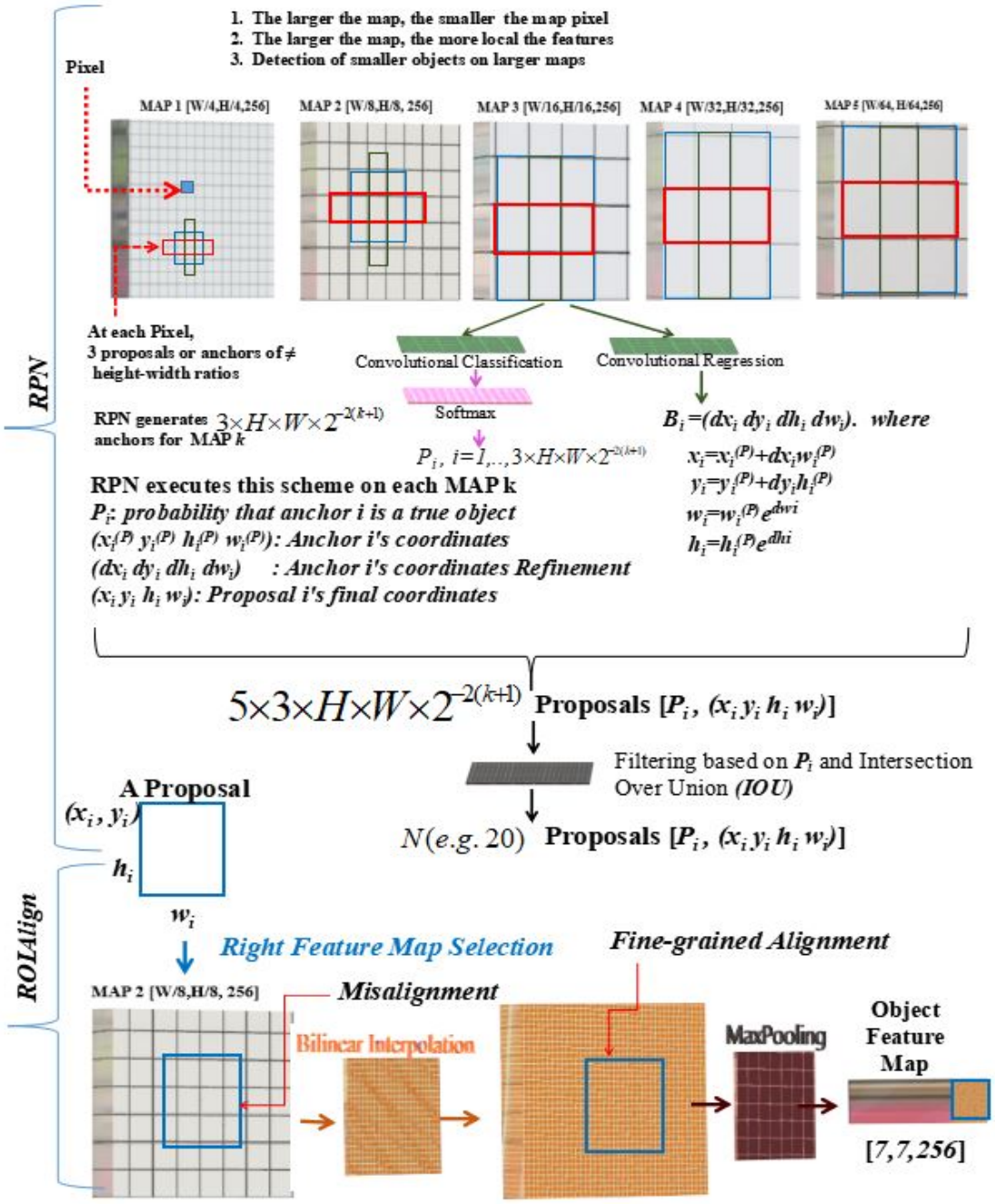


Figure 4.4 A Simplistic View of RobotVQA's Region Proposal Network

As illustrated by figure 4.4, the RPN(s) takes as input the input scene image 's feature maps ($[W'_i \times H'_i \times D']-$dimensional arrays) from the basenet and outputs a fix-cardinal set of object feature maps ($[W" \times H" \times D" \times N]-$dimensional array), where N (e.g. 20) and $[W" \times H" \times D"]$ ($[7, 7, 256]$) are respectively the number of objects of interest a.k.a the set cardinal and the dimension of each object's feature map . After determining the bounding box ($[X, Y, W", H"] = [x_j, y_j, w_j, h_j]$) of an interesting object in the image, a very special operation called Region Of Interest Pooling (ROIPool) is charged to extract the object feature maps ($[W" \times H" \times D"]-$dimensional arrays) from the basenet ($[W'_i \times H'_i \times D']-$dimensional arrays). However, if the equation 4.3.2 (i.e. preservation spatial distribution of information) is not satisfied, any attempt to properly align the bounding box $[X, Y, W", H"]$ on the right feature maps' plan (W'_i, H'_i) and project it later onto the image plan (W, H) will result in condiderable deviations due to extrem rounding. To solve this issue, Mask R-CNN proposed a much more accurate version of ROIPool called Region Of Interest Align and Pooling (ROIAlign) that leverages bilinear interpolation to properly align the bounding box $[X, Y, W", H"]$ on the right feature maps' plan (W'_i, H'_i) before extracting the object feature maps. RPN(s) has been a major breakthrough in the field of visual understanding of multi-object scenes and can be seen as an attention mechanism. In this thesis, RPN(s) is incorporated in the proposed model to localize manipulable objects (e.g. table, utensils, food boxes) in the robot scene. MaskRCNN's RPN(s) is particularly adopted for both its improvements over Faster R-CNN's RPN(s) (ROIAlign) and the transfer learning it enables.

4.3.4 Model Head

This part of the network, often regarded as the network output, finalizes the mechanism and outputs the intended information as the model is fed with an input. It has been clearly stated earlier in this thesis that the underlying PVSURMT can naturally be decomposed into two subproblems namely the dense description of objects and the description of spatial relations among those objects.

Obviously, one of the most intuitive way of addressing these subproblems, given the set of interesting scene objects from the RPN(s) and the criterion of structuredness required by Robotics aplications, would be to build two distinct modules at the head of the model, the one head for object description and the other for spatial relation description. While the object describer takes an object's feature maps and outputs its properties, the spatial relation describer combines two objects' feature maps and outputs the most relevant spatial relation between them. This approach, called Multiple Network Heads, Multiple Outputs (MNHMO) in this thesis, is usually referred to as multi-task learning in the field of Deep Learning. A common alternative to MNHMO is qualified to as Single Network Head, Multiple Outputs (SNHMO). SNHMO aims at combining the feature maps of all scene objects from the RPN(s) to return a single multidimmensional output vector containing all the required information (i.e. scene graph).

When comparing both alternatives, it is suggested to go for explicit multi-tasking namely MNHMO. When dealing with SNHMO, there are numerous drawbacks to face. Firstly, it makes transfer learning at the network head very difficult, not to say impossible. Secondly, it makes it very hard to train the model on different datasets, since they are usually annotated for single tasks. Thirdly, SNHMO makes it impossible to design task-specific procedures (e.g. classification, regression) and therefore hard for the single network head to actually understand the global function (scene graph) being computed. Contrarily to SNHMO, MNHMO more than presenting the same advantages as SNHMO namely the paralellelism and multi-tasking, prevents the above issues faced by SNHMO. One could even argue that MNHMO would be spatio-temporally more complex than SNHMO. However, drawbacks of SNHMO, inherent massive paralellism in DL-based architectures and todays massively parallel architectures (e.g. GPGPU(s)) show that this argument does not deserve any support.

4.3.4.1 Object Description

In this section, the solution to the problem of object description is discussed in details. Recall that the problem of object description consists for each localized object of interest in estimating its category, color, material, shape, openability, $6D$-pose, and segmentation mask.

Since the proposed model RobotVQA is an ANN(s) intended to be trained supervisedly, this problem of object description can be viewed as a classification problem, regression problem or a mixture of both. Furthermore, structuredness being crucial for Robotics applications and particularly one of the goals of this work, it is suggested consequently to adopt a classification/regression scheme that will take each object's feature maps ($[W" \times H" \times D"]-$dimensional array) and outputs a description that can be decoded without ambiguity to get the object's properties. The process will then be iterated over the list of objects of interest found in the scene image. Note that this is a loose RNN(s) and benefits consequently from some advantages of RNN(s) such as the avoidance of weight number explosion and the invariance to input length as well as arrangement due to input (i.e. set of objects) sequentialization and weight sharing. The mechanism is qualified as loose because there is but logically no need to keep an internal state (a.k.a sequentiality tracker) on the one hand, since it is not a typical sequence problem, and on the other hand because the maximal size N of the input list of objects is known in advance.

The most common solution to output an object description without ambiguity is to accept a multidimensional output where each dimension corresponds to a well-defined information (i.e. object property in this case) obtained either by classification or regression. Given that an object's category, color, material, shape, openability and segmentation can only take their values from well-defined discrete sets, their estimation can be achieved through a classification, whereas the object's pose due to its continuity can better be regressed. There are two ways for an ANN(s) to return multidimensional outputs. Whereas the first alternative SNHMO consists in treating the

output vector as a single unified response to the input, the second alternative MNHMO relies on explicit multi-tasking, where the ANN(s) is made up of several heads, each of them outputting a single dimension of the output vector. When comparing both alternatives, it is suggested, as demonstrated earlier just before this section, to go for explicit multi-tasking namely MNHMO. This makes the whole model RobotVQA massively parallel, multi-tasking and adaptable.

Now that the broad structure and nature of the proposed model's head and the object describer have been explained, let go deeper into the internal working of the object describer from the inputs, through internal mechanism to outputs.

Object's DNA Code

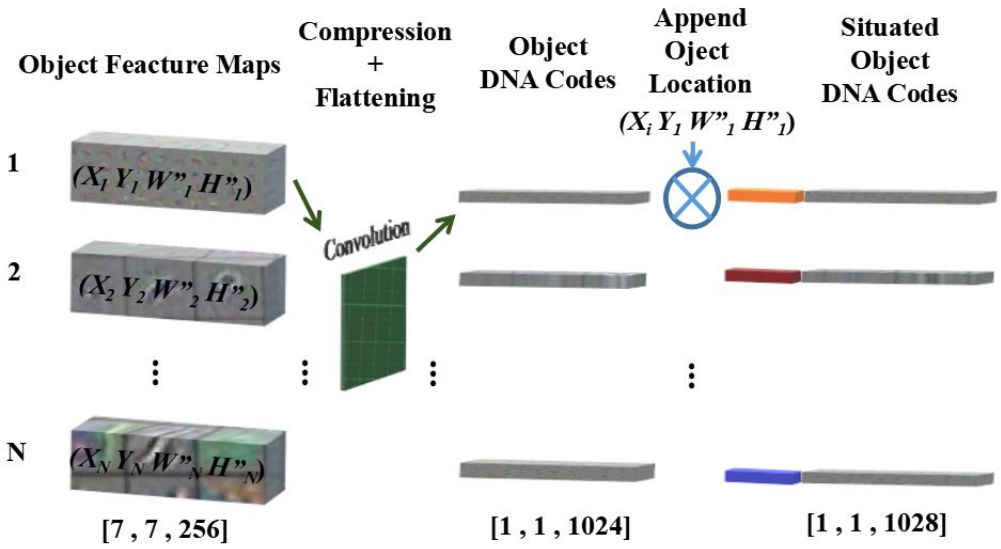


Figure 4.5 Object DNA Codes

It is known that the proposed model RobotVQA possesses multiple heads, each of them being responsible for outputting an object's property or spatial relation among objects. Naturally, it also follows from the above argumentation that each of this network head takes as input an object's feature maps and outputs the desired property's value. The process is then iterated over the list of or pairs of objects of interest found in the scene image. However, each of these network heads from the object category to the spatial relation recognizer differs from each other on how to reason on object feature maps. While the ones require a global reasoning about what each object's feature maps contain, the others very locally operate. An object's material has been shown to be an extremly emergent property that requires a global reasoning on the object's feature maps, while an object's semantic segmentation can better be achieved while reasoning locally, since the object's feature maps encode the semantics while preserving the spatial distribution of information. One can argue that the material recognizer can be fed directly with the whole object's feature maps. Unfortunately, the dimension of the object's

feature maps, being extremly big (i.e. $[W" \times H" \times D"]-$dimensional array), will cause the number of parameters in the material recognizer, made up of fully connected layers, to grow unreasonably and inevitably cause overfitting (i.e. curse of dimensionality).

To address this issue, the object's feature maps are preserved for the instance semantic segmentation task, while a much lower-dimensional version of the object's feature maps, called object DNA code, is derived through a convolution-based summarization of the object's feature maps to act as input for other tasks. Analogically, this object DNA code corresponds to the concept of Deoxyribo Nucleic Acid (DNA) in animal (human) genetics, which carries condensed information (i.e. genotypes) serving as principal determinant of the physical characters or properties of those beings (i.e. phenotypes={color, shape, etc}).

Additionally, it is important to note that the object's DNA code as well as its feature maps only carry information about the object itself and its close neighborhood in the scene, something that is not desirable for the estimation of some of its extrinsic properties such as its location and spatial relations to other objects. To address this issue, the object's bounding box ($[X, Y, W", H"]$), pre-estimated by the RPN(s), is further incorporated into its DNA code. This incorporation will play, as it is shown later in this thesis, a major role in the spatial relation recognition. Figure 4.5 illustrates the construction of object DNA codes.

Object's Intrinsic Properties

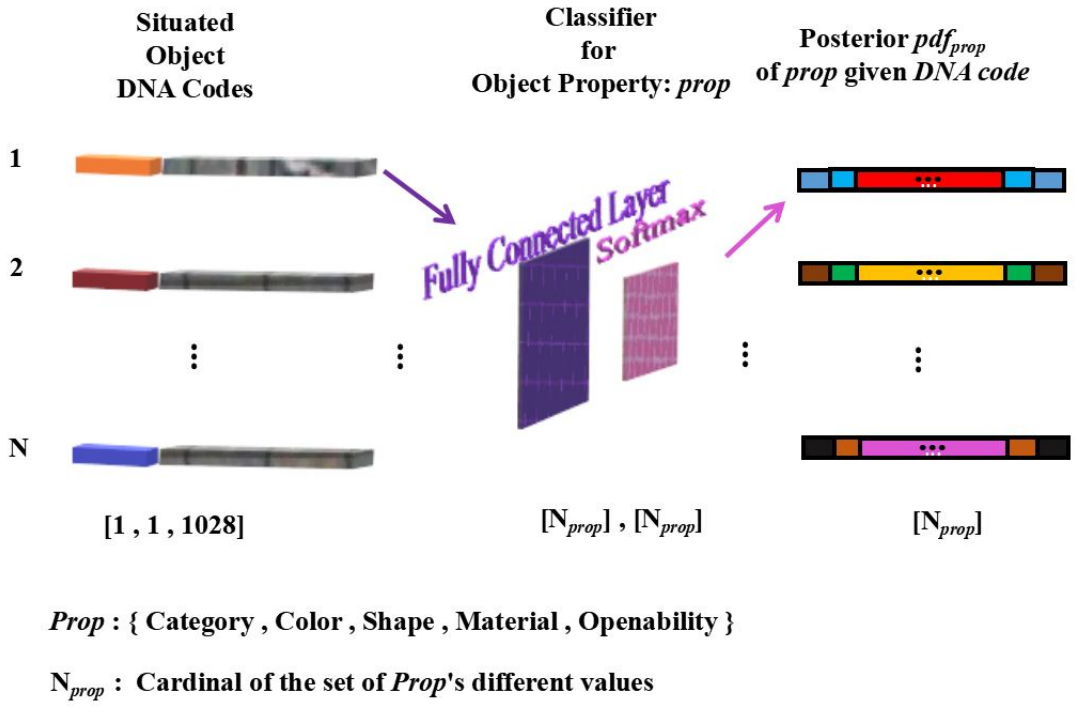


Figure 4.6 Architechture of a Single RobotVQA's Classifier

An object's intrinsic property is one, which remains unchanged no matter the context, it is observed in (i.e. visual constancy). In this work, the intrinsic properties comprise the object's category, color, shape, material and openability. Each of these properties takes its values from a finite set of well-defined values. Therefore, a classification scheme is suitable to recover them from DNA codes. As shown by figure 4.1, each of these properties *prop* is recovered by a distinct loose-RNN(s)-based MLP(s)-classifier, that takes as input an object's DNA code from the list of N DNA codes and outputs a $[N_{prop}]$-dimensional float-array corresponding to the probability distribution pdf_{prop} of the property *prop*, where N_{prop} designates the cardinal of the value domain of property *prop*. This probability distribution is then queued into an output list of size N. Recall that for each property, there is a distinct output list. Furthermore, as clearly shown in the second chapter, the probability distribution pdf_{prop} is conditioned on the input (i.e. images, DNA codes). This process is iterated N times by every loose-RNN(s)-based MLP(s)-classifier, that is untill the input list of DNA codes is completed. Then, for each classifier's output list, an argmax operation (i.e inverse of softmax's maximum) is applied on each probability distribution pdf_{prop} in entry to get the final property's value (e.g. red, blue, etc for the color property). Since the existence of an object's category fundamentally, conditioned by the existence of a bounding box, conditions the existence of all other properties, a filter is finally applied to reject inconsistent results.

Essentially, as demonstrated by figure 4.6, the internal of each classifier is made up of a MLP(s) with alternated relu, leaky relu and linear activations to allow non linearity while avoiding neural saturation (e.g. neural death). Note that the output layer (i.e. softmax) has already been described above.

Object's Extrinsic Properties

In contrast to intrinsic properties, extrinsic properties take their values depending on the context, they are observed in. In this thesis, they include the object's $6D$-pose and segmentation mask.

Object's $6D$-Pose. It records the $3D$-position of the object as well as its $3D$-orientation in the image's $3D$-coordinate system. Since the $6D$-pose is defined over a continuous space namely $\mathbb{R}^6$, the pose estimation is solved as a regression problem. To abstract camera-specific details (e.g. resolution, intrinsic matrix), a loose-RNN(s)-based MLP(s)-regressor is designed to regress the pose in the image coordinate system. The loose-RNN(s)-based MLP(s)-regressor works in a quite similar manner as the classifiers described above. It takes as input an object's DNA code from the list of N DNA codes and outputs a $[6 \times N_{cat}]$-dimensional vector; That is a $6D$-vector $(X, Y, Z, \Theta_x, \Theta_y, \Theta_z)$ for each object category *cat* (e.g. Bowl, Mug), where N_{cat} is the cardinal of the object category set, (X, Y, Z) and $(\Theta_x, \Theta_y, \Theta_z)$ are respectively the $3D$-position and $3D$-orientation of the object respectively around the axes X, Y and Z. Such a $[6 \times N_{cat}]$-dimensional vector is called a joint pose and queued into the output list. The process is then iterated over the complete input list of DNA codes. The reason for joining the object category to the pose, as done by Mask R-CNN in computing bounding boxes of objects, is that the pose of an object more than its bounding box stronly depends on its category. Finally, the right $6D$-pose is then inferred

by conditioning the joint pose on the object's category specified by the category classifier.

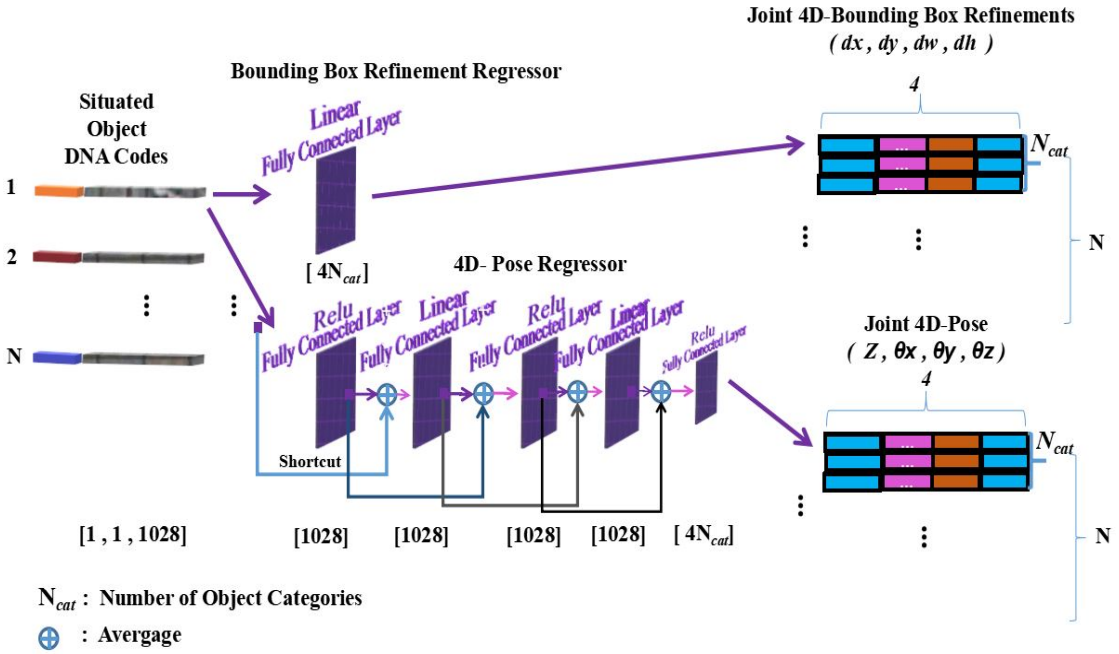


Figure 4.7 Architechture of RobotVQA's $6D$-Pose Regressor

As shown by figure 4.7, the internal working of the pose regressor is however not obvious as it was for classifiers. Before jumping into it, let make some remarks. The first remark is that the center of an object's bounding box is ideally its $2D$-position (X, Y) on the image plane except for very curved objects, which are fortunately unlikely to be encountered in Robotics manipulation areas. Secondly, it is almost unnecessary to recover the depth Z ,when working in RGBD-mode. But since the network should also work in RGB-mode, it is meaningful to explicitly regress the depth Z. This regression can be regarded as a denoising operation while working in RGBD-mode. Finally, it should be noted that those coordinates are well-bounded. More than being all non-negative real numbers, (X, Y) are upper-bounded by the image's maximal size, while Z is upper-bounded by a fixed maximal depth and $(\Theta_x, \Theta_y, \Theta_z)$ by 2π. To avoid a computation depending on the coordinates' boundaries (e.g. extrapolation) and ease it (e.g. covariate shift), each of these coordinates is normalized to the interval $[0, 1]$ and the whole $6D$-pose to the space $[0, 1]^6$.

To compute (X, Y), a separate regressor takes as input an object's DNA code and computes the refinement $([\Delta_{X"}, \Delta_{Y"}, \Delta_{W"}, \Delta_{H"}])$ of the object's bounding box $([X", Y", W", H"])$. Note that the bounding box itself was already pre-computed by the RPN(s). But since the computation was just a proposal and therefore not precise enough to be directly outputted, an object-DNA-code-based bounding box's offset is still computed to adjust the proposal made by the RPN(s).

For adjusting the proposed bounding box, the offset is added to it and the center of the resulting bounding box is returned as (X, Y). Finally, a second regressor was designed to take as input an object's DNA code and compute the $4D$-vector $(Z, \Theta_x, \Theta_y, \Theta_z)$. This computation has been separated because the mechanism for computing the bounding box from both the RPN(s) proposal and the object's DNA code was already proposed in MaskRCNN. Consequently, a transfer learning could efficiently be achieved.

The structure of the (X, Y)-regressor is left unchanged as proposed by Mask R-CNN. It is a MLP(s) with specialized relu activations that produce consistent outputs regarding the bounded space, the values of (X, Y) are drawn from, notably $[0, 1]^2$. As far as the $(Z, \Theta_x, \Theta_y, \Theta_z)$-regressor is concerned, it is made up of a MLP(s) with residual connections and specialized relu activations that ensure that the obtained values are consistent regarding their bounds and furthermore allow to tackle the complexity of the pose function while avoiding ANN(s)-depth-related problems such as overfitting and vanishing gradient.

Instance Segmentation. As shown by figure 4.8, the segmentation mask of an object is computed as a binary pixelwise classification over the input scene image's region delimited by the bounding box of the object.

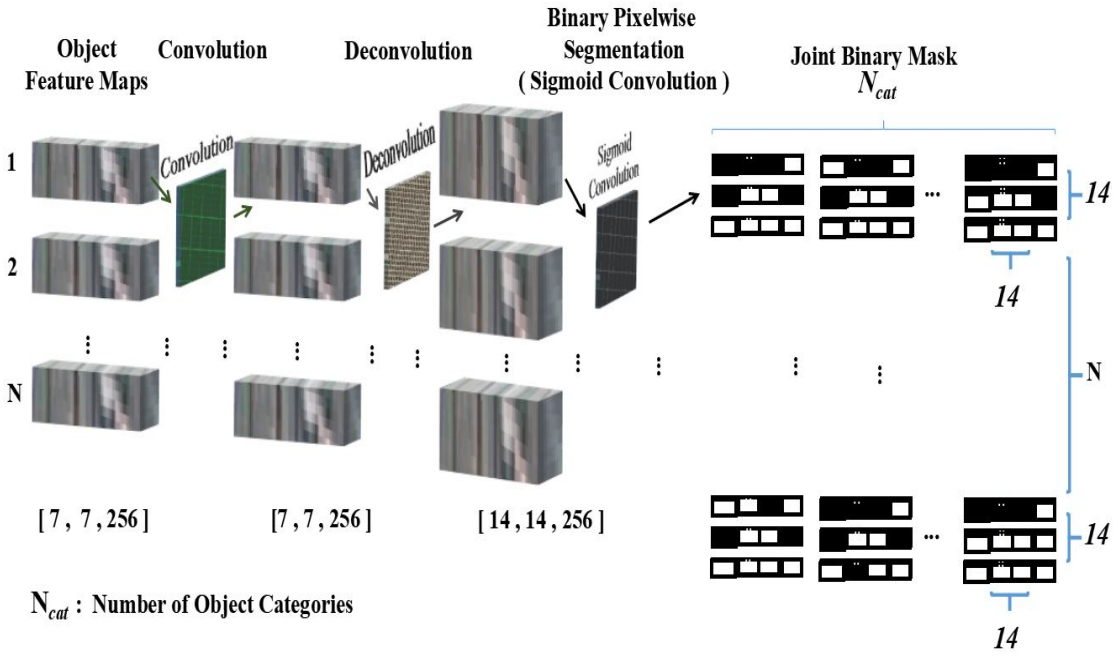


Figure 4.8 Architechture of RobotVQA's Instance Segmenter

It takes as input an object's feature maps, which constitute the semantics of the object neighbourhood in the scene image and preserve the spatial distribution of information in contrast to the object's DNA code, and then applies a sequence of convolutions and deconvolutions in order

to return the binary mask of the object. Since the object's mask is also closely related to its category, rather than directly computing the mask, a joint mask is computed, as done above with the object's pose, where a separated mask is computed for each category. Note besides that producing a full binary mask of the object for each class prevents also the classes to compete as it is the case with the standard pixelwise multiclass classification. The right mask will then be derived by conditioning the joint mask on the object's category returned by the category classifier. The topology proposed by Mask R-CNN is adopted in order to take advantage of transfer learning.

4.3.4.2 Spatial Relations

Explicit spatial relations among objects in the scene is one of the most important contribution of this work. Attempts to address this problem was certainly done in the past however with either shallow approaches (i.e. symbolism) or implicitly through VQA. This thesis points out the main challenges of this task and provides a DL-based solution to address them.

The first challenge in spatial reasoning is contextualization. As shown in the related works, a sparse analysis (i.e. symbolism) of the scene context fails to address this problem. That is, $6D$-poses and spatial relations, though most often interchangeably interpretated, are semantically different from each other. Spatial relations offer a higher semantics of the relative space occupancy by scene objects that cannot be reduced to $6D$-poses (i.e. visual constancy). This semantics is useful for understanding how scene objects interact with each other. To achieve it, spatial reasoning combines $6D$-poses and appearances of scene objects (i.e. object's feature maps) together with the scene background (i.e. $3D$-coordinate system of the scene). Unlike spatial relations, $6D$-poses provide precise absolute location as well as orientation of scene objects and consequently insights into how to grasp each of them. Notice that both the $6D$-poses and spatial relations are complementary.

The second challenge is the architecture of the spatial reasoner. That is its inputs, outputs and internal working. Since the set of spatial relations is finite and well-defined, classification appears to be the most appropriate scheme to address the problem. The reasoner takes a list of scene objects (i.e. object's feature maps) combined with the scene background and returns a complete directed graph, whose nodes are scene objects and directed edges are spatial relations between their vertices. To produce such a graph, there are, according to the literature, two common approaches. The first approach consists in inputting the whole list of N object together with the scene background at once and then outputting the whole list of N^2 spatial relations among objects at once, whereas the second approach serializes both the input (i.e. list of N objects) as well as the outputs (i.e. list of N^2 spatial relations), in which the the spatial reasoner accepts a pair of objects toghether with the scene background at a time and then returns the most plausible spatial relation among both objects. However, as argued earlier in the previous section

and further in the second chapter, the first approach does not suit well to this problem. On the one hand, the number of parameters will exponentially explode due to the size $(N \times M \times B)$ of the input layer, where M is the size of each object and B the size of the scene background. On the other hand, there is no weight sharing: This means that the position of an object in the list as well as the length of the latter play a crucial role though they should not all all. More than avoiding these issues, the second approach enforces invariance to input length as well as arrangement of objects in the list and further allows to define an intuitive network topology appropriate to spatial reasoning.

As presented in the third chapter, a strong quasi-lossless compression format can be used to significantly reduce the number of directed edges in the graph of spatial relations among scene objects. In this case, though the graph length in term of edge count becomes highly variable, the input of the reasoner remains unchanged. Notice that this variation further demonstrates the handicap of the first approach. As far as the second approach is concerned, one simple way to solve this problem consists in introducing background edges: An edge is annotated as background if and only if there should actually be no one under this compression scheme proposed in chapter 3. Unfortunately, though the compact format, for representing the description of spatial relations, eases the system pipeline from the data annotation to model validation phase, it makes the inference and training phase a little bit harder in the sense that it requires the system to compress the graph of spatial relations while operating locally (i.e. pairs of objects). This means that the spatial relation between two objects does not just rely on both objects' features and the scene background anymore, but rather on other objects and spatial relations among them too. A very natural way, to solve this issue while employing the second approach, consists in considering the graph compression as a multi-step problem-solving or sequential reasoning. At the first step, the normal connected graph is computed, where each edge solely depends on its vertices and the scene background. At the second step, the graph obtained at the first step is summarized and the resulting summary is combined with each $i^{th}-$edge E_{i0} to yield a new edge E_{i1}. The second step is repeated until a fully compressed graph is obtained. Notice that this mechanism is very inspired from the work of [170] exposed as state of the art in the second chapter.

Finally come the questions of what constitutes an object, a pair of objects, a graph summary and a scene background. It is known that a node in the graph of spatial relations represents a scene object and a directed edge denotes the spatial relation between its vertices. Figure 4.9 constiitutes a brief illustration of the spatial reasoner's internal working.

Object's Features

It was mentioned earlier that spatial reasoning intuitively requires $6D$-poses and appearances of the scene objects together with the scene background. The same intuition suggests that by inputting two scene objects together with the scene background, the spatial reasoner should be able to determine the spatial relations between them. Moreover, it has been argued that the

dimension of an object's feature maps is too high for the object's feature maps to constitute the inputs of a strongly emergent system (i.e. MLP(s)) such as the spatial reasoner. The curse of dimensionality was then addressed by projecting the feature maps onto a lower-dimensional space and the resulting feature maps were called DNA codes. An object's DNA code, as explained earlier, contains an implicit $6D$-pose of the object and its appearance. Note that due to numerous convolutions (local) successively performed on the scene image's feature maps, an object's feature maps does not only include its appearance, but rather also involve its direct neighbourhood's appearance, which is crucial in discovering contact spatial relations (e.g. in, on, near-front, near-left) among objects. This being said, it follows that an object's DNA code is suffiently informative to represent an object in the context of spatial reasoning.

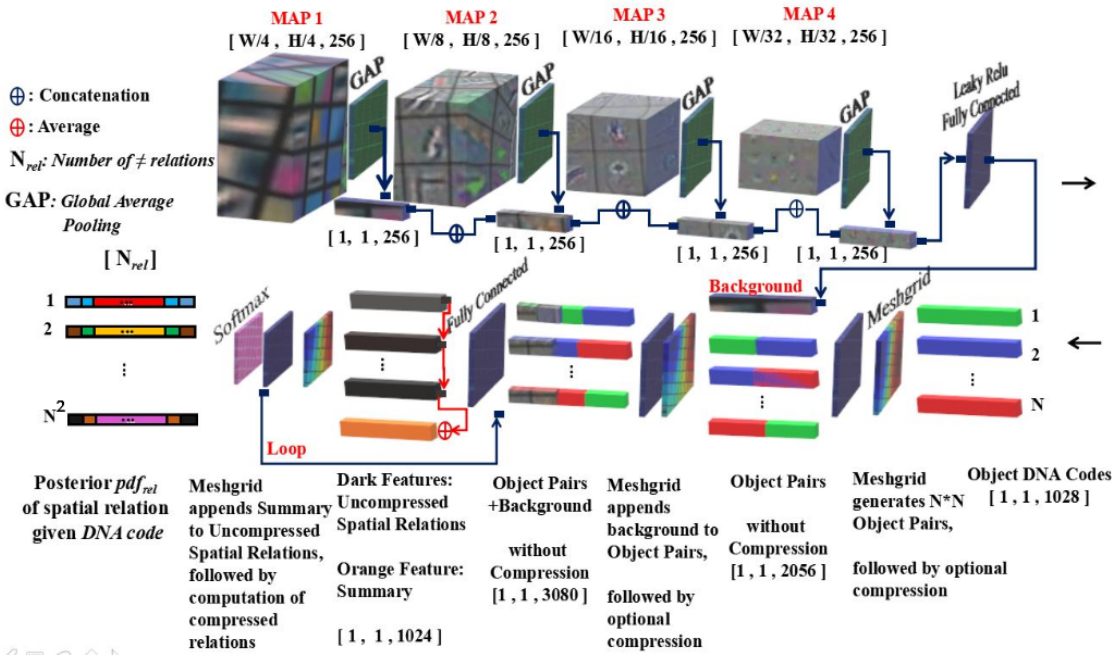


Figure 4.9 Architechture of RobotVQA's Spatial Reasoner

Background Extraction.

As discussed earlier, the DNA codes of two objects are very informative but not enough to fully determine the spatial relations between them. One fundamental reason to this is that spatial relations are invariant to some of the most frequent affine transformations of the scene such as rotations, which $6D$-poses are variant to. If a scene is captured and rotated, while the $6D$-pose will change, the spatial relations will remain unchanged. The spatial reasoner should therefore be able to understand the inclination of the scene in order to acquire this invariance to scene rotation. Naturally, as humans usually do, an inclination of the scene is a very global feature of the scene and can be acquired from a summarization of the scene's generic features such as

edges, corners, etc [176].

The scene's generic features are outputted by the basenet and gathered within a $[W' \times H' \times D']-$dimensional float-array, where the dimensions (W', H') represent the image plane and D' the semantics depth. Since in indoor environments, the scene inclination is deeply related to commonalities in scene objects such as orientations of wall's, table's and bottle's edges, the scene's feature maps are first summarized along the dimensions (W', H') through a Global Average Pooling (GAP) operation to yield a compact global feature map of the scene within a $[D']-$dimensional float-array. This array is subsequently passed to a MLP(s) with leaky relu activation for further reducing the dimension of the global feature vector while preserving its semantics. This last non-linear projection yields a new global feature ($[B]-$dimensional vector) called scene background. From this description, it follows that the background extraction operation is very intuitive and not complex regarding the space, time and parameters required. This operation is very similar to the recent achievement known as Squeeze-Excitation Block proposed by [84].

Pair of objects & Background

In the graph of spatial relations, any edge represents a spatial relation between its vertices. Since spatial relations are directional, the graph edges must be directed. This means that while specifying a spatial relation between two objects, the order of objects in the couple matters. In the context of ANN(s), a very simple way of specifying such a couple as input to a network, is concatenation. Though concatenation preserves the order of object in the couple, it doubles the input dimension. This problem is solved by projecting the concatenation onto a space of much lower-dimension throuh a non-linear perceptron. The same operation is subsequently applied on the pair vector and the background vector to yield a single lower-dimensional vector encoding the triple (original object, destination object, background).

Graph's Edges & Summary.

Given a list of N scene objects, there are $N^2 - N$ triples (original object, destination object, background). Notice that the N reflexive triples (i.e. original object = destination object) were removed. At the heart of the spatial reasoner, a MLP(s) is continuously fed with one such triple at a time till completion of the triple list. This yields a list of $N^2 - N$ graph's edges. The resulting graph is noted as G_0. To summarize this graph G_0, a global sum of its constitutive edges is applied. This summarization is very efficient since it is invariant to the arrangement of edges in G_0, non-parameterized and capable of information preservation (e.g. content addressable memory) [137, 16]. Then, each edge in G_0 is merged, as done above (i.e. concatenation + MLP(s)), with the resulting graph summary S_0 and subsequently passed to a MLP(s) to yield a compressed version G_1 of G_0. This compression step is then iterated as long as desired to yield a much more compressed version G_n of G_0. Finally, each edge in G_n is passed to a softmax layer to output the name of the corresponding spatial relation.

4.4 Training Design

It has been shown that training DL-based architectures is a tricky task. In this section, details are given on how such a challenging task was achieved.

Loss

RobotVQA, as depicted by figure 4.1 and further explained in the previous sections, is a multi-task learner. Earlier, it has been given the benefits offered by multi-tasking, however not regarding the learner's accuracy. As well indicated by [163], multi-tasking can be regarded as a strong implicit learning regularizer, since it prevents one sublearner (i.e. learner for a subtask) to overfit the training dataset while computing a compact feature maps of the scene that must satisfy all sublearners and be therefore exempted from noise. Furthermore, multi-tasking enables an effective collaboration between sublearners that consequently yields consistent global results (e.g. scene graph) and improved local results (e.g. properties). It can be enforced by introducing a multi-loss (i.e. model's global loss) computed as a weighted sum of individual losses, where each subtask's loss is assigned a representative weight in the multi-loss computation [4.1]. Weighting is very important because it allows to define some priorities over the set of subtasks. By setting a task priority to zero, this prevents it to participate during the training: This allows to train the system on partially annoted datasets and consequently on a very broad range of datasets.

Another important fact is the assignment of the weight to the spatial reasoning subtask. Since, due to compression, the resulting graph of spatial relations should usually be a sparse graph (i.e. $\mathcal{O}(N)$ instead of $\mathcal{O}(N^2)$ edges), the loss contribution of background edges would become so important that the system would tend to output empty graph (i.e. very unbalanced training dataset). To address this issue, the spatial reasoning subtask's loss weight is shared between the foreground edges' loss and background edges' loss. The more some edges' loss contributes to the spatial reasoning subtask's loss, the smaller their weight [4.4].

For training the classifiers, the mean cross entropy loss or mean Kullback–Leibler divergence is used as in Mask R-CNN to minimize the mean error contained in the probability distribution function returned by the sofmax layers [4.2, 4.3]. This loss is very generic and robust since it aims at minimizing the error between probability distributions rathen than the simple euclidean distance (i.e. not so meaningful) between labels. Regressors, in contrast to classifiers, are optimized using the mean smooth-L1 loss as in Mask R-CNN too. This loss leverages the robustness of the mean L1-loss (i.e. mean of absolute errors does not vanish and is outlier-insensitive) while acquiring the differentiability of the mean L2-loss (i.e. mean of squarre errors is differentiable) [4.5]. Formally, given a scene image $\mathcal{I}$ containing N scene objects:

$$
\begin{aligned}
&(\mathcal{W}, b), && \text{parameters of RobotVQA}\\
&\mathcal{C} = \{cat, sha, mat, opn, col, rel, msk\}, && \text{set of information inferred through classification}\\
&\mathcal{R} = \{bbox, pos\}, && \text{set of information inferred through regression}\\
&\mathcal{L} = \sum_{i \in (\mathcal{C} \cup \mathcal{R})} \alpha_i \times \mathcal{L}_i, && \alpha_i \in \mathbb{R}^+, \text{ global loss as weighted sum of individual losses}\\
&(\mathcal{W}_h, b_h) = \arg\min_{(\mathcal{W}, b)} \mathcal{L}, && \text{Learning phase}
\end{aligned}
\tag{4.1}
$$

$$
\begin{aligned}
&\mathcal{S}_i, && \text{is the value domain of property } i \in \mathcal{C} \setminus \{rel, mask\}\\
&\mathcal{Y}_i, \hat{\mathcal{Y}}_i \in [0,1]^{|\mathcal{S}_i| \times N}, && \text{are the inferred and expected propability}\\
& && \text{distributions of } i \text{ over the input scene image } \mathcal{I}\\
&\mathcal{L}_i = D_{KL}(\hat{\mathcal{Y}}_i \| \mathcal{Y}_i) = -\sum_j \frac{1}{N} \hat{\mathcal{Y}}_{ij} \log \mathcal{Y}_{ij}, && \text{Kullback-Leibler Divergence}
\end{aligned}
\tag{4.2}
$$

$$
\begin{aligned}
&\mathcal{S}_i, && \text{is the size of property } i \in \{mask\}\\
&\mathcal{Y}_i, \hat{\mathcal{Y}}_i \in [0,1]^{2 \times \mathcal{S}_i \times N}, && \text{are the inferred and expected propability}\\
& && \text{distributions of } i \text{ over the input scene image } \mathcal{I}\\
&\mathcal{L}_i = D_{KL}(\hat{\mathcal{Y}}_i \| \mathcal{Y}_i) = -\sum_j \frac{1}{N\mathcal{S}_i} \hat{\mathcal{Y}}_{ij} \log \mathcal{Y}_{ij}, && \text{Kullback-Leibler Divergence}\\
&2 \text{ values for each of the } S_i \text{ pixels}, && \text{since binary mask.}
\end{aligned}
\tag{4.3}
$$

$$
\begin{aligned}
&\mathcal{S}_i, && \text{is the value domain of property } i \in \{rel\}\\
&\mathcal{Y}_i, \hat{\mathcal{Y}}_i \in [0,1]^{|\mathcal{S}_i| \times N^2}, && \text{are the inferred and expected propability}\\
& && \text{distributions of } i \text{ over image } \mathcal{I}\\
&\hat{\mathcal{Y}}_i = [\hat{\mathcal{Y}}_i^{(1)}, \hat{\mathcal{Y}}_i^{(2)}], && \hat{\mathcal{Y}}_i^{(1)}, \hat{\mathcal{Y}}_i^{(2)} \text{ expected back, for edges}\\
&\mathcal{Y}_i = [\mathcal{Y}_i^{(1)}, \mathcal{Y}_i^{(2)}], && \mathcal{Y}_i^{(1)}, \mathcal{Y}_i^{(2)} \text{ inferred back, for edges}\\
&\mathcal{L}_i = \gamma D_{KL}(\hat{\mathcal{Y}}_i \| \mathcal{Y}_i) + (1-\gamma) D_{KL}(\hat{\mathcal{Y}}_i^{(2)} \| \mathcal{Y}_i^{(2)}), && \gamma \in [0,1], \text{ weighted } D_{KL}
\end{aligned}
\tag{4.4}
$$

$$\begin{aligned}
&\mathcal{S}_i, && \text{domain of property } i \in \mathcal{R}\\
&\mathcal{Y}_i, \hat{\mathcal{Y}}_i \in [0,1]^{4N}, && \text{are the regressed and expected}\\
& && i \text{ over the input scene image } \mathcal{I}\\
&\mathcal{L}_i^{(1)} = |\hat{\mathcal{Y}}_i - \mathcal{Y}_i|, && \text{L1-Loss for } N \text{ objects.}\\
&\mathcal{L}_i^{(1)} = \frac{1}{4N}\sum_j \frac{1}{2}(\mathcal{L}_{ij}^{(1)} < 1)(\mathcal{L}_{ij}^{(1)})^2 + (\mathcal{L}_{ij}^{(1)} \geq 1)(\mathcal{L}_{ij}^{(1)} - \frac{1}{2}), && \text{mean Smooth L1-Loss for}
\end{aligned} \tag{4.5}$$

Learning Rate & Regularization

As already defined in the second chapter however intuitively, the learning rate technically expresses the proportion of an inference error that should contribute to updating the model parameters. Whereas bigger learning rates accerate training but easily exposes to divergence or local extrema, smaller learning rates are learning-convergence-safe but considerably slow down, not to say stop, the training process. Moreover, DL-based models are densily parameterized and before actually learning, they tend to find any vuneralbility point, they can rely on, to overfit the training dataset. For addressing these issues, regularization techniques are used namely the momentum, adaptive learning rate, weight decay, gradient norm clipping, stochastic sampling and pixelwise augmentation of training images.

Massively Thread-safe Parallelized Data Loading

To feed the model with training samples, a massively thread-safe parallel algorithm, that runs on Central Processing Unit(s) (CPU(s)) and loads the training samples from hard disk to GPGPU(s)'s memory, is built. During a training epoch (i.e. complete pass through the whole training dataset), all the training data points are progressively and parallelly loaded into GPGPU(s)'s memory and sequentially passed to the model. This parallelism considerably accelerates the training process and the acceleration is directly proportional to the number of available CPU(s).

Heterogeneous-Dataset-enabled Training

Since real datasets can only very difficultly, not say cannot, present very rich annotations such as synthetic datasets do, as shown in chapter 3, the proposed model has been designed to allow training from different datasets. That is, the system can be trained with a real dataset only destinated to object categorization by setting the weight of the object category classifier's loss to 1 and others to 0 so that the former contributes alone to the model's multi-loss. In this way, the other subtasks are hidden. However, by hidding themselves, they unconsciously return a satisfaction acknowledgement to the updater of model weights. This action can significantly compromise their performances given that the updater of model weights will start to overfit the training dataset on the object categorization task: Collaboration among sublearners is paradoxi-

cally inhibited in multi-tasking. This issue is avoided by freezing all the model layers during the model training except the category classifier. Notice however that while traing on heterogeneous datasets, layer freezing can be relaxed if the different sublearners are permanently activated at least in a round robin manner so that the penalization, mentioned above, does not degrade the learning process, but rather acts as a regularization process.

4.5 Implementation

It has been shown that the proposed model is greatly inspired by Mask R-CNN [75]. Therefore, to provide an implementation of the proposed model that is as efficient, error-free and not time-consuming as possible, the implementation proposed by Mask R-CNN was forked from [76] and extended as RobotVQA in [100]. Another great motivation of this reliance on the implementation of Mask-RCNN is the fact that the latter was achieved by using state-of-the-art frameworks destinated to DL-based applications. The above points altogether significantly ease the extension of Mask R-CNN, deployment and usage of RobotVQA.

Programming Language & Frameworks

Python (version 2.7) is by far the most used programming language in the field of Machine Learning for its flexibility, ease to read and ability to cope (i.e.higher abstraction) with typical ML-operations. For building, training, running and visualizing DL-based models, both on GPGPU(s) and CPU(s), **Tensorflow (version 1.12)** is actually the state-of-the-art framework. RobotVQA is entirely developed on Tensorflow however by interacting importantly with **Keras (version 2.1.6)**, which is a higher-level interface to Tensorflow. It abstracts the complexity of Tensorflow by providing an easily understandable library of common needed functionalities when using Tensorflow.

Mask R-CNN Extension

Since the implemnentation of the proposed model is an extension of MaskRCNN's implementation, this section exposes the extensions achieved.

In contrast to Mask R-CNN, RobotVQA introduces multimodal inputs. It operates on RGB-images as well as RGBD-images. The implementation of RobotVQA provides a mechanism to handles this feature. Furthermore, the Object's DNA code proposed by Mask R-CNN does not intuitively include sufficient global features to ease the discovery of spatial relations between pairs of objects. To address this issue, the implementation of RobotVQA extends the object's DNA code in Mask R-CNN by introducing the object's $(X, Y)-$location into it. Additionally, while Mask R-CNN's implementation limits the description of object to category, bounding box and segmentation mask, the current implementation extends it to pose, material, color, openability

and shape. Another original contribution to Mask R-CNN as well as to its implementation is the introduction of a spatial reasoner that describes the spatial relations among scene objects.

As far as the dataset is concerned, as indicated several times in previous sections, online data augmentation is crucial for a learning regularization. In RobotVQA's implementation, a mechanism was proposed to pixelwise augment the training data points (i.e. images) online (i.e. during data loading in training phase). Moreover, training deep learners, especially on extremly large datasets, is coniderably time-consuming (i.e. months) and Mask R-CNN, being aware of this, proposed in its implementation a mechanism to parallel data loading. However, their proposal is neither thread-safe (i.e. dead-lock), nor redundance-free (i.e. during an epoch, load the same image several times and miss to load some images). While thread-unsafety might stop the training process, data redundance leads to inconsistent and unstable learning. These issues are fixed in RobotVQA's implementation.

Finally, the model's raw inputs and results, namely the dataset cartography, the training statistics and the inference outputs, are accessible through a package of utilities. The utilities provided by Mask R-CNN's implementation were greatly extended to meet the criteria and format of RobotVQA's outputs and inputs (e.g. scene graphs).

RobotVQA Topology's Statistics

The questions, how many convolution blocks or which DL-related blocks make up RobotVQA's topology, where and how big they are in terms of neuron and parameter count, are answered in the following table 4.1. Notice, that this is a precision of the information delivered by figure 4.1.

Block	Operation	Output Shape	Activation
Input Images (RGBD)	Input-Layer Block	$[640 \times 640 \times 7]$	$[-127.5; 127.5]$
Input Images (RGB)	Input-Layer Block	$[640 \times 640 \times 3]$	$[-127.5; 127.5]$
Feature Extraction	MaxPool2D[K=(3,3);N=64]	{Feature Maps}	Max ($\mathbb{R}$)
K=Kernel Size	MaxPool2D[K=(1,1);N=256]	$[10 \times 10 \times 256]$	Linear ($\mathbb{R}$)
N=Kernel Count	UpSampling2D[K=(2,2);N=768]	$[20 \times 20 \times 256]$	Relu ($\mathbb{R}_+$)
$\{ResNet101 + FPN\}$	Conv2D[K=(3,3);N=35200]	$[40 \times 40 \times 256]$	
	Conv2D[K=(7,7);N=64]	$[80 \times 80 \times 256]$	
	Conv2D[K=(1,1);N=1024]	$[160 \times 160 \times 256]$	

Table 4.1 RobotVQA Topology's Statistics (First part)

Block	Operation	Output Shape	Activation
RPN(s)		$\{Regions/Maps\}$	
K=Kernel Size	Conv2D[K=(3,3);N=512]	$[10 \times 10 \times 8 \times 3]$	Linear ($\mathbb{R}$)
N=Kernel Count	Conv2D[K=(1,1);N=6]	$[20 \times 20 \times 8 \times 3]$	Softmax ($[0;1]$)
	Conv2D[K=(1,1);N=12]	$[40 \times 40 \times 8 \times 3]$	Relu ($\mathbb{R}_+$)
		$[80 \times 80 \times 8 \times 3]$	
		$[160 \times 160 \times 8 \times 3]$	
ROIAlign	Object Feature Pooling	$[20 \times 7 \times 7 \times 256]$	Linear ($\mathbb{R}$)
Object DNA Code	Conv2D[K=(7,7);N=1024]	$[20 \times 1028]$	Relu ($\mathbb{R}_+$)
	Conv2D[K=(1,1);N=1024]		Relu ($\mathbb{R}_+$)
Background Extraction	GlobalMaxPool2D	[1024]	Leaky-Relu ($\mathbb{R}$)
N= Neuron Count/Layer	MLP(s) [L=1;N=1024]		
L= Hidden Layer Count			
Object's Intrinsic Properties	5× MLP(s) [L=1;N_i]	[20×6]	Relu ($\mathbb{R}_+$)
N,N_i= Neuron Count/Layer		[20×7]	Softmax ($[0;1]$)
L= Hiden Layer Count		[20×3]	
$N_i \in \{21, 12, 6, 7, 3\}$		[20×21]	
		[20×12]	
Instance Segmentation	Conv2D[K=(3,3);N=1024]	$[20 \times 21 \times 14 \times 14]$	Relu ($\mathbb{R}_+$)
	Deconv2D[K=(2,2);N=256]		Sigmoid ($[0;1]$)
	Conv2D[K=(1,1);N=1]		
4D-Bounding Box	MLP(s) [L=1;N=21×4]	$[20 \times 21 \times 4]$	Linear ($\mathbb{R}$)
6D-Pose	MLP(s) [L=4;N=1028]	$[20 \times 21 \times 6]$	Linear ($\mathbb{R}$)
	MLP(s) [L=1;N=21×6]		Relu ($\mathbb{R}_+$)
Spatial Relations	MLP(s) [L=1;N=1024]	$[20 \times 20 \times 5]$	Relu ($\mathbb{R}_+$)
	MLP(s) [L=1;N=5]		Linear ($\mathbb{R}$)

Table 4.2 RobotVQA Topology's Statistics (Second part)

4.6 Conclusion

The DL-based model, RobotVQA, proposed to address the PVSURMT has been detailly described in this chapter. Particularly remarkable was the reliance on the state-of-the-art achievements to face the challenges raised by PVSURMT. Transfer learning was exploited to a great extend to ease the training and boost the performance of the model. Massive multi-tasking ensured the speed up of the system on training as well as on inference, the structuredness of the system's outputs, a significant learning regularization and finally a great reductibility and extensibility of the system, making the latter therefore general-purpose and inherent to completeness. At the beginning, the model offers a mechanism to handle multimodal input images, which are then passed to a FPN+ResNet basenet. The latter extracts from the input images a deep hierarchical stack of features called the input scene image's feature maps. The particularity of this basenet is Manifold. First of all, it counts its ability to go deeper in computation while avoiding network-depth-related problems (e.g. vanishing gradient). Moreover, it preserves the spatial distribution of information while extracting the features from the input images, easing therefore spatial reasoning such as instance segmentation. Finally, this basenet is invariant to some common transformations (e.g. scaling, translation, small deformations) of the input images, reducing consequently the theoretical amount of training data points needed. The feature maps from the basenet are then passed to a RPN(s) that explicitly localizes and proposes interesting objects in the input scene image. This explicit localization of interesting objects acts as an attention mechanism. Each object's feature maps are then accurately extracted from the input scene image's feature maps by a special operator called ROIAlign and subsequently compressed into the object's DNA code. Afterwards, each object's DNA code is passed to each subtask's solver (i.e. category, shape,..., spatial relation recognition). However, since instance segmentation requires a preservation of the spatial distribution of information during the extraction of features from the input scene image, each object's feature maps, in contrast to its DNA code which does not possess this property (i.e. preservation), is directly passed to the instance segmenter. Each subtask's solver is essentially either a classifier or regressor. Except the object's $6D$-pose which is regressed due to the infiniteness and continuity of the pose space, all other system outputs are reached through a classification. For object describers (classifiers and regressors) that are unary operators, in the sense that they inherently estimate the properties of a single object, the list of objects of interest is processed sequentially. This prevents a weight count explosion and ensures invariance to the length of the input list of objects as well as to the arrangement of objects in the list. This is likewise for binary operators such as the spatial relation classifier, except that the latter is rather passed a list of object pairs. Since spatial relation discovery requires global observation of the scene, a scene background is extracted by summarizing the image's feature maps and appending it to each object pair. Finally, The training of DL-based models being a tricky task, the proposed model is equiped with a set of mechanisms notably regularizers and data loading parallelism to ease it. The model is implemented by forking and extending Mask

RCNN's implementation, since Mask R-CNN was of a great importance to this work. On the one hand, This ensures that RobotVQA's implementation is as not time-consuming and error-free as possible. On the other hand, Mask R-CNN was implemented using state-of-the-art frameworks destinated to DL-applications namely the programming language Python, the frameworks Tensorflow and its higher-level interface Keras for DL- based model building, running, training and visualization on both CPU(s) and GPGPU(s). In the next chapter, the proposed model RobotVQA is evaluated and a discussion of the results is provided.

Chapter 5

Experimentation

5.1 Introduction

In the previous chapter, the architecture of the deep model RobotVQA, proposed to address the PVSURMT, was clearly presented. This section aims at evaluating RobotVQA based on the implementation provided and under various circumstances in order to make some rational judgements about its performance.

5.2 Setup 1: RGB Mode

RobotVQA handles multimodal inputs namely Red,Blue,Red (RGB) and RGBD images. This section is concerned with the evaluation RobotVQA under the RGB mode.

5.2.1 Scenario 1: Synthetic Data

As clearly stated in the introductory chapter, the primary goal of this thesis can be interpreted as consisting in addressing the PVSURMT in virtual environments based on Deep Learning and then transfer the acquired knowledge to real worlds. Before testing the knowledge transferability from virtual worlds to real worlds, the proposed system RobotVQA is first trained and evaluated in virtual environments, that is with synthetic data. This aims at testing the computability of the PVSURMT under DL-based computational models.

5.2.1.1 Description

Goals

The aims of this test consists in evaluating the computability of PVSURMT under DL-based computational models. That is how good DL-based computational models can track the PVSURMT in terms of input-output correspondance. Note that this test does not necessarily require real data since computability can be tested by translating the underlyng problem (i.e. inputs, outputs of PVSURMT) into a much more abstract question (e.g. language recognition). This is even how computability testing under DL-based computational models have been being carried out in the community. Since RobotVQA is made up of regressors and classifier, several metrics are used to this effect. To measure the performance of the $6D$-pose regressor, the Smooth L1-Loss and L1-Loss are used. As far as classifers are concerned, the confusion matrix notably the acuracy, precision and recall have been chosen for their rich semantics and ease to understand. However, note that on classification problems, precision and recall are only needed beside accuracy in case of imbalanced datasets as it is the case for the spatial relation classifier in this thesis (i.e. background edges extremly dominant over foreground edges).

Settings

RobotVQA is trained on the supercomputer ASUS ROG ZENITH EXTREME of the IAI-Bremen's laboratory. The supercomputer possesses around $1000GB$ of hard disk that serve for saving datasets, RobotVQA's checkpoints as well as RobotVQA's intermediate weights during training. It hosts 32 CPU(s)@(Intel, 4GHz) with $120GB$ of working memory. Besides, 4 GPGPU(s)@(NVIDIA, GeForce GTX 1080 Ti, $166 - 10609$ GFLOPS) with a total amount of $44,8$ GB of GPGPU(s)-memory are incorporated for massively parallel computation. From the 71000 annotated synthetic images acquired from the virtual world, as described in the third chapter, 50790 are randomly and uniformly sampled for the model training, 10105 for the model validation and 10105 for the model testing. As discussed earlier, the training dataset is augmented with around 111 real images acquired from the robot operating environment(i.e. EASE's kitchen) as well as from online publicly available datasets and then manually annoted with LabelMe as well as with a self-implemented software in order to capture some background noise missing in virtual worlds. The other 500 real images were kept for testing. By fixing the maximum number of finally detectable objects to 20, the momentum to 0.9, the batch size to 1, the number of epochs to 230, varying the learning rate between 10^{-2} and 10^{-6}, the weight decay between 10^{-4} and 10^{-8} and clipping the gradient norm between 0 and 20, RobotVQA was trained using the Stochastic Gradient Descent (SGD) approach with online pixelwise image augmentation for approximatetly 3 weeks. However, as shown by figure 5.1, this training was no straight forward task, as it constantly required the revision of RobotVQA's hyperparameters as well as loss functions. In this figure, notice that after having considerably improved, the RobotVQA instaneously perform worse and improves again afterwards: These are revision's checkpoints.

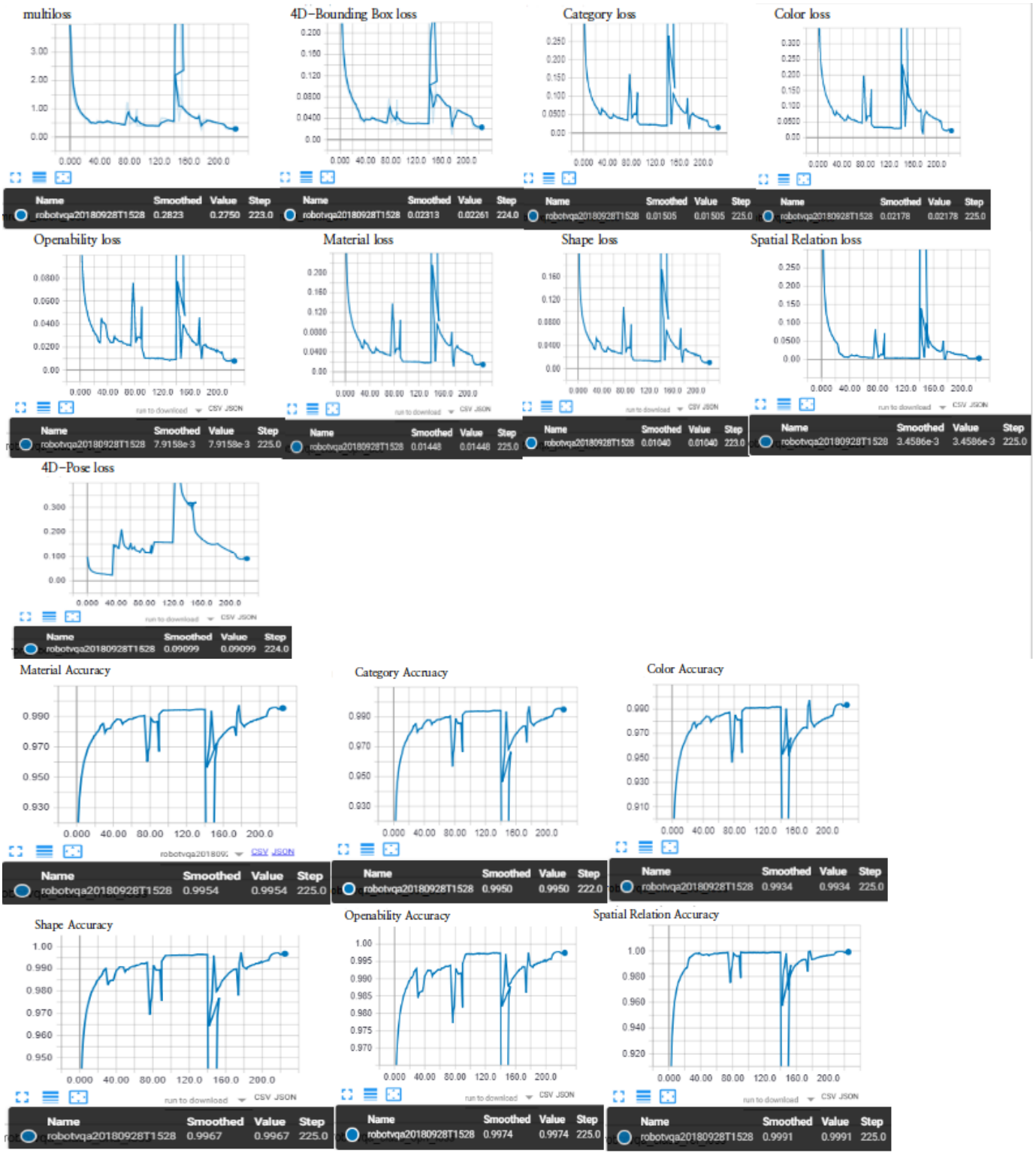


Figure 5.1 Evolution of RobotVQA's performance in terms of loss and accuracy during training. For each graph, the last value is explicitly indicated in the black box.

5.2.1.2 Results

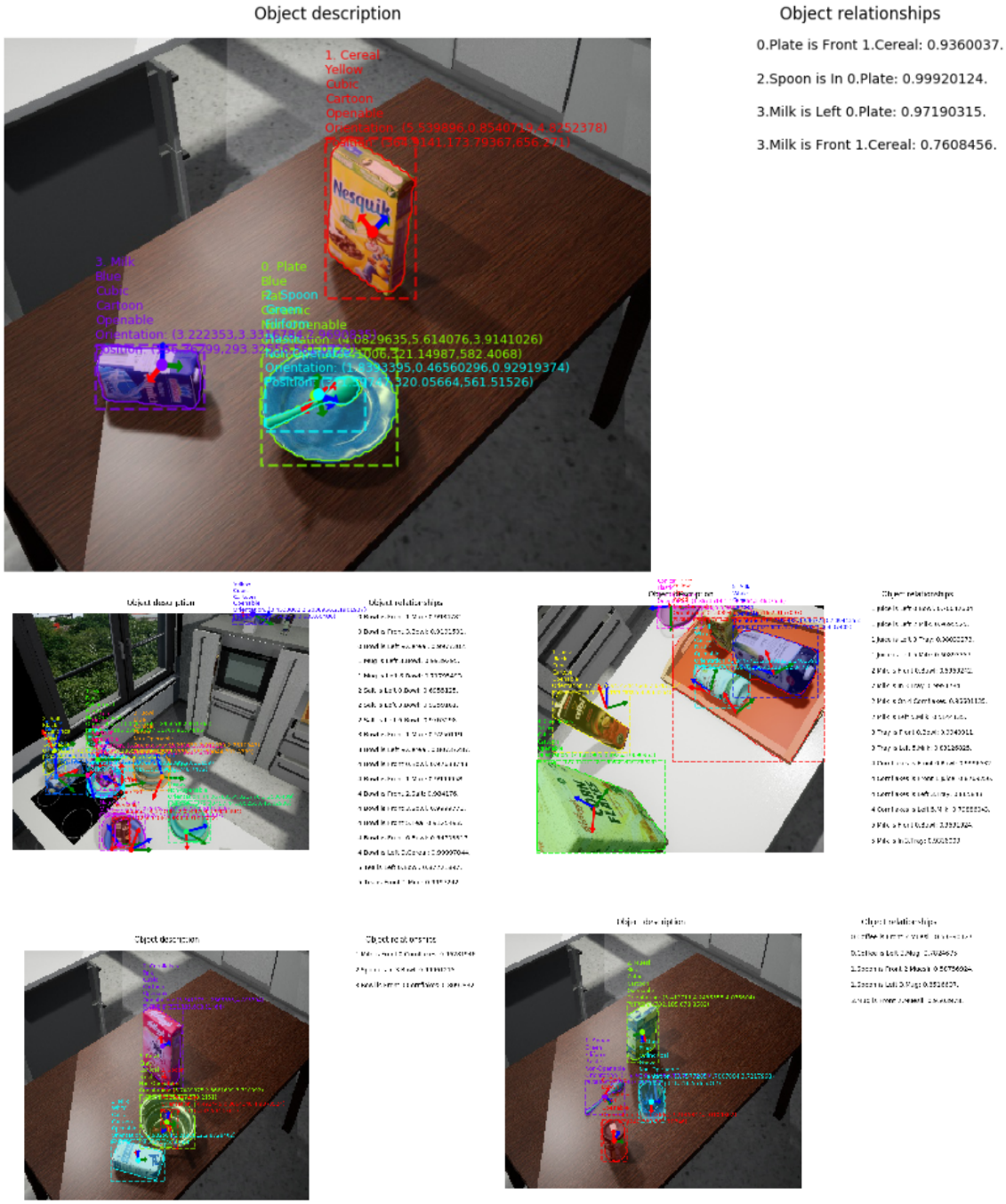


Figure 5.2 Typical RGB Virtual Scenes' Graphs Outputted By RobotVQA

After training and testing RoboVQA, as described above, the results seem to be extremely satisfactory. As shown by table 5.1, all the classifiers show an average accuracy rate near 98.68%. Typically remarkable is on the one hand the accuracy rate of both the color and instance segmentation classifiers respectively of 99.21% and 96.93%, which appear as the lowest. On the other hand, since the spatial relations' super classes (i.e. foreground relations ($\mathcal{O}(N)$) Vs background relations ($\mathcal{O}(N^2)$)) are extremely imbalanced, precision and recall metrics were required. However, instead of naively going for these metrics, a much more efficient way to understand the spatial relation classifier was employed. That is, the classifier was first evaluated on foreground relations where it showed a near perfect accuracy of 99.97% and then on all relations (i.e. background+foreground) where it demonstrates an accuracy of 94.06. The spatial relation classifier is followed by the openability classifier with an accuracy of 99.73%.

Task	Solver	Metric	Value
Category	Classifier	Accuracy	99.47%
Color	Classifier	Accuracy	99.21%
Shape	Classifier	Accuracy	99.64%
Material	Classifier	Accuracy	99.47%
Openability	Classifier	Accuracy	99.73%
Instance Segmentation	Classifier	Accuracy	96.93%
Spatial Relation without Background edges	Classifier	Accuracy	99.97%
Spatial Relation with Background edges	Classifier	Accuracy	94.06%
6D-Pose	Regressor	L1-loss (relative error)	0.128≈12.8%
6D-Pose	Regressor	Smooth L1-loss	0.02825
Depth	Regressor	L1-loss (relative error)	0.0154≈1.54%
4D-Bounding-Box Refinement	Regressor	Smooth L1-loss	0.0398
4D-Bounding-Box Refinement	Regressor	L1-loss (relative error)	0.141≈14.1%
Inference	Timer	Elapsed time	$0.13s$
Image loading+Inference	Timer	Elapsed time	$0.238s$
Space Complexity	OS	Memory	$5.5GB$

Table 5.1 Performance of RobotVQA on RGB Synthetic Data.

As far as the $4D$-bounding box refinement of each detected object is concerned, a L1-loss of 14.1% and smooth L1-loss of 3.98% were recorded. The complementary $4D$-pose $(Z, \Theta_x, \Theta_y, \Theta_z)$, related to each object's depth and orientation, showed a smooth L1-loss of 2.825% and a L1-loss of 12.8% however of only 1.54% on the Z-depth. One more time, note that it is all about the bounding box refinement and not the full bounding box itself, since the full default bounding box was pre-determined earlier by the RPN(s) before a final refinement took place. Finally, an average time of $0.13s$ was reported for each single forward pass or system inference and $0.238s$ including the image loading phase, as well as a spatial complexity of $5.5GB$ GPGPU(s) memory. Figure 5.2 shows some scene graphs derived by RobotVQA from some synthetic scene images.

5.2.2 Scenario 2: Real Data

In this scenario, the knowledge transferability from the virtual world to the real world is tested.

5.2.2.1 Description

Goals

Contrarily to the first scenario, this scenario aims at testing the knowledge transferability in RobotVQA from the virtual world to the real world. That is, how good RobotVQA, which was trained within a virtual world (i.e. with synthetic data), can behave or extrapolate in the real world. The working mode is RGB. The metrics of evaluation are kept unchanged as described in the first scenario.

Settings

The experimental settings are preserved as presented in the first scenario. However, the test dataset consists of around 500 annotated real RGB images, which were, as described in third chapter, acquired from the EASE's kitchen and some publicly available datasets on the internet. As also indicated in that chapter, remember that the real images, due to burdensome manual annotation, were only partially annotated with appearance-related object features such as the object's category, color, openability and material in contrast to the object's pose, shape, bounding box and spatial relations to other objects. But, since those real images were easily annotable with object bounding boxes and shapes, the latter were also part of the annotation. In this case, the bounding box constitutes an approximative semantic mask of the object, which though appearance-related is extremly burdensome to annotate. Notice that this restriction on the object features to evaluate does not bias the transferability test since it is itself more about appearance-related object features rather than depth-related ones for instance.

5.2.2.2 Results

After training and testing RobotVQA, as described above, the results showed a quite similar satisfaction as in the first scenario, however with a slight drop. As shown by table 5.2, all classifiers show an average accuracy near 93.74%: That is a drop of 5% in the accuracy when compared to the first scenario's results. Typically remarkable also is on the one hand the quasi preservation, while moving to real world, of the standard deviation of all these classifiers' accuracies, which is about 1.99% and was around 1.84% in the virtual world. On the other hand, while the color, shape and openability classifiers are significantly heading the accuracy race, contrarily as it was in the first scenario, the category and material classifiers are left back with an average distance of about 3.34%.

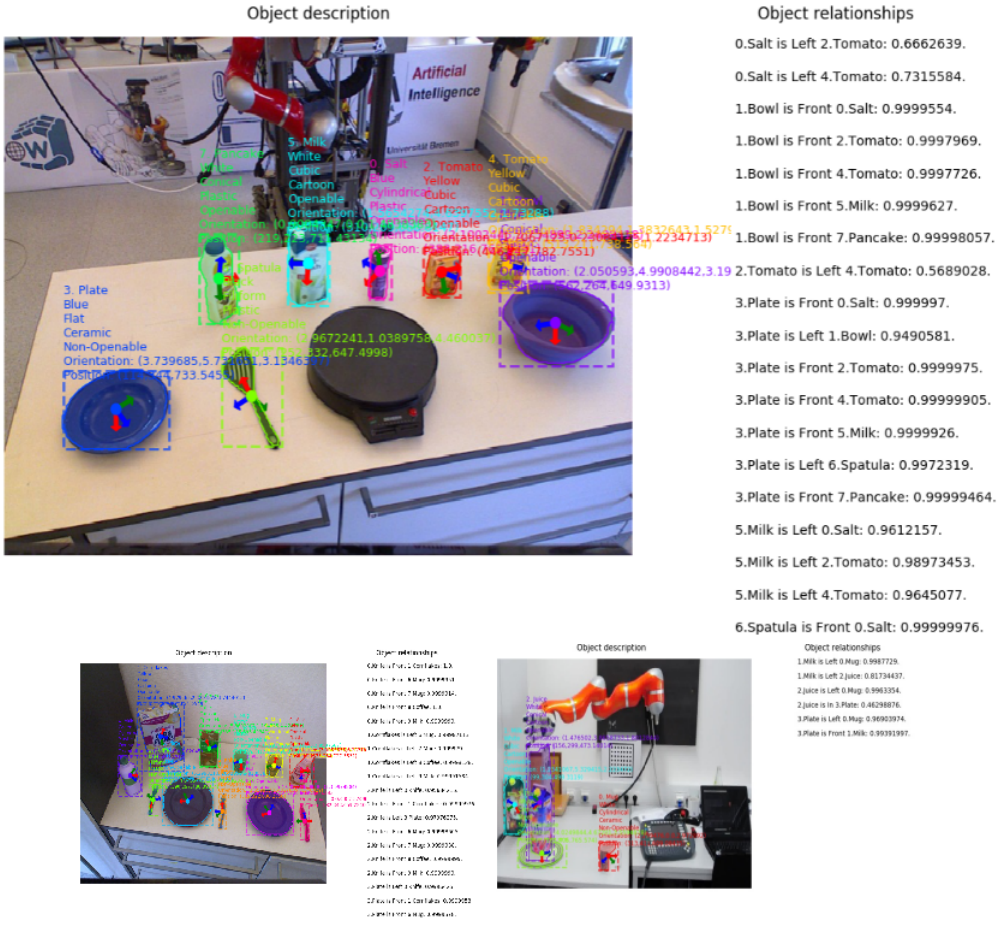


Figure 5.3 Typical RGB Real Scenes' Graphs Outputted By RobotVQA.

As far as the $4D$-bounding box refinement of each detected object is concerned, a quasi doubling of the error rate is observed while moving from the virtual world to the reald world, however

remaining absolutely reasonable. That is, a smooth L1-loss of 9.63% and L1-loss of 23.63% were noted. Finally, an average time of 0.13s was reported for each single forward pass or system inference and 0.147s including the image loading phase, about twice shorter than in the virtual world. Figure 5.3 demonstrates the derivation of a scene graph by RobotVQA from a scene image.

Task	Solver	Metric	Value
Category	Classifier	Accuracy	91.72%
Color	Classifier	Accuracy	93.34%
Shape	Classifier	Accuracy	94.96%
Material	Classifier	Accuracy	91.76%
Openability	Classifier	Accuracy	96.93%
4D-Bounding-Box Refinement	Regressor	Smooth L1-loss	0.0963
4D-Bounding-Box Refinement	Regressor	L1-loss (relative error)	0.2363≈23.63%
Inference	Timer	Elapsed time	0.13s
Image loading+Inference	Timer	Elapsed time	0.147s
Space Complexity	OS	Memory	5.5GB

Table 5.2 Performance of RobotVQA on RGB Real Data.

5.3 Setup 2: RGBD Mode

In this section, RobotVQA is tested under the RGBD mode. The training setup is quite similar as in the previous setup except that RGBD images rather than RGB images are used.

5.3.1 Scenario 1: Virtual Data

In this scenario, the computability of PVSURMT is tested under the RGBD mode.

5.3.1.1 Description

Goals

As seen in the previous setup, the computability of PVSURMT can be evaluated even without real data. That is, synthetic data are enough to achieve the computability test. The test aims at evaluating how computable the PVSURMT is under DL-based computational models while accepting RGBD input images. This test will additionally provide information about the influence of scene's depth information on the computation. As far as the metrics are concerned, they are preserved as described in the first experimental setup.

Settings

The setting of this test is quite similar as the one of the first scenario of the first experimental setup. Since the synthetic dataset, collected and described in chapter 3, contains depth information about each scene image, the training, validation and test datasets are maintained. However, in contrast to the first experimental setup (i.e. RGB mode), the training of RobotVQA under this RGBD mode was not supported by external real dataset due to the constraint imposed by this thesis's completion delay; That is, RobotVQA was only trained with synthetic data. Figure 5.4 demonstrates the evolution of RobotVQA's performance in terms of loss and accuracy during training.

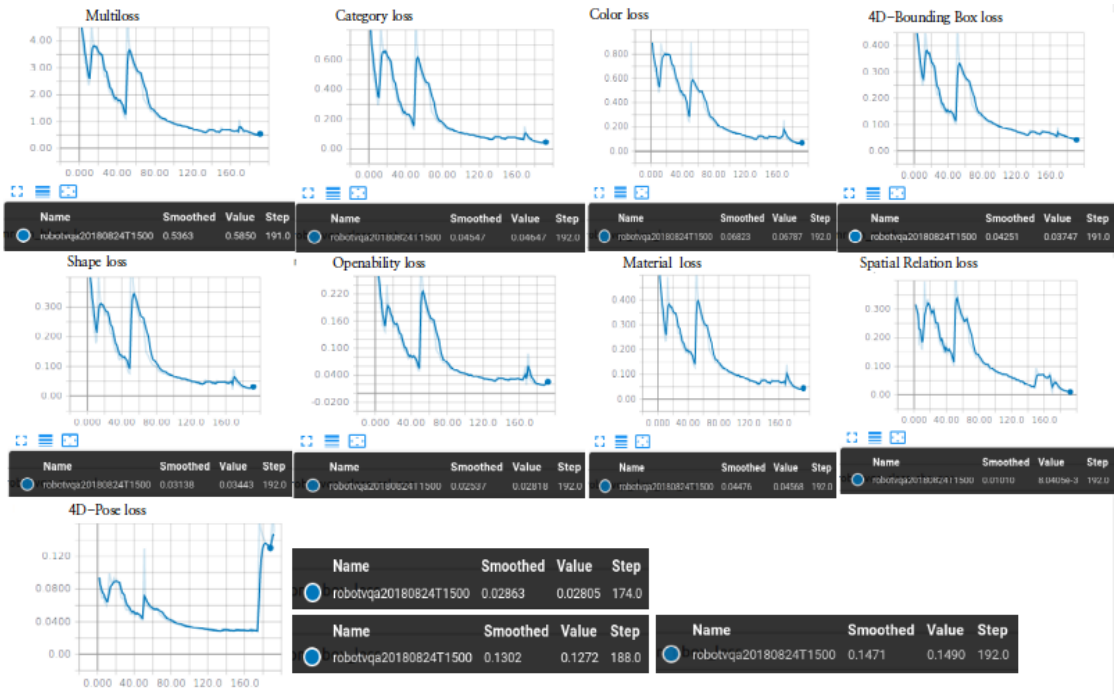


Figure 5.4 Evolution of RobotVQA's performance in terms of loss and accuracy during training. For each graph, the last value is explicitly indicated in the black box. (First part)

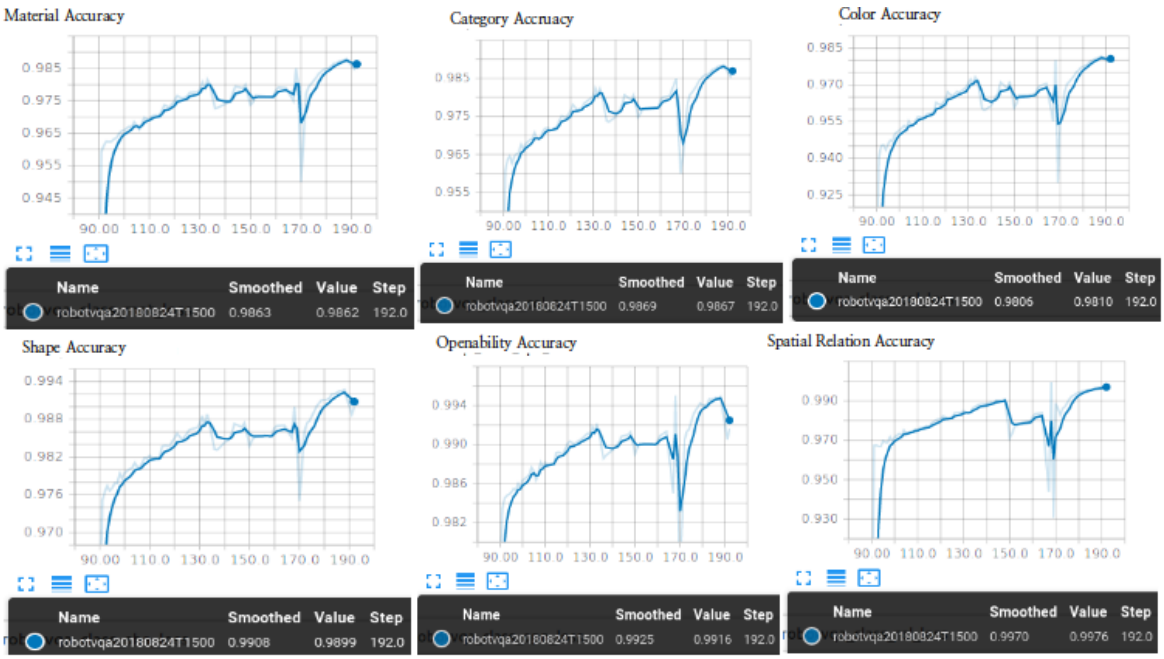
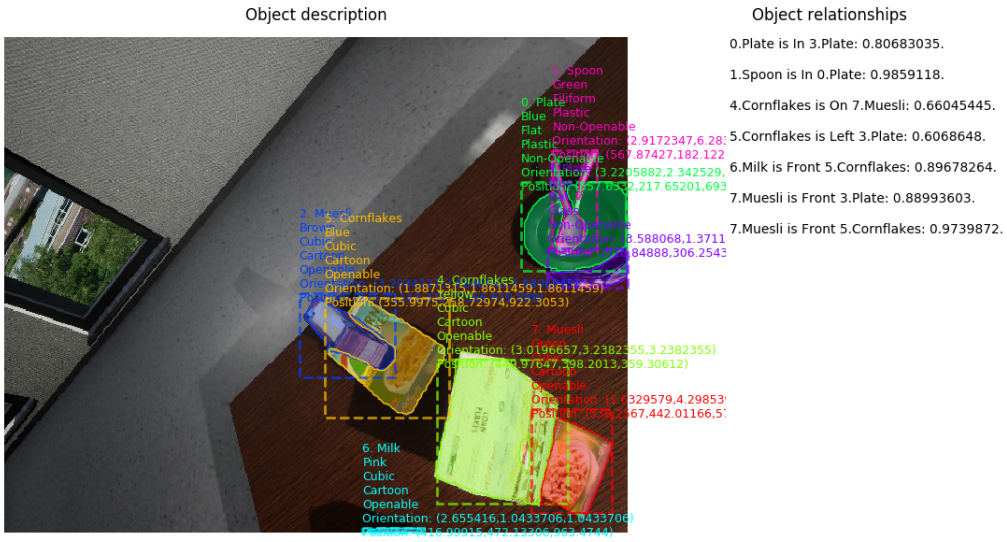


Figure 5.5 Evolution of RobotVQA's performance in terms of loss and accuracy during training. For each graph, the last value is explicitly indicated in the black box. (Second part)

5.3.1.2 Results



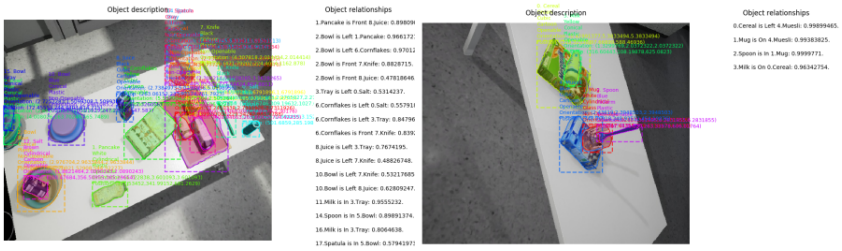


Figure 5.6 Typical RGBD Virtual Scenes' Graphs Outputted By RobotVQA

Task	Solver	Metric	Value
Category	Classifier	Accuracy	99.09%
Color	Classifier	Accuracy	98.56%
Shape	Classifier	Accuracy	99.43%
Material	Classifier	Accuracy	99.08%
Openability	Classifier	Accuracy	99.60%
Instance Segmentation	Classifier	Accuracy	96.76%
Spatial Relation without Background edges	Classifier	Accuracy	99.73%
Spatial Relation with Background edges	Classifier	Accuracy	94.07%
6D-Pose	Regressor	L1-loss (relative error)	0.1386≈13.86%
6D-Pose	Regressor	Smooth L1-loss	0.03315
Depth	Regressor	L1-loss (relative error)	0.0164≈1.64%
4D-Bounding-Box Refinement	Regressor	Smooth L1-loss	0.04798
4D-Bounding-Box Refinement	Regressor	L1-loss (relative error)	0.1668≈16.81%
Inference	Timer	Elapsed time	$0.13s$
Image loading+Inference	Timer	Elapsed time	$0.241s$
Space Complexity	OS	Memory	$5.5GB$

Table 5.3 Performance of RobotVQA on RGBD Synthetic Data.

After training and testing RoboVQA, as described above, the results seem to be extremely satisfactory, however and in contrast to the intuitive expectation with a very slight drop of

about 0.02% in accuracy when compared to the previous setup's first scenario's results. As shown by table 5.3, all classifiers show an average accuracy near 98.66%. Typically remarkable is on the one hand the fact that the accuracy race is headed by depth-based classifer namely the spatial relation and shape classifiers, rather than by appearance-related classifiers as it is the case in the previous setup's first scenario. On the other hand, the spatial relation classifiers show a quite similar aptitude as in RGB mode to compress the graph of spatial relations and recognize valid or foreground spatial relations.

As far as the $4D$-bounding box of each detected object is concerned, a smooth L1-loss of 4.79% was recorded; That is a drop of around 20% in accuracy when moving from RGB mode to RGBD mode. The complementary $4D$-pose $(Z, \Theta_x, \Theta_y, \Theta_z)$, related to each object's depth and orientation, showed a smooth L1-loss of 3.315% and a L1-loss of 13.86%; That is a drop of around 17.34% in accuracy when moving from RGB mode to RGBD mode. However, only a slight drop of 0.65% was observed on the Z-depth's accuracy. Finally, an average time of $0.13s$ was reported for each single forward pass or system inference and $0.241s$ including the image loading phase (i.e. $0.03s$ longer than in RGB mode). This insignificant difference in the time complexity of RobotVQA under both operating modes was also observed in its spatial complexity. Figure illustrates a scene graph outputted by RobotVQA from a virtual scene's RGBD image.

5.3.2 Scenario 2: Real Data

Finally, RobotVQA is tested on RGBD real data.

5.3.2.1 Description

Goals

The goal of this test consists in evaluating RobotVQA on RGBD real images. That is, how good the knowledge acquired by RobotVQA from training on RGBD synthetic images can be useful in understanding RGBD real images (i.e. knowledge transferability). The performance metrics used so far are preserved.

Settings

To carry out this experiment, the setting used in the first setup's second scenario is preserved, since it is about the same test but in RGB mode. Note however one more time that the training was not supported by any external real dataset. Only the synthetic data were used for training.

5.3.2.2 Results

After training and testing RobotVQA, as described above, the results seem satisfactory however with a drop of 4% in the classification's average accuracy when compared to results obtained in the virtual world. Notice that this performance is slightly better than the one demonstrated in RGB mode. As shown by table 5.4, all the classifiers show an average accuracy near 94.44%. Typically remarkable is also the fact that the accuracy race is headed by most of the classifiers, especially the color and openability classifiers showing more than 95.5% accuracy; The exception is the category classifier which lands at the last position with an accuracy of 92.57%.

Task	Solver	Metric	Value
Category	Classifier	Accuracy	92.57%
Color	Classifier	Accuracy	95.14%
Shape	Classifier	Accuracy	94.39%
Material	Classifier	Accuracy	94.02%
Openability	Classifier	Accuracy	96.11%
4D-Bounding-Box Refinement	Regressor	Smooth L1-loss	0.3783
4D-Bounding-Box Refinement	Regressor	L1-loss (relative error)	0.4684≈46.84%
Inference	Timer	Elapsed time	$0.13s$
Image loading+Inference	Timer	Elapsed time	$0.286s$
Space Complexity	OS	Memory	$5.5GB$

Table 5.4 Performance of RobotVQA on RGBD Real Data.

On the other hand, the $4D$-bounding box regressor on each detected object averagely showed an important drop in accuracy when compared to the one obtained in the virtual world. That is, a smooth L1-loss of 37.83% and a L1-loss of 9% were recorded. Finally, an average time of $0.13s$ was reported for each single forward pass or system inference and $0.286s$ including the image loading phase. Figure 5.7 presents a scene graph inferred by RobotVQA from a real scene's RGBD image.

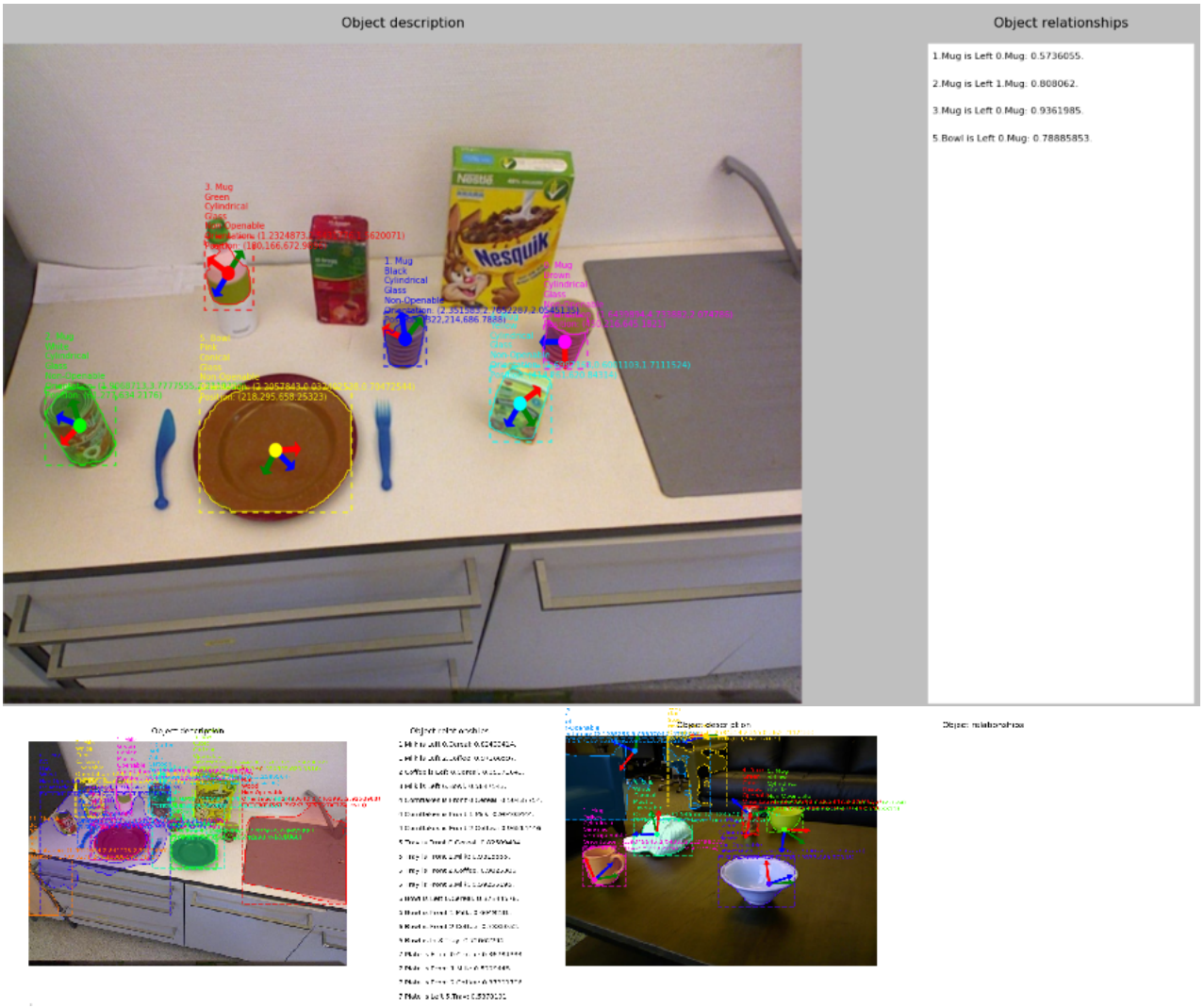


Figure 5.7 Typical RGBD Real Scenes' Graphs Outputted By RobotVQA

5.4 Discussion

This chapter was essentially dedicated to testing RobotVQA, the DL-based computational model proposed to solve the PVSURMT. Four experiments were carried out. On the one hand, the computability of PVSURMT by RobotVQA was evaluated on RGB as well as RGBD synthetic images. On the other hand, the ability of RobotVQA to transfer knowledge from the virtual world to the real world was also tested. That is, RobotVQA was trained on RGB as well as RGBD synthetic images and evaluated on RGB as well as RGBD real images.

While testing RobotVQA on RGB synthetic images, the results suggest that DL-based computational models notably RobotVQA can accurately compute the PVSURMT. However, the color recognizer and instance segmenter showed a slight delay in the accuracy race compared to other classifiers. As far as the color classification is concerned, this might certainly be due to the fact that scene objects are usually multicolor and summarizing those multiple colors into a single color might be a bit tricky for the classifier. More than being most often multicolor, object color annotation happens before the virtual world's simulation. This might lead to extrem confusion if one side of the object, differently colored as with the summary color, is captured during the virtual world's simulation. Regarding the instance segmentation classifier, it has shown an accuracy of 96.93%. Though this performance is worse than for any other classifier, it is an extremly good, not to say perfect, performance for pixelwise segmentation. That is, given the size of scene objects in terms of pixels (e.g. $\approx 64 \times 64$ pixels), such a pixelwise missegmentation rate (i.e. $\approx 3.03\%$) is insignificant (i.e. less than the number of border pixels) and usually due to misannotation or scene noise such as bad illumination. Also important was the spatial relation classifier faced to very imbalanced spatial relations' super-classes notably the the foreground and background relations. Experiments showed that the spatial relation classifier can accurately distinguish background from foreground relations and then foreground sub-classes from each other, making it therefore a good classifier. In such a situation, a bad classifier would indeed either tend to return empty graphs of spatial relations (i.e. only background relations) or show a random behavior faced to background relations (i.e. two much false foreground relations). As far as the regressors are concerned, both the $4D$-bounding box and the complementary $4D$-pose showed similar reasonable losses. However, the bounding boxes are definitvely more accurate than the $4D$-poses, since the former was pre-computed by the RPN(s) and then just refined by the regressor. But notice from the evolution of the training process in figure 5.1 that the $4D$-pose regressor is still converging however very slowly. This slowdown might have several explanations. First and foremost, the very high accuracy observed in the object Z-depth indicates that a great part of the inaccuracy found in the the object $4D$-pose results from the object orientation $(\theta_x, \theta_y, \theta_z)$. Secondly, due to extrem self-symmetry of scene objects (e.g. cylinder, cube), the regressor might be from time to time confused. Thirdly, since RobotVQA's basenet, which is an advanced CNN(s) possessing a couple pose-related invariance properties, the pose regressor, which require pose-related features, might be coming into conflict from time to time with the basenet or with some other submodules such as the spatial reasoner requiring some pose-related invariance. Another plausible explanation of this slowdown in the evolution of the pose regressor learning is the lack of parameters in the regressor: This regressor overfitted a small dataset of 200 images with a quasi zero loss. Finally, it was found that the RobotVQA can load and process ≥ 5 and ≤ 8 images per second, each single inference requiring averagely $5.5GB$ of GPGPU(s) memory; This makes RobotVQA realtime and accessible.

Secondly, the ability of RobotVQA to transfer knowledge from the virtual world to the real world under the RGB mode was tested. Though the results were quite satisfactory, a average drop of 5%

was observed in the accuracy of classifiers. With regard to this accuracy drop, it was first noticed that the classifiers' accuracies are very normally distributed around the mean 93.74% with the standard deviation 1.84. Secondly, it is also known but obviously from RobotVQA's architecture that the category classifier constitutes a filter for other classifiers (e.g. there is no color without object a.k.a category). Thirdly, though the construction of the synthetic dataset achieved in chapter 3 tries the best to capture all the important concepts from EASE's real kitchen, there are infinitely many sub-concepts to a concept; For instance, though the virtual kitchen tries to capture the concept of milk, there are infinitely many featured milk boxes. Since the real dataset acquired from the EASE's real kitchen to evaluate RobotVQA faces this problem, this could have caused a drop in the category classifier's accuracy, which in turn could have bottlenecked through filtering other classifiers' accuracy: hence, the general accuracy drop and normal distribution of accuracies. Additionally, a visual inspection of the RobotVQA's outputs revealed that the virtual world seems to have been limited in capturing the real world's background noise and the object material. While the real world's background noise is extremly manifold, the light properties of an object's material, more than being massively emergent and though accurately reproductible in virtual worlds, appear to show some limitations in the determination of the material. That is, though the virtual world can accurately replicate a particular model of light behaviors on an object's surface, there are so many such models for a particular material. However, while collecting the dataset as shown in chapter 3, only the central or standard light model for each material, such as recommended by experts in [88], was adopted. Note that humans usually make use of cues such as the scene context and the haptic sense to address this issue. To address such a weakness at visually recognizing materials, the scene context, as already being done by RobotVQA, could be further exploited and synthetic data, as done in chapter 3, could be further augmented with more and more real data or synthetic noise (i.e. Augmented Reality + Virtual Reality). As far as the $4D$-bounding box regressor is concerned, a quasi doubling of the loss was observed however remaing visually reasonable since the regression takes place just for refining the bounding box predetermined by the RPN(s). Finally, it was found that RobotVQA, in the real world, can load and process ≥ 7 and ≤ 8 images per second; This makes RobotVQA realtime again. Notice that this speed is slightly greater than the one obtained in the virtual world; This is due to the fact that the average speed in the virtual world was computed over a much longer sequence (i.e. large dataset), introducing consequently more external computing overhead due for instance to ressource management: hence, the sign $\geq$.

After testing RobotVQA in RGB mode, the same tests were performed in RGBD mode, that is with scene's depth information. While testing the PVSURMT computability by RobotVQA, that is on synthetic data, in contrast to the intuive expectation (i.e. great improvement because of depth), the results appear to be quite one-to-one similar to, not to say around 0.02% on classification, 0.65% on the Z-depth regression and 19% on other regressions less accurate than, those obtained in RGB mode. This might convey the idea that the common knowledge held that the scene's depth would improve the model's performance is false. However, instead of naively

swallowing such an idea, it is hypothesized that RobotVQA actually leverages the scene's depth, depth leveraging which is unfortunately hidden by some sort of secondary subtle problem. After some inspection of RobotVQA's architecture, it was found two subtle problems with regard to this depth issue. The first problem is the fact that transfer learning from Mask R-CNN to RobotVQA in RGBD mode is not effective as in RGB mode, since the very first layers of RobotVQA in RGBD mode are trained from scracth: The input image's shape of RobotVQA in RGBD mode differs from the one of Mask R-CNN and does not consequently allow a transfer of Mask R-CNN's first layers' weights, which are extremly relevant. More than this scratch training of RobotVQA's first layers in RGBD mode, the number of parameters required by those scratch first layers is quasi tripled: hence, the second subtle problem. These two problems together are sufficiently bad to occlude the contribution of the scene's depth in the performance of RobotVQA. To further check how RobotVQA exploits the scene's depth information, an additional test was performed where an input image was constituted of one scene's depth map and another scene's color map. The result of this test reveals that RobotVQA quasi-completely relies on the scene's depth information to localize and segment the interesting objects of the scene. However, as shown in the next paragraph, though the scene's depth information would enable RobotVQA to gain significant knowledge gain, it makes knowledge transferability difficult, partly due to its extrem contribution and manifold representation format (e.g. single-channel byte-array, multi-channel byte-array, single-channel float-array). Finally, since the RGBD mode requires RobotVQA to load and process extra depth images, the average processing speed of RobotVQA in this mode is reduced to $\geq 4.15 \approx 5$ and ≤ 8 RGBD images per second, each single inference requiring averagely $5.5GB$ of GPGPU(s) memory; This makes RobotVQA realtime and accessible. Notice that the deviation from the computational complexity obtained in RGB mode is almost neglectable: This is due to RobotVQA's architecture that compactly integrates scene depth into the processing pipeline.

Finally, RobotVQA was tested in its ability to transfer knowledge from the RGBD virtual world to the RGBD real world. More than being satisfactory, the results of this test indicate that RobotVQA transfers knowlegde from the virtual world to the real world in RGBD mode better than in RGB mode: A slight improvement of 0.7% was noted in the knowledge transfer's accuracy while moving from RGB to RGBD mode. It is hypothesized that this superiority is due to a relaxation of the bottleneck created by the category classifier around other classifiers: The category classifier relies on the depth-map to improve the detection of objects and consequently relaxes the bottleneck. However, as just noted in the previous paragraph, the results also demonstrate a weakness of RobotVQA at transferring knowledge about the regression of bounding boxes from the virtual to the real world: a L1-loss of about 46.84% was noted; That is, a bounding box refinement is about twice bigger or smaller than the expected refinement (see figure 5.7). This difficulty can be explained in two steps. Firstly, under RGBD mode, RobotVQA relies to a very great extend, not to say completely, on the scene's depth information. Furthermore, the representation format of the scene's depth information is so manifold and camera-dependent that any

two depth-supporting image-based datasets are likely to have different depth format. For these two reasons, any system including RobotVQA is more likely to behave weakly out of its training domain. In Machine Learning, this phenomenon is known as *covariate shift*. In the present case, the training depth format was a 32-bit float format for representing the distance in meters between the camera center and each point of the observed scene, however cast into bytes for compatibility with other image channels (i.e. RGB) as well as for avoiding as much as possible the covariate shift problem. Contrarily, the real images used for testing were recorded under an unsigned byte format, more precisely the standard format for grayscale images. This standard format usually results from a huge compression (e.g. coarse discretization) of the big but precise depth data returned by the camera , which in turn causes a very big loss of information and shifts the distribution of depth maps. Finally, it was found that RobotVQA operates at a rate of ≥ 4 and ≥ 8 RGBD images per second in the real world. Since there is no difference in the size of RobotVQA's inputs while moving from the virtual to the real world, the observed difference in its is merely due to some external computing overhead.

Chapter 6

Conclusion

In this chapter, essential conclusions about the PVSURMT and RobotVQA are drawn and some recommendations for future related works are suggested as well.

6.1 Summary

Thesis's Goals

This thesis aimed at proposing an artificial vision system to solve the PVSURMT, a problem which is very challenging for Household Robotics. Solving this problem requires the artificial vision system to provide in realtime a structured, complete and probabilistic semantics of the robot scene that can allow the robot to effectively and efficiently complete human-scale manipulation tasks. While scene graphs constitute an adequate formalism for representing such a rich robot scene's semantics, Deep Learning, which has revolutionized AI and particularly the field of CV, is continuously impressive, however only difficultly exploitable due to crises of big rich annotated data, required to train DL-based models. In this thesis, a DL-based vision system, RobotVQA, was proposed to address the PVSURMT, while leveraging Virtual Reality for training it.

Training & Testing Datasets

Deep Learning has theoretically as well as empirically proved to be computationally powerful, however requires big rich data for model training, which is rarely accessible in real world. To raise this issue, this thesis virtualized the robot environment with all necessary actors (e.g. robots) and components (e.g. utensils, table) in UE. But for an effective evaluation of RobotVQA, the training and testing environment was restricted to EASE's kitchen and manipulation tasks to cooking activities. Then, a big rich dataset of 71000 annotated images were collected while allowing an external program to interact with EASE's virtual kitchen through an adapted version

of the UE's plugin UnrealCV. Since synthetic images do not incorporate enough scene's background noise (e.g. cables, papers, pictures), the synthetic dataset was augmented with around 610 real images, collected from the real world (i.e. EASE's kitchen, publicly available datasets on internet) and annotated through a self-built software with the support of the free software LabelMe.

Proposed Model

To address the PVSURMT, a DL-based computational model, known as RobotVQA, was proposed. RobotVQA was built on top of the state-of-the-art achievement Mask R-CNN for its leadership in various fundamental CV tasks and strong capability to enable an efficient transfer learning as well as implementation. The system offers a mechanism to accept multimodal inputs namely RGB and RGBD images. A mode bit is used to select the right input mode. From the input image, a deep hierarchical stack of feature maps is extracted by RobotVQA's (FPN+ResNet)-basenet. FPN preserves the spatial distribution of information and explicitly introduces scale-invariance while extracting the features from the input scene image, whereas ResNet allows a deep computation while preventing network-depth-related problems such as the vanishing gradient problem. Then, the scene image's feature maps are passed to an attention mechanism, called the RPN(s), which localizes and pools out the regions/objects of interest. Each object is then parallely described by multiple classifiers and regressors, where each of them is responsible for recognizing a single object property. Because of their continuity and the infiniteness of their value space, the object's $6D$-pose is regressed, while all the other properties (e.g. shape, category) are inferred from a classification. As far as the spatial relations among scene objects are concerned, the scene background is extracted by summarizing the scene image's feature maps using a particular operator known as GAP, subsequently followed by a MLP(s). Afterwards, this scene background is associated with each pair of objects to constitute the input of the spatial relation classifier. This way of processing (i.e. description of object properties and spatial relation) the list of detected scene objects of interest has many advantages. First, it is massively multi-tasking. Secondly, it processes the list of scene objects sequentially: This prevents a weight count explosion in RobotVQA due to weight sharing and makes RobotVQA invariant to the length of the list of objects as well as the arrangement of objects in the list. And finally, it increases the model reductibility and extensibility. At the end, the system returns a scene graph, which is a directed graph whose nodes are descriptions of objects of interest, detected in the input scene image, and directed edges are spatial relations among objects. An object description comprises the object's category, color, material, shape, openability, segmentation mask and $6D$-pose. RobotVQA was implemented by forking and extending the implementation of Mask R-CNN. This encouraged error-free and time-saving coding.

Evaluation

After training RobotVQA on around 51000 synthetic images combined with approximatedly 110 real images, randomly and uniformly drawn from the dataset collected in chapter 3, for around

230 epochs with stochastic gradient descent of batch size 1, it was tested both on synthetic and real images under both RGB as well as RGBD mode.

Foremost, the ability of RobotVQA to compute the PVSURMT was evaluated by testing it on synthetic images. The results were quite satisfactory under both operating modes. However, it was found that under RGBD mode, the influence of scene's depth information is twofold. On the one hand, it slightly enhances the system performance; It was hypothesized that an important part of the scene depth's influence on RobotVQA's performance was occluded by the inefficient transfer learning from Mask R-CNN to RobotVQA: In RGBD mode, RobotVQA trains its capital high-parameterized first layers from scratch. On the other hand, it was revealed that RobotVQA relies to a great extend, not to say completely, on this scene's depth map to localize and segment the objects of interest in the input scene image. Finally, though the $6D$-pose regressor shows a reasonable loss (e.g. highly accurate $3D$-position), its learning process appears to converge very slowly; It was hypothesized that the regressor (i.e. $3D$-orientation) is coming into conflict from time to time with some of the training dataset's and RobotVQA's constraints: the frequent self-symmetry of scene objects and the invariance of some of RobotQA's submodules to certain pose-related features.

The second experimental setup was concerned with the ability of RobotVQA to transfer the knowledge acquired from the virtual world to the real world. This ability was tested by evaluating RobotVQA's performance on real images. While testing under RGB mode, RobotVQA demonstrates a very great performance at dealing with real data but with some difficulties to grasp the very high entropy of the extremly featured real world. Under the RGBD mode, RobotVQA's classifiers slightly leverage the depth-map to avoid as much as possible the appearance-induced high entropy of the real world in order to improve RobotVQA's performance shown in RGB mode. However, the test of the $6D$-pose regressor revealed a significant drop in the performance. This inability to transfer knowledge about the $6D$-pose regression from the virtual to real world under RGBD mode was explained by two facts. First of all, RobotVQA completely relies on the scene's depth map to localize and segment objects of interest. Moreover, the depth map's format is very manifold and camera-dependent. The format of the depth images used to train RobotVQA was completely different from the one of the images used to test it. This phenomenon, well-known as covariate shift in Machine Learning, can be regarded as a type of disembodiement. That is, by testing RobotVQA on images with a depth format very similar to the one of the images used to train it, one would certainly get satisfactory results as well.

As far as the computational complexity of RobotVQA is concerned, an average delay of $0.13s$ per forward pass or inference and $0.226s$ including the image loading phase were noted, where a single inference consists in taking a scene image as input and outputting the corresponding scene graph. Notice that RobotVQA processes individual scene images at a speed (i.e. ≥ 5 and ≤ 8 images/s) close to the one of the conscious human visual system, which operates at approximatedly $7-13$ frames per second. Moreover, it was found that RobotVQA only requires

5.5GB of GPGPU(s) memory in average to complete a single inference, which is very reasonable as space complexity since it lies in the neighboorhood of today's standard computer memory sizes [193].

As shown by the above narration, RobotVQA was designed to address the PVSURMT and has empirically provided evidences for its ability to address it while learning from virtual worlds. It is a deep learner, meaning that it is computationally robust and able of adaptation. Secondly, it is multi-tasking. This implicitly encourages reasoning under uncertainty and regularizes learning. Thirdly, it provides a structured semantics of the scene, which is easily accessible to primitive agents such as robots. Fourthly, it aims at information completeness. It provides robots with a sufficiently broad range of information about the scene that can allow them to efficiently carry out manipulation tasks at different scales. Fifthly, it accepts multimodal input images namely RGB and RGBD images, which increases its portability (a.k.a democratization). Finally, it is realtime and not memory-greedy, which is suitable for realtime systems such as robots. With all these features together, one can conclusively state that RobotVQA constitutes a good solution candidate when it comes to solve the PVSURMT.

6.2 Recommendations

Though RobotVQA has shown a great ability to address the PVSURMT, it has its own drawbacks which may constitute the foundations of future related works. RobotVQA has demonstrated a reasonable behavior faced to computational ressources. However, the democratization of RobotVQA on low-power, extremly realtime and long-time-running isolated machines (e.g. undersea and space robots, autonomous vehicles) remains questionable (i.e. no computational resources, no energy). In these cases, the argument that today's robots are no more standalone but rather computationally supported by supercomputers becomes invalid. Furthermore, notice that even if the augmentation of the target machine's architecture with supercomputers is practical, the resulting setting is unfortunately very expensive and therefore not accessible to some communities. For these reasons, further researches should be done to minimize the computational complexity of RobotVQA or DL-based robot vision systems.

Another interesting but unexplored aspect of the PVSURMT is the analysis of videos rather than individual images. Instead of independently analyzing individual images in videos, the whole video can be analyzed as a sequence of related images. This relation between images in the video sequence can considerably improve the results: A video can be regarded as markov chain of images. However, as intensively argued in this thesis, pure video analysis requires far bigger rich annotated datasets. A failure to meet this requirement would systematically yield a performance drop compared to image analysis.

Moreover, as shown in this thesis, an understanding of the interactions among scene objects is relevant to the PVSURMT. Given that this thesis merely focused on spatial relations, other interesting relations such as membership relations (i.e. has-a) can be explored in future works given much richer robot environments (e.g. fridges with drawer, doors with handles, machines with buttons).

It is obvious that RobotVQA aiming at information completeness provides robots with a sufficiently broad range of information about the scene that allows them to manipulate at different scales. Among those information are the object's shape, segmentation mask and $6D$-pose that enable object grasping. However, prior to grasping, the robot should infer the object's grasp points from these information, which can somtimes be tricky. This grasping operation could be further eased by providing RobotVQA with an additional computational head to output the grasp points of a given object.

Since a single real material (e.g. glass) has shown to be able to react very differently to light, in contrast to the current work that particularly ascribed a standard reaction behavior to each material, future works related to virtualization are encouraged to model various reaction behaviors to or of light for each material. Furthermore, instead of relying on augmentation with real data to incorporate background noise into synthetic datasets, which becomes quickly fastidious as the size of those real data increases, future works on synthetic datasets are encouraged as well to directly model such noise in the virtual world (e.g. Augmented Reality).

Finally, it should not be forgotten that RobotVQA was tested within a restricted domain namely the EASE's kitchen, and though its performance was quite satisfactory, it did not give very much insight into how RobotVQA trained and tested out of this domain would behave. For this reason, though the strong adaptability of deep learners to new domains and the loose bijection among robot operating domains (i.e. manipulation areas look alike) already speak in favor of RobotVQA, any effort of training and testing RobotVQA outside the EASE's kitchen, in order to better appreciate its strengths, is still encouraged.

Appendix A

Appendix

A.1 List of Algorithms

A.2 List of Figures

A.3 List of Tables

A.4 References

[1] William A. Kuperman and Philippe Roux. "Underwater Acoustics". In: Jan. 2014, pp. 157–212. DOI: *10.1007/978-1-4939-0755-7_5*.

[2] Mm Rodríguez Del Águila and A R González-Ramírez. "Sample size calculation." In: *Allergologia et immunopathologia* 42 5 (2014), pp. 485–92.

[3] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars M. Mescheder, Andreas Geiger, and Carsten Rother. "Augmented Reality Meets Computer Vision : Efficient Data Generation for Urban Driving Scenes". In: *CoRR* abs/1708.01566 (2017).

[4] Cockburn Alistair. *Writing Effective Use Cases*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN: 0201702258.

[5] Thotakuri Anuradha, T.Kalyani, Vucha Mahendra, M. C. chinnaiah, and Telagam Nagarjuna. "Survey on Robot Vision: Techniques, Tools and Methodologies". In: *International Journal of Applied Engineering Research* 12 (Sept. 2017).

[6] Arash Ardakani, Carlo Condo, and Warren J. Gross. "Sparsely-Connected Neural Networks: Towards Efficient VLSI Implementation of Deep Neural Networks". In: *CoRR* abs/1611.01427 (2016). arXiv: *1611.01427*. URL: *http://arxiv.org/abs/1611.01427*.

[7] Ludovic Arnold, Sébastien Rebecchi, Sylvain Chevallier, and Hélène Paugam-Moisy. "An Introduction to Deep Learning". In: vol. 1. Jan. 2011, pp. 477–488.

[8] Good Audience. *Using TensorFlow Autoencoders with Music*. Accessed: 2019-03-24. URL: *https://blog.goodaudience.com/10-reasons-every-early-stage-company-needs-a-social-media-manager-995a96780204*.

[9] Nathan Aviezer. "Intelligent Design versus Evolution". In: *Rambam Maimonides medical journal* 1 (July 2010), e0007. DOI: *10.5041/RMMJ.10007*.

[10] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *CoRR* abs/1511.00561 (2015). arXiv: *1511.00561*. URL: *http://arxiv.org/abs/1511.00561*.

[11] Ferenc Balint-Benczedi and Michael Beetz. "Variations on a Theme: "It's a Poor Sort of Memory that Only Works Backwards"". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*. 2018, pp. 8390–8396. DOI: *10.1109/IROS.2018.8594001*. URL: *https://doi.org/10.1109/IROS.2018.8594001*.

[12] Chitta Baral, Olac Fuentes, and Vladik Kreinovich. "Why Deep Neural Networks: A Possible Theoretical Explanation". In: *Constraint Programming and Decision Making: Theory and Applications*. Ed. by Martine Ceberio and Vladik Kreinovich. Cham: Springer International Publishing, 2018, pp. 1–5. ISBN: 978-3-319-61753-4. DOI: *10.1007/978-3-319-61753-4_1*. URL: *https://doi.org/10.1007/978-3-319-61753-4_1*.

[13] Niklas Barkmeyer. "Learning to recognize new objects using deep learning and contextual information". MA thesis. niklas.barkmeyer@tum.de: Technical University of Munich, Sept. 2016.

[14] Emilie Lundin Barse, Håkan Kvarnström, and Erland Jonsson. "Synthesizing Test Data for Fraud Detection Systems". In: *Proceedings of the 19th Annual Computer Security Applications Conference*. ACSAC '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 384–. ISBN: 0-7695-2041-3. URL: *http://dl.acm.org/citation.cfm?id=956415.956464*.

[15] Pooja Battalwar, Janhvi Gokhale, and Utkarsha Bansod. "Infrared Thermography and IR Camera". In: 2015.

[16] William Bechtel and Adele Abrahamsen. *Connectionism and the Mind*. Wiley-Blackwell, 1991.

[17] Michael Beetz. "Cognition-Enabled Autonomous Robot Control for the Realization of Home Chore Task Intelligence". In: *Proceedings of the IEEE* 100 (2012), pp. 2454–2471.

[18] Michael Beetz. *EASE - Everyday Activity Science and Engineering*. Accessed: 2018-11-19. May 2017. URL: *http://ease.informatik.uni-bremen.de/wp-content/uploads/2017/05/ease.pdf*.

[19] Michael Beetz, Ferenc Balint-Benczedi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer, and Zoltan Marton. "RoboSherlock: Unstructured information processing for robot perception". In: May 2015, pp. 1549–1556. DOI: *10.1109/ICRA.2015.7139395*.

[20] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoglu, and Georg Bartels. "KnowRob 2.0 – A 2nd Generation Knowledge Processing Framework for Cognition-enabled Robotic Agents". In: *International Conference on Robotics and Automation (ICRA)*. Accepted for publication. Brisbane, Australia, 2018.

[21] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. "CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei, Taiwan, Oct. 2010, pp. 1012–1017.

[22] Michael Beetz and Kerstin Schill. *EASE - Everyday Activity Science and Engineering*. Accessed: 2018-11-19. May 2017. URL: *http://ease.informatik.uni-bremen.de/wp-content/uploads/2017/05/overview.pdf*.

[23] Michael Beetz, Moritz Tenorth, Dominik Jain, and Jan Bandouch. "Towards Automated Models of Activities of Daily Life". In: *Technology and Disability* 22.1-2 (2010), pp. 27–40.

[24] Mordechai Ben-Ari and Francesco Mondada. "Robots and Their Applications". In: *Elements of Robotics*. Cham: Springer International Publishing, 2018, pp. 1–20. ISBN: 978-3-319-62533-1. DOI: *10.1007/978-3-319-62533-1_1*. URL: *https://doi.org/10.1007/978-3-319-62533-1_1*.

[25] Yoshua Bengio. "Learning deep architectures for AI". In: *Foundations and Trends in Machine Learning* 2.1 (2009). Also published as a book. Now Publishers, 2009., pp. 1–127. DOI: *10.1561/2200000006*.

[26] Lopes Noelnand Ribeiro Bernardete. "Deep Belief Networks (DBNs)". In: *Machine Learning for Adaptive Many-Core Machines - A Practical Approach*. Cham: Springer International Publishing, 2015, pp. 155–186. ISBN: 978-3-319-06938-8. DOI: *10.1007/978-3-319-06938-8_8*. URL: *https://doi.org/10.1007/978-3-319-06938-8_8*.

[27] Hugues Bersini. "Connectionism vs Gofai: A Brief Critical Analysis". In: *Expert Systems in Structural Safety Assessment*. Ed. by Aleksandar S. Jovanović, Karl F. Kussmaul, Alfredo C. Lucia, and Piero P. Bonissone. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 472–493. ISBN: 978-3-642-83991-7.

[28] Simone Bianco, Claudio Cusano, and Raimondo Schettini. "Color constancy using CNNs". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2015), pp. 81–89.

[29] Margaret A. Boden. "GOFAI". In: *The Cambridge Handbook of Artificial Intelligence*. Ed. by Keith Frankish and William M.Editors Ramsey. Cambridge University Press, 2014, pp. 89–107. DOI: *10.1017/CBO9781139046855.007*.

[30] Rodney A. Brooks, Cynthia Breazeal (Ferrell), Robert Irie, Charles C. Kemp, Matthew Marjanović, Brian Scassellati, and Matthew M. Williamson. "Alternative Essences of Intelligence". In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. AAAI '98/IAAI '98. Madison,

Wisconsin, USA: American Association for Artificial Intelligence, 1998, pp. 961–968. ISBN: 0-262-51098-7. URL: *http://dl.acm.org/citation.cfm?id=295240.295935*.

[31] G. Batchelor Bruce and Whelan Paul. *Intelligent Vision Systems for Industry*. Jan. 1997. DOI: *10.1007/978-1-4471-0431-5*.

[32] C. Russell Bryan, Torralba Antonio, Murphy Kevin, and T. Freeman William. "LabelMe: A Database and Web-Based Tool for Image Annotation". In: *International Journal of Computer Vision* 77 (May 2008). DOI: *10.1007/s11263-007-0090-8*.

[33] D. Camilleri and T. Prescott. "Analysing the limitations of deep learning for developmental robotics". In: *Proceedings of the 6th International Conference on Biomimetic and Biohybrid Systems*. Lecture Notes in Computer Science. Stanford, CA, USA: Springer, 2017, pp. 86–94. ISBN: 9783319635361. DOI: *10.1007/978-3-319-63537-8_8*. URL: *https://doi.org/10.1007/978-3-319-63537-8_8*.

[34] Gang Chen. "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation". In: *CoRR* abs/1610.02583 (2016). arXiv: *1610.02583*. URL: *http://arxiv.org/abs/1610.02583*.

[35] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *CoRR* abs/1606.00915 (2016). arXiv: *1606.00915*. URL: *http://arxiv.org/abs/1606.00915*.

[36] Xi Cheng, Bohdan Khomtchouk, Norman Matloff, and Pete Mohanty. "Polynomial Regression As an Alternative to Neural Nets". In: *CoRR* abs/1806.06850 (2018). arXiv: *1806.06850*. URL: *http://arxiv.org/abs/1806.06850*.

[37] Sam Corbett-Davies. "Real-World Material Recognition for Scene Understanding". In: 2013.

[38] N. R. Corby. "Machine Vision for Robotics". In: *IEEE Transactions on Industrial Electronics* IE-30.3 (Aug. 1983), pp. 282–291. ISSN: 0278-0046. DOI: *10.1109/TIE.1983.356739*.

[39] China Daily. *Top 10 industrial robotic companies in the world*. Accessed: 2019-01-12. Nov. 2015. URL: *http://www.chinadaily.com.cn/bizchina/2015-11/19/content_22483256_8.htm*.

[40] Ashish Dandekar, Remmy A. M. Zen, and Stéphane Bressan. "A Comparative Study of Synthetic Dataset Generation Techniques". In: *DEXA (2)*. Vol. 11030. Lecture Notes in Computer Science. Springer, 2018, pp. 387–395.

[41] C. Darwin. *The Origin of Species*. P. F. Collier & Son, 1909. URL: *https://books.google.de/books?id=YY4EAAAAYAAJ*.

[42] "Deliberation for Autonomous Robots". In: *Artif. Intell.* 247.C (June 2017), pp. 10–44. ISSN: 0004-3702. DOI: *10.1016/j.artint.2014.11.003*. URL: *https://doi.org/10.1016/j.artint.2014.11.003*.

[43] Renaud Detry, Jeremie Papon, and Larry Matthies. "Semantic and Geometric Scene Understanding for Task-oriented Grasping of Novel Objects from a Single View". In: 2017.

[44] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian D. Reid. "Deep-6DPose: Recovering 6D Object Pose from a Single RGB Image". In: *CoRR* abs/1802.10367 (2018). arXiv: *1802.10367*. URL: *http://arxiv.org/abs/1802.10367*.

[45] Carl Doersch. "Tutorial on Variational Autoencoders". In: *CoRR* abs/1606.05908 (2016).

[46] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural computation* 18 (Aug. 2006), pp. 1527–54. DOI: *10.1162/neco.2006.18.7.1527*.

[47] EASE. *Everyday Activity Science and Engineering*. Accessed: 2018-11-19. URL: *http://ease-crc.org/*.

[48] EASE. *Open Knowledge for AI-Enabled Robots*. Accessed: 2018-11-19. URL: *http://www.open-ease.org/*.

[49] Aaron Ladd Edsinger. "Robot Manipulation in Human Environments". PhD thesis. MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2007.

[50] EITC. *Computer Vision*. Accessed: 2019-01-14. URL: *http://www.eitc.org/research-opportunities/future-internet-and-quantum-communications/photos/robot-visions-family-tree/image_preview*.

[51] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin A. Riedmiller, and Wolfram Burgard. "Multimodal Deep Learning for Robust RGB-D Object Recognition". In: *CoRR* abs/1507.06821 (2015). arXiv: *1507.06821*. URL: *http://arxiv.org/abs/1507.06821*.

[52] Mustafa Ersen, Erhan Oztop, and Sanem Sariel. "Cognition-Enabled Robot Manipulation in Human Environments: Requirements, Recent Work, and Open Problems". In: *IEEE Robotics and Automation Magazine* 24 (2017), pp. 108–122.

[53] Mark Everingham, S. M. Eslami, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *Int. J. Comput. Vision* 111.1 (Jan. 2015), pp. 98–136. ISSN: 0920-5691. DOI: *10.1007/s11263-014-0733-5*. URL: *http://dx.doi.org/10.1007/s11263-014-0733-5*.

[54] Chen Hua Feng, Yang Xiao, A. Willette, W. W. McGee, and Vineet R. Kamat. "Towards Autonomous Robotic In-Situ Assembly on Unstructured Construction Sites Using Monocular Vision". In: 2014.

[55] Michael Firman. *A list of RGBD datasets*. Accessed: 2019-04-01. URL: *http://www.michaelfirman.co.uk/RGBDdatasets/index.html*.

[56] Roland W. Fleming. "Visual perception of materials and their properties". In: *Vision Research* 94 (2014), pp. 62–75.

[57] Jianlong Fu and Yong Rui. "Advances in deep learning approaches for image tagging". In: *APSIPA Transactions on Signal and Information Processing* 6 (2017), e11. DOI: *10.1017/ATSIP.2017.12*.

[58] Pawel Gajewski, Paulo Abelha Ferreira, Georg Bartels, Chaozheng Wang, Frank Guerin, Bipin Indurkhya, Michael Beetz, and Bartlomiej Sniezynski. "Adapting Everyday Manip-

ulation Skills to Varied Scenarios". In: *CoRR* abs/1803.02743 (2018). arXiv: *1803.02743*. URL: *http://arxiv.org/abs/1803.02743*.

[59] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez-de-Santos, and Manuel Armada. "The evolution of robotics research". In: *Robotics and Automation Magazine, IEEE* 14 (Apr. 2007), pp. 90–103. DOI: *10.1109/MRA.2007.339608*.

[60] Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. "Visual Turing test for computer vision systems." In: *Proceedings of the National Academy of Sciences of the United States of America* 112 12 (2015), pp. 3618–23.

[61] GitBook. *Object Detection*. Accessed: 2019-04-01. URL: *https://apple.github.io/turicreate/docs/userguide/object_detection/*.

[62] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. *http://www.deeplearningbook.org*. MIT Press, 2016, pp. 1–24.

[63] Kristen Grauman and Bastian Leibe. "Visual Object Recognition". In: *Visual Object Recognition*. 2011.

[64] Alex Graves and MarcÁurelio Ranzato. *Unsupervised Deep Learning*. Accessed: 2019-03-21. URL: *https://ranzato.github.io/publications/tutorial_deep_unsup_learning_part1_NeurIPS2018.pdf*.

[65] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines". In: *CoRR* abs/1410.5401 (2014). arXiv: *1410.5401*. URL: *http://arxiv.org/abs/1410.5401*.

[66] Roger Grosse. *CSC321 Lecture 15: Exploding and Vanishing Gradients*. Accessed: 2019-03-21. URL: *http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec15.pdf*.

[67] Manuel Günther, Peiyun Hu, Christian Herrmann, Chi-Ho Chan, Min Jiang, Shufan Yang, Akshay Raj Dhamija, Deva Ramanan, Jürgen Beyerer, Josef Kittler, Mohamad Al Jazaery, Mohammad Iqbal Nouyed, Cezary Stankiewicz, and Terrance E. Boult. "Unconstrained Face Detection and Open-Set Face Recognition Challenge". In: *CoRR* abs/1708.02337 (2017). arXiv: *1708.02337*. URL: *http://arxiv.org/abs/1708.02337*.

[68] Akhand M. A. H., Akash Anik, and S. Mollah A. "Improvement of Haar Feature Based Face Detection in OpenCV Incorporating Human Skin Color Characteristic". In: *International Journal of Computer Applications & Information Technology* 1 (Nov. 2016), p. 8.

[69] Volker Haarslev, Ralf Möller, and Michael Wessel. "The Description Logic ALCNHR+ Extended with Concrete Domains: A Practically Motivated Approach". In: *IJCAR*. 2000.

[70] Andrei Haidu. *ROBCOG - Robot Commonsense Games*. Accessed: 2018-11-22. May 2017. URL: *http://www.robcog.org*.

[71] Liu Han and Cocea Mihaela. "Traditional Machine Learning". In: Jan. 2018, pp. 11–22. ISBN: 978-3-319-70057-1. DOI: *10.1007/978-3-319-70058-8_2*.

[72] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. "SceneNet: Understanding Real World Indoor Scenes With Synthetic Data". In: *CoRR* abs/1511.07041 (2015). arXiv: *1511.07041*. URL: *http://arxiv.org/abs/1511.07041*.

[73] Li Haoxiang, Lin Zhe, Shen Xiaohui, Brandt Jonathan, and Hua Gang. "A convolutional neural network cascade for face detection". In: June 2015, pp. 5325–5334. DOI: *10.1109/CVPR.2015.7299170*.

[74] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. "FuseNet: Incorporating Depth into Semantic Segmentation via Fusion-Based CNN Architecture". In: *ACCV*. 2016.

[75] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: *1703.06870*. URL: *http://arxiv.org/abs/1703.06870*.

[76] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. *Mask R-CNN for Object Detection and Segmentation*. Accessed: 2019-02-23. URL: *https://github.com/matterport/Mask_RCNN*.

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: *1512.03385*. URL: *http://arxiv.org/abs/1512.03385*.

[78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: *1512.03385*. URL: *http://arxiv.org/abs/1512.03385*.

[79] D.O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. A Wiley book in clinical psychology. Wiley, 1949. ISBN: 9780471367277. URL: *https://books.google.de/books?id=dZ0eDiLTwuEC*.

[80] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiri Matas, and Carsten Rother. "BOP: Benchmark for 6D Object Pose Estimation". In: *CoRR* abs/1808.08319 (2018). arXiv: *1808.08319*. URL: *http://arxiv.org/abs/1808.08319*.

[81] Matej Hoffmann and Rolf Pfeifer. "Robots as Powerful Allies for the Study of Embodied Cognition from the Bottom Up". In: *CoRR* abs/1801.04819 (2018). arXiv: *1801.04819*. URL: *http://arxiv.org/abs/1801.04819*.

[82] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2 (1989), pp. 359–366.

[83] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In: *CoRR* abs/1709.01507 (2017). arXiv: *1709.01507*. URL: *http://arxiv.org/abs/1709.01507*.

[84] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In: *CoRR* abs/1709.01507 (2017). arXiv: *1709.01507*. URL: *http://arxiv.org/abs/1709.01507*.

[85] Guang-Bin Huang, Qin-Yu Zhu, and Caroline Teoh Sheu Siew. "Extreme learning machine: a new learning scheme of feedforward neural networks". In: *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)* 2 (2004), 985–990 vol.2.

[86] IAI-Bremen. *Robots in our Lab*. Accessed: 2019-01-12. URL: *http://ai.uni-bremen.de/research/robots*.

[87] IAI-Bremen. *Welcome To RoboSherlock*. Accessed: 2019-01-12. URL: *http://robosherlock.org/_images/pr2-looking_2_small.jpg*.

[88] Epic Games Inc. *Materials*. Accessed: 2019-02-28. URL: *https://docs.unrealengine.com/en-US/Engine/Rendering/Materials*.

[89] Félix Ingrand and Malik Ghallab. "Deliberation for autonomous robots: A survey". In: *Artificial Intelligence* 247 (2017). Special Issue on AI and Robotics, pp. 10–44. ISSN: 0004-3702. DOI: *https://doi.org/10.1016/j.artint.2014.11.003*. URL: *http://www.sciencedirect.com/science/article/pii/S0004370214001350*.

[90] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: *1502.03167*. URL: *http://arxiv.org/abs/1502.03167*.

[91] Kayoko Ishii. "Cognitive-Robotics to Understand Human Beings". In: *Cognitive Studies* 14.1 (2007), pp. 11–30. DOI: *10.11225/jcss.14.11*.

[92] Deng J., Dong W., Socher R., Li L.-J., Li K., and Fei-Fei L. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[93] S. Jeschke, R. Schmitt, and A. Dröge. *Exploring Cybernetics: Kybernetik im interdisziplinären Diskurs*. Springer Fachmedien Wiesbaden, 2015, pp. 66–66. ISBN: 9783658117559. URL: *https://books.google.de/books?id=jWMiCwAAQBAJ*.

[94] Malik Jitendra, Arbeláez Pablo, Carreira João, Fragkiadaki Katerina, Girshick Ross, Gkioxari Georgia, Gupta Saurabh, Hariharan Bharath, Kar Abhishek, and Tulsiani Shubham. "The Three R's of Computer Vision". In: *Pattern Recogn. Lett.* 72.C (Mar. 2016), pp. 4–14. ISSN: 0167-8655. DOI: *10.1016/j.patrec.2016.01.019*. URL: *https://doi.org/10.1016/j.patrec.2016.01.019*.

[95] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. "Image retrieval using scene graphs". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 3668–3678.

[96] Bernhard Jung. "Embodying and Situating Cognitive Vision What Cognitive Vision can learn from Embodied and Situated Cognition". In: 2005.

[97] Andrej Karpathy. *Convolutional Neural Networks for Visual Recognition*. Accessed: 2019-03-17. URL: *http://cs231n.github.io/neural-networks-1/*.

[98] Charles C. Kemp and Aaron Edsinger. "Robot Manipulation of Human Tools : Autonomous Detection and Control of Task Relevant Features". In: 2006.

[99] Charles C. Kemp, Aaron Edsinger, and Eduardo Torres-Jara. "Challenges for robot manipulation in human environments". In: *IEEE Robotics and Automation Magazine* 14 (2007), pp. 20–29.

[100] Franklin Kenghagho Kenfack. *RobotVQA: Scene-graph-oriented Visual Scene Understanding for Complex Robot Manipulation Tasks based on Deep Learning Architectures and Virtual Reality*. Accessed: 2019-02-23. URL: *https://github.com/fkenghagho/RobotVQA*.

[101] David Kriesel. *A Brief Introduction to Neural Networks*. 2007, pp. 71–104. URL: *http://www.dkriesel.com*.

[102] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: *10.1145/3065386*.

[103] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105. URL: *http://dl.acm.org/citation.cfm?id=2999134.2999257*.

[104] Lars Kunze, Chris Burbridge, Marina Alberti, Akshaya Thippur, John Folkesson, Patric Jensfelt, and Nick Hawes. "Combining top-down spatial reasoning and bottom-up object class recognition for scene understanding". In: *2014 13th International Conference on Control Automation Robotics and Vision (ICARCV)* (2014), pp. 1528–1535.

[105] Weser Kurier. *Roboter hilft in der Kueche*. Accessed: 2019-03-12. URL: *https://www.weser-kurier.de/bremen/bremen-stadt_artikel,-Roboter-hilft-in-der-Kueche-_arid,511394.html*.

[106] P.A. Laplante. *Real-Time Systems Design and Analysis*. Wiley, 2004. ISBN: 9780471648284. URL: *https://books.google.de/books?id=1qDuaC17yxAC*.

[107] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. "An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation". In: *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. Corvalis, Oregon, USA: ACM, 2007, pp. 473–480. ISBN: 978-1-59593-793-3. DOI: *10.1145/1273496.1273556*. URL: *http://doi.acm.org/10.1145/1273496.1273556*.

[108] Cecilia Laschi, Paolo Dario, Maria Chiara Carrozza, Eugenio Guglielmelli, Giancarlo Teti, Davide Taddeucci, Fabio Leoni, Bruno Massa, Massimiliano Zecca, and Roberto Lazzarini. "Grasping and Manipulation in Humanoid Robotics". In: 2000.

[109] Quoc V. Le. *A Tutorial on Deep Learning, Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks*. Accessed: 2019-03-24. URL: *http://robotics.stanford.edu/~quocle/tutorial2.pdf*.

[110] Ian Lenz, Honglak Lee, and Ashutosh Saxena. "Deep Learning for Detecting Robotic Grasps". In: *CoRR* abs/1301.3592 (2013). arXiv: *1301.3592*. URL: *http://arxiv.org/abs/1301.3592*.

[111] Ling Li and Yaser S. Abu-Mostafa. *Data Complexity in Machine Learning*. Computer Science Technical Report CaltechCSTR:2006.004. Caltech:adr: Caltech, May 2006. URL: *http://resolver.caltech.edu/CaltechCSTR:2006.004*.

[112] Linhui Li, Bo Qian, Jing Lian, Wei-Na Zheng, and Yafu Zhou. "Traffic Scene Segmentation Based on RGB-D Image and Deep Learning". In: *IEEE Transactions on Intelligent Transportation Systems* 19 (2018), pp. 1664–1669.

[113] Shiyu Liang and R. Srikant. "Why Deep Neural Networks?" In: *CoRR* abs/1610.04161 (2016). arXiv: *1610.04161*. URL: *http://arxiv.org/abs/1610.04161*.

[114] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. "Feature Pyramid Networks for Object Detection". In: *CoRR* abs/1612.03144 (2016).

[115] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: *1405.0312*. URL: *http://arxiv.org/abs/1405.0312*.

[116] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single Shot MultiBox Detector". In: *CoRR* abs/1512.02325 (2015). arXiv: *1512.02325*. URL: *http://arxiv.org/abs/1512.02325*.

[117] Thomas Lukasiewicz. "Expressive probabilistic description logics". In: *Artificial Intelligence* 172 (Apr. 2008), pp. 852–883. DOI: *10.1016/j.artint.2007.10.017*.

[118] C. Lutz. "Description Logics with Concrete Domains—A Survey". In: *Advances in Modal Logic 2002 (AiML 2002)*. Final version appeared in Advanced in Modal Logic Volume 4, 2003. Toulouse, France, 2002.

[119] Johnson-Roberson M., Bohg J., Skantze G., Gustafson J., Carlson R., Rasolzadeh B., and Kragic D. "Enhanced visual scene understanding through human-robot dialog". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. 2011, pp. 3342–3348.

[120] Stefano Mafrica. "Bio-Inspired Visual Sensors for Robotic and Automotive Applications". PhD thesis. July 2016, pp. 4–5.

[121] Elias N. Malamas, Euripides G. M. Petrakis, Michalis E. Zervakis, Laurent Petit, and Jean-Didier Legat. "A survey on industrial vision systems, applications, tools". In: *Image Vision Comput.* 21 (2003), pp. 171–188.

[122] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York, NY, USA: Henry Holt and Co., Inc., 1982, pp. 20, 32. ISBN: 0716715678.

[123] Silvetti Massimo and Verguts Tom. "Reinforcement Learning, High-Level Cognition, and the Human Brain". In: Jan. 2012, pp. 283–96. DOI: *10.13140/2.1.1983.3922*.

[124] MathWorks. *Introduction to Deep Learning: What Are Convolutional Neural Networks?* Accessed: 2019-03-23. URL: *https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html*.

[125] Warren S. McCulloch and Walter Pitts. "Neurocomputing: Foundations of Research". In: ed. by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press, 1988. Chap. A Logical Calculus of the Ideas Immanent in Nervous Activity, pp. 15–27. ISBN: 0-262-01097-6. URL: *http://dl.acm.org/citation.cfm?id=65669.104377*.

[126] P. Meerlo, R.M. Benca, and T. Abel. *Sleep, Neuronal Plasticity and Brain Function*. Current Topics in Behavioral Neurosciences. Springer Berlin Heidelberg, 2015. ISBN: 9783662468777. URL: *https://books.google.de/books?id=PPYlrgEACAAJ*.

[127] Qinxue Meng, Daniel R. Catchpoole, David B. Skillicorn, and Paul J. Kennedy. "Relational Autoencoder for Feature Extraction". In: *CoRR* abs/1802.03145 (2018). arXiv: *1802.03145*. URL: *http://arxiv.org/abs/1802.03145*.

[128] Blanca Miller and David Feil-Seifer. "Embodiment, Situatedness, and Morphology for Humanoid Robots Interacting with People". In: *Humanoid Robotics: A Reference*. Ed. by Ambarish Goswami and Prahlad Vadakkepat. Dordrecht: Springer Netherlands, 2016, pp. 1–23. ISBN: 978-94-007-7194-9. DOI: *10.1007/978-94-007-7194-9_130-1*. URL: *https://doi.org/10.1007/978-94-007-7194-9_130-1*.

[129] Peikari Mohammad, Salama Sherine, Nofech-Mozes Sharon, and Martel Anne. "A Cluster-then-label Semi-supervised Learning Approach for Pathology Image Classification". In: *Scientific Reports* 8 (Dec. 2018). DOI: *10.1038/s41598-018-24876-0*.

[130] Luis A. Morgado-Ramirez, Sergio Hernandez-Mendez, Luis F. Marin- Urias, Antonio Marin-Hernandez, and Homero V. Rios-Figueroa. "Visual Data Combination for Object Detection and Localization for Autonomous Robot Manipulation Tasks". In: *Research in Computing Science* (2011).

[131] Onyesolu Moses and Udoka Eze Felista. "Understanding Virtual Reality Technology: Advances and Applications". In: Mar. 2011. ISBN: 978-953-307-173-2. DOI: *10.5772/15529*.

[132] Shah Mubarak. *Fundamentals of Computer Vision*. Orlando, FL: University of Central Florida, Dec. 1997.

[133] Siegfried Nijssen and Uzay Kaymak. *COMPUTATONAL INTELLIGENCE*. Accessed: 2019-03-17. URL: *http://liacs.leidenuniv.nl/~nijssensgr/CI/2011/1%20overview.pdf*.

[134] Daniel Nyga. "Interpretation of Natural-language Robot Instructions". PhD thesis. University Of Bremen, May 2017.

[135] Daniel Nyga. *Markov Logic Networks in Python*. Accessed: 2018-11-22. 2017. URL: *http://pracmln.org/*.

[136] Daniel Nyga. *PRAC - Probabilistic Action Cores*. Accessed: 2018-11-22. Dec. 2017. URL: *http://www.actioncores.org/*.

[137] Mike Oaksford, Nick Chater, and Keith Stenning. "Connectionism, Classical Cognitive Science and Experimental Psychology". In: *AI Soc.* 4 (Jan. 1990), pp. 73–90. DOI: *10.1007/BF01889765*.

[138] Aude Oliva. "Gist of the Scene". In: *Neurobiology of Attention*. Ed. by Laurent Itti, Geraint Rees, and John K. Tsotsos. Academic Press, 2005, pp. 696–64.

[139] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. "Realistic Evaluation of Deep Semi-Supervised Learning Algorithms". In: *CoRR* abs/1804.09170 (2018). arXiv: *1804.09170*. URL: *http://arxiv.org/abs/1804.09170*.

[140] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *CoRR* abs/1511.08458 (2015). arXiv: *1511.08458*. URL: *http://arxiv.org/abs/1511.08458*.

[141] Ioannis Pachoulakis and Georgios Pontikakis. "Combining features of the Unreal and Unity Game Engines to hone development skills". In: (Nov. 2015).

[142] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "Understanding the exploding gradient problem". In: *CoRR* abs/1211.5063 (2012). arXiv: *1211.5063*. URL: *http://arxiv.org/abs/1211.5063*.

[143] Nikolaos G. Paterakis, Elena Mocanu, Madeleine Gibescu, Bart Stappers, and Walter van Alst. "Deep learning versus traditional machine learning methods for aggregated energy demand prediction". In: *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2017, Torino, Italy, September 26-29, 2017*. 2017, pp. 1–6. DOI: *10.1109/ISGTEurope.2017.8260289*. URL: *https://doi.org/10.1109/ISGTEurope.2017.8260289*.

[144] Rakesh Ranjan Pathak. "Sample size: from formulae to concepts - II". In: *International Journal of Basic & Clinical Pharmacology* 2 (Feb. 2013). DOI: *10.5455/2319-2003.ijbcp20130119*.

[145] Partha Sarathi Paul, Surajit Goon, and Abhishek Bhattacharya. "HISTORY AND COMPARATIVE STUDY OF MODERN GAME ENGINES". In: 2012.

[146] Henry Peter, Krainin Michael, Herbst Evan, Ren Xiaofeng, and Fox Dieter. "RGB-D Mapping: Using Kinect-Style Depth Cameras for Dense 3D Modeling of Indoor Environments". In: *International Journal of Robotic Research - IJRR* 31 (Apr. 2012), pp. 647–663. DOI: *10.1177/0278364911434148*.

[147] Cristina Petri. "Decision Trees". In: 2010.

[148] Mareike Picklum. "To See What No Robot Has Seen Before". MA thesis. •: University of Bremen, Jan. 2015.

[149] Dean A. Pomerleau, Jay Gowdy, and Charles E. Thorpe. "Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance". In: 1991.

[150] Weichao Qiu and Alan L. Yuille. "UnrealCV: Connecting Computer Vision to Unreal Engine". In: *CoRR* abs/1609.01326 (2016).

[151] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, and Yizhou Wangand Alan Yuille. *UnrealCV: Connecting Computer Vision to Unreal Engine*. Accessed: 2019-02-15. URL: *https://github.com/unrealcv/unrealcv*.

[152] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, and Yizhou Wangand Alan Yuille. "UnrealCV: Virtual Worlds for Computer Vision". In: *ACM Multimedia Open Source Software Competition* (2017).

[153] H. Rasshofer R and K Gresser. "Automotive Radar and Lidar Systems for Next Generation Driver Assistance Functions". In: *Advances in Radio Science - Kleinheubacher Berichte* 3 (May 2005). DOI: *10.5194/ars-3-205-2005*.

[154] MarcÁurelio Ranzato. *Supervised Deep Learning*. Accessed: 2019-03-21. URL: *https://www.labri.fr/perso/vlepetit/teaching/selected_topics_on_computer_vision_material/ranzato_cvpr2014_DLtutorial.pdf*.

[155] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: *1506.02640*. URL: *http://arxiv.org/abs/1506.02640*.

[156] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: *1506.01497*. URL: *http://arxiv.org/abs/1506.01497*.

[157] Softbank Robotics. *Pepper, Softbank*. Accessed: 2019-01-12. URL: *https://www.softbankrobotics.com/us/sites/all/themes/krm/img/Pepper/SBR_Pepper-Hero_FullRobot.png*.

[158] Marko A. Rodriguez and Joe Geldart. "An Evidential Path Logic for Multi-Relational Networks". In: *Technosocial Predictive Analytics, Papers from the 2009 AAAI Spring Symposium, Technical Report SS-09-09, Stanford, California, USA, March 23-25, 2009*. 2009, pp. 114–119. URL: *http://www.aaai.org/Library/Symposia/Spring/2009/ss09-09-021.php*.

[159] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Berlin, Heidelberg: Springer-Verlag, 1996. ISBN: 3-540-60505-3.

[160] Lior Rokach and Oded Maimon. "Decision Trees". In: vol. 6. Jan. 2005, pp. 165–192. DOI: *10.1007/0-387-25465-X_9*.

[161] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015).

[162] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review* (1958), pp. 65–386.

[163] Sebastian Ruder. "An Overview of Multi-Task Learning in Deep Neural Networks". In: *CoRR* abs/1706.05098 (2017). arXiv: *1706.05098*. URL: *http://arxiv.org/abs/1706.05098*.

[164] Javier Ruiz-del-Solar, Patricio Loncomilla, and Naiomi Soto. "A Survey on Deep Learning Methods for Robot Vision". In: *CoRR* abs/1803.10862 (2018). arXiv: *1803.10862*. URL: *http://arxiv.org/abs/1803.10862*.

[165] Prof. Rus, Teller Siegwart, and Nourbakhsh. *Cameras, Images, and Low-Level Robot Vision*. Feb. 2011. URL: *http://courses.csail.mit.edu/6.141/spring2011/pub/lectures/Lec05-CameraVision.pdf*.

[166] Albawi Saad, Abed MOHAMMED Tareq, and ALZAWI Saad. "Understanding of a Convolutional Neural Network". In: Aug. 2017. DOI: *10.1109/ICEngTechnol.2017.8308186*.

[167] Mousavi Sajad, Schukat Michael, and Howley Enda. "Deep Reinforcement Learning: An Overview". In: June 2018, pp. 426–440. ISBN: 978-3-319-56990-1. DOI: *10.1007/978-3-319-56991-8_32*.

[168] Lila San Roque, Kobin H. Kendrick, Elisabeth Norcliffe, Penelope Brown, Rebecca Defina, Mark Dingemanse, Tyko Dirksmeyer, N. J. Enfield, Simeon Floyd, Jeremy Hammond,

Giovanni Rossi, Sylvia Tufvesson, Saskia van Putten, and Asifa Majid. "Vision verbs dominate in conversation across cultures, but the ranking of non-visual verbs varies". English. In: *Cognitive Linguistics* 26.1 (2015), pp. 31–60. ISSN: 0936-5907.

[169] Andrew Sanders. *An Introduction to Unreal Engine 4*. Natick, MA, USA: A. K. Peters, Ltd., 2016. ISBN: 1498765092, 9781498765091.

[170] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy P. Lillicrap. "A simple neural network module for relational reasoning". In: *CoRR* abs/1706.01427 (2017). arXiv: *1706.01427*. URL: *http://arxiv.org/abs/1706.01427*.

[171] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *CoRR* abs/1404.7828 (2014). arXiv: *1404.7828*. URL: *http://arxiv.org/abs/1404.7828*.

[172] Tanja Schultz, John Bateman, Gordon Cheng, and Hagen Langer. *EASE - Everyday Activity Science and Engineering*. Accessed: 2018-11-19. May 2017. URL: *http://ease.informatik.uni-bremen.de/wp-content/uploads/2017/05/research-areas.pdf*.

[173] Brent Schwarz. "Lidar: Mapping the world in 3D". In: *Nat. Photonics* 4 (July 2010). DOI: *10.1038/nphoton.2010.148*.

[174] Mohammad Javad Shafiee, Akshaya Kumar Mishra, and Alexander Wong. "EvoNet: Evolutionary Synthesis of Deep Neural Networks". In: *CoRR* abs/1606.04393 (2016). arXiv: *1606.04393*. URL: *http://arxiv.org/abs/1606.04393*.

[175] Lin Shao, Ye Tian, and Jeannette Bohg. "ClusterNet: Instance Segmentation in RGB-D Images". In: *CoRR* abs/1807.08894 (2018). arXiv: *1807.08894*. URL: *http://arxiv.org/abs/1807.08894*.

[176] C. Shi-kuo, J. E, and T. Genoveffa. *Intelligent Image Database Systems*. Series On Software Engineering And Knowledge Engineering. World Scientific Publishing Company, 1996. ISBN: 9789814499934. URL: *https://books.google.de/books?id=4OzsCgAAQBAJ*.

[177] Ravid Shwartz-Ziv and Naftali Tishby. "Opening the Black Box of Deep Neural Networks via Information". In: *CoRR* abs/1703.00810 (2017). arXiv: *1703.00810*. URL: *http://arxiv.org/abs/1703.00810*.

[178] Hava T. Siegelmann and Eduardo D. Sontag. "On the Computational Power of Neural Nets". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 440–449. ISBN: 0-89791-497-X. DOI: *10.1145/130385.130432*. URL: *http://doi.acm.org/10.1145/130385.130432*.

[179] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: *http://arxiv.org/abs/1409.1556*.

[180] SimpleBiology. *CONDUCTION OF NERVE IMPULSE*. Accessed: 2019-03-17. URL: *http://simplebiologyy.blogspot.com/2014/08/conduction-of-nerve-impulse.html*.

[181] Aarti Singh and Barnabas Poczos. *Neural Networks*. Accessed: 2019-03-21. URL: *http://www.cs.cmu.edu/~aarti/Class/10701_Spring14/slides/NeuralNetworks.pdf*.

[182] Nikhil Somani, Emmanuel C. Dean-Leon, Caixia Cai, and Alois Knoll. "Perception and Reasoning for Scene Understanding in Human-Robot Interaction Scenarios". In: 2013.

[183] 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. URL: *http://jmlr.org/papers/v15/srivastava14a.html*.

[184] Ioannis Stamos. *3D Computer Vision: CSc 83020*. Accessed: 2019-01-14. istamos (at) hunter.cuny.edu. URL: *http://www.cs.hunter.cuny.edu/%C4%A9oannis/CSc83020_lec00_Introduction.ppt*.

[185] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning". In: *CoRR* abs/1712.06567 (2017). arXiv: *1712.06567*. URL: *http://arxiv.org/abs/1712.06567*.

[186] R. Sun and F. Alexandre. *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*. David Fulton / Nasen. Taylor & Francis, 2013. ISBN: 9781134802135. URL: *https://books.google.de/books?id=n7_DgtoQYlAC*.

[187] Ron Sun. "Hybrid Connectionist-Symbolic Modules: A Report from the IJCAI-95 Workshop on Connectionist-Symbolic Integration". In: *AI Magazine* 17 (1996), pp. 99–103.

[188] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *CoRR* abs/1602.07261 (2016). arXiv: *1602.07261*. URL: *http://arxiv.org/abs/1602.07261*.

[189] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842 (2014). arXiv: *1409.4842*. URL: *http://arxiv.org/abs/1409.4842*.

[190] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. "Deep Neural Networks for Object Detection". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 2553–2561. URL: *http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf*.

[191] Huang T. "Computer Vision: Evolution And Promise". In: (1996). URL: *http://cds.cern.ch/record/400313*.

[192] Lei Tai and Ming Liu. "Deep-learning in Mobile Robotics - from Perception to Control Systems: A Survey on Why and Why not". In: *CoRR* abs/1612.07139 (2016). arXiv: *1612.07139*. URL: *http://arxiv.org/abs/1612.07139*.

[193] TechTalk. *Memory Trends*. Accessed: 2019-04-20. URL: *https://techtalk.pcpitstop.com/research-charts-memory/*.

[194] Damien Teney, Lingqiao Liu, and Anton van den Hengel. "Graph-Structured Representations for Visual Question Answering". In: *CoRR* abs/1609.05600 (2016). arXiv: *1609.05600*. URL: *http://arxiv.org/abs/1609.05600*.

[195] Moritz Tenorth and Michael Beetz. "KNOWROB — knowledge processing for autonomous personal robots". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009), pp. 4261–4266.

[196] Moritz Tenorth, Jan Oliver Winkler, Daniel Beßler, and Michael Beetz. "Open-EASE: A Cloud-Based Knowledge Service for Autonomous Learning". In: *KI* 29.4 (2015), pp. 407–411. DOI: *10.1007/s13218-015-0364-1*. URL: *https://doi.org/10.1007/s13218-015-0364-1*.

[197] Simon I. Thorpe, Denis Fize, and Catherine Marlot. "Speed of processing in the human visual system". In: *Nature* 381 (1996), pp. 520–522.

[198] Petri Toiviainen. "Symbolic AI versus Connectionism in Music Research". In: *Readings in Music and Artificial Intelligence*. 2000.

[199] Dmitry Tsarkov and Ian Horrocks. "DL Reasoner vs. First-Order Prover". In: *Description Logics*. 2003.

[200] A. M. TURING. "I.—COMPUTING MACHINERY AND INTELLIGENCE". In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423. DOI: *10.1093/mind/LIX.236.433*. eprint: *http://oup.prod.sis.lan/mind/article-pdf/LIX/236/433/9866119/433.pdf*. URL: *https://dx.doi.org/10.1093/mind/LIX.236.433*.

[201] Julia Valder. *Comparison between Unreal Engine 4 and Vizard VR Toolkit as platforms for virtual experiments in pedestrian dynamics using the Oculus Rift*. Tech. rep. pdf darf open access sein. FH Aachen, 2015, 22 p. URL: *http://juser.fz-juelich.de/record/280982*.

[202] David Vernon. "The Space of Cognitive Vision". In: *Cognitive Vision Systems: Sampling the Spectrum of Approaches*. Ed. by Henrik I. Christensen and Hans-Hellmut Nagel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 7–24. ISBN: 978-3-540-33972-4. DOI: *10.1007/11414353_2*. URL: *https://doi.org/10.1007/11414353_2*.

[203] Analytics Vidhya. *The Evolution and Core Concepts of Deep Learning & Neural Networks*. Accessed: 2019-03-17. URL: *https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/*.

[204] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. "Deep Learning for Computer Vision: A Brief Review". In: *Hindawi Computational Intelligence and Neuroscience* 2018 Article ID 7068349 (2018). URL: *https://doi.org/10.1155/2018/7068349*.

[205] T Wiedemeyer, F Bálint-Benczédi, and Michael Beetz. "Pervasive 'calm'∗ perception for autonomous robotic agents". In: 2 (Jan. 2015), pp. 871–879.

[206] WikiHow. *4 Ways to Make Basic Instant Coffee*. Accessed: 2019-01-23. URL: *https://www.wikihow.com/Make-Instant-Coffee*.

[207] Peratham Wiriyathammabhum, Douglas Summers-Stay, Cornelia Fermüller, and Yiannis Aloimonos. "Computer Vision and Natural Language Processing: Recent Approaches in Multimedia and Robotics". In: *ACM Comput. Surv.* 49 (2016), 71:1–71:44.

[208] Donald Wise. ""Intelligent" Design versus Evolution". In: *Science (New York, N.Y.)* 309 (Aug. 2005), pp. 556–7. DOI: *10.1126/science.309.5734.556c*.

[209] Kam W. Wong. "Machine Vision,Robot Vision,Computer Vision,and Close-Range Photogrammetry". In: *Photogrammetric Engineering and Remote Sensing (PE&RS)* (Aug. 1992). URL: *https://www.asprs.org/wp-content/uploads/pers/1992journal/aug/1992_aug_1197-1198.pdf*.

[210] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes". In: *CoRR* abs/1711.00199 (2017). arXiv: *1711.00199*. URL: *http://arxiv.org/abs/1711.00199*.

[211] Jianxiong Xiao and Leslie A. Kolodziejski. "A 2 D + 3 D Rich Data Approach to Scene Understanding by Jianxiong Xiao". In: 2013.

[212] M. Xie. *Fundamentals of Robotics: Linking Perception to Action*. Series in machine perception and artificial intelligence. World Scientific Pub., 2003, pp. 24–24. ISBN: 9789812383358. URL: *https://books.google.de/books?id=ixSdP6ENB5EC*.

[213] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated Residual Transformations for Deep Neural Networks". In: *CoRR* abs/1611.05431 (2016). arXiv: *1611.05431*. URL: *http://arxiv.org/abs/1611.05431*.

[214] Danfei Xu, Yuke Zhu, Christopher Bongsoo Choy, and Li Fei-Fei. "Scene Graph Generation by Iterative Message Passing". In: *CoRR* abs/1701.02426 (2017). arXiv: *1701.02426*. URL: *http://arxiv.org/abs/1701.02426*.

[215] Danfei Xu, Yuke Zhu, Christopher Bongsoo Choy, and Li Fei-Fei. "Scene Graph Generation by Iterative Message Passing". In: *CoRR* abs/1701.02426 (2017). arXiv: *1701.02426*. URL: *http://arxiv.org/abs/1701.02426*.

[216] Jungang Xu, Hui Li, and Shilong Zhou. "An Overview of Deep Generative Models". In: *IETE Technical Review* 32 (Dec. 2014), pp. 131–139. DOI: *10.1080/02564602.2014.987328*.

[217] Chengxi Ye, Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. "What Can I Do Around Here? Deep Functional Scene Understanding for Cognitive Robots". In: *CoRR* abs/1602.00032 (2016). arXiv: *1602.00032*. URL: *http://arxiv.org/abs/1602.00032*.

[218] Peter Yuen and Mark Richardson. "An introduction to hyperspectral imaging and its application for security, surveillance and target acquisition". In: *Imaging Science Journal* 58 (Oct. 2010), pp. 241–253. DOI: *10.1179/174313110X12771950995716*.

[219] Alom Md. Zahangir, M. Taha Tarek, Yakopcic Christopher, Westberg Stefan, Hasan Mahmudul, C Van Esesn Brian, Awwal Abdul, and Asari Vijayan. "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches". In: (Mar. 2018), pp. 11–12.

[220] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *CoRR* abs/1311.2901 (2013). arXiv: *1311.2901*. URL: *http://arxiv.org/abs/1311.2901*.

A.5 List of Abbreviations

AE(s) Auto-Encoder(s), S. 11, 47–49, 134
AI Artificial Intelligence, S. 2, 17, 27, 31, 32, 63, 127
ANN(s) Artificial Neural Network(s), S. 11, 31, 36, 67, 87, 90, 91, 95, 99, 134

CNN(s) Convolutional Neural Network(s), S. 11, 44–46, 48, 51, 52, 54, 57, 59, 61, 81, 85, 123, 134, 135
CPU(s) Central Processing Unit(s), S. 102, 103, 107, 110
CV Computer Vision, S. 3, 13, 14, 32, 44, 50, 57, 58, 82, 83, 87, 127, 128

DBMS DataBase Management System, S. 20
DBN(s) Deep Belief Network(s), S. 11, 48–50, 134, 135
DL Deep Learning, S. 3, 8, 14, 24–29, 32, 50, 58, 59, 61–64, 80–83, 90, 96, 100, 102–104, 106, 107, 109, 110, 117, 122, 127, 128, 130
DL(s) Description Logic(s), S. 11, 19, 20, 22, 23, 134
DNA Deoxyribo Nucleic Acid, S. 92, 105
DT(s) Decision Tree(s), S. 11, 13, 14, 31, 134

EASE Everyday Activity in Science and Engineering, S. 3, 4, 9–11, 15, 16, 26–28, 62, 65, 66, 69, 73, 79, 80, 110, 114, 124, 127, 128, 131, 133, 134, 136

FOL(s) First-Order Logic(s), S. 11, 19, 27, 31, 134
FPN Feature Pyramidal Network, S. 85–87, 104, 106, 128

GAP Global Average Pooling, S. 99, 128
GOFAI Good Old-Fashioned Artificial Intelligence, S. 2, 31
GPGPU(s) General Purpose Graphical Processing Unit(s), S. 87, 88, 90, 102, 103, 107, 110, 114, 123, 125, 130
GUI Graphical User Interface, S. 71, 79, 136

HOG Histogram of Oriented Gradient, S. 11, 134
HTTP Hypher-Text Transfer Protocol, S. 72

IAI-Bremen Institute For Artificial Intelligence At University of Bremen, S. 2–4, 15, 16, 62, 80, 110, 133, 134

IIS Integrated Intelligent Systems, S. 3

KNN(s) K-Nearest Neighbours Classifier(s), S. 13, 14, 31

LBP Linear Binary Pattern, S. 24

ML Machine Learning, S. 8, 103
MLN Markov Logic Network, S. 27
MLN(s) Markov Logic Network(s), S. 11, 134
MLP(s) Multi-Layer Perceptron(s), S. 36–38, 44, 46, 47, 57, 93, 95, 98, 99, 105, 128, 134
MNHMO Multiple Network Heads, Multiple Outputs, S. 89–91

PCA Principal Component Analysis, S. 11, 134
PVSURMT Problem of Visual Scene Understanding for Robot Manipulation Tasks, S. 17, 23–29, 81, 83, 89, 106, 109, 110, 116, 117, 122, 124, 127–131

RBM(s) Restricted Boltzmann Machine(s), S. 48–50, 135
ResNet Residual Network, S. 86, 104, 106, 128
RGB Red,Blue,Red, S. 109, 113–116, 120–126, 128–130, 136, 137
RGBD Red,Blue,Red,Depth, S. 26, 109, 116, 117, 119–122, 124–126, 128–130, 137
RNN(s) Reccurent Neural Network(s), S. 11, 46, 47, 57, 59, 61, 81, 90, 93, 134, 135
ROIAlign Region Of Interest Align and Pooling, S. 89, 105, 106
ROIPool Region Of Interest Pooling, S. 89
RPN(s) Region Proposal Network(s), S. 54, 57, 87–89, 92, 94, 95, 105, 106, 114, 123, 124, 128, 135
RV Robot Vision, S. 3, 4, 14, 32, 82

SGD Stochastic Gradient Descent, S. 110
SIFT Scale Invariant Feature Transform, S. 11, 24, 134
SNHMO Single Network Head, Multiple Outputs, S. 89, 90
SR Stimulus-Response, S. 7
SSD Single Shot MultiBox Detection, S. 53, 54, 87
SURF Speeded Up Robust Feature, S. 11, 134
SVM(s) Support Vector Machine(s), S. 11, 13, 134

UAT Universal Approximation Theorem, S. 36
UC(s) Use Case(s(, S. 20
UE Unreal Engine, S. 64, 69–72, 80, 127, 128, 136
UML Unified Modeling Language, S. 20

VQA Visual Question Answering, S. 57, 96
VR Virtual Reality, S. 79, 80

YOLO You Only Look Once, S. 53, 54, 87

A.6 Glossary

back-propagation
is a learning rule based on gradient descent or ascent and the derivative's chain rule to back propagate the error resulting from a neural inference, going from the output neurons to the input neurons., S. 35–37, 39, 47–49

Behaviorism
A scientific theory from Psychology that states that biological agents can be trained to learn a function just by controlling their environmental stimuli., S. 41

CRAM
Cogntive Robot Abstract Machine, is a framework that integrates sensorimotor feedbacks, knowledge and goals to provide realtime action planning., S. 4

GFLOPS
GFLOPS stands for Giga FLoating Operations Per Second and constitutes a very comprehensive unit for measuring the computing speed of microprocessors, S. 110

Gibb's sampling
Consists in randomly sampling a sequence based on conditional probabilities, S. 48

GISKARD
is a framework that controls robot motorics especially designed for manipulation tasks., S. 4

IOU
Intersection Over Union Operator of tho two areas A and B is the fraction of area A in area B., S. 53

KnowRob
Knowledge for robots, S. 4

OpenEASE
is a service that on the one hand provides an online database containing meaningfull robot experiences and commonsense knowledge. On the other hand, it offers researches and robots with tools to access and share those knowledge., S. 4

OS
OS stands for Operating System and as the most fundamental software within a computer system, it also provides a set of utilities to visualize the computational resources of running processes, S. 113, 116, 119, 121

POMDP
Partially Observable Markov Decision Process., S. 41

PRAC
Probabilistic Action Cores, is a framework that relies on probabilistic logical reasoning to disambiguate robot instructions formatted in natural language., S. 4

ROBCOG
is a virtual reality framework that allows robots to acquire knowledge from virtual worlds., S. 4, 69, 70

RoboSherlock
is a cognitive robot vision especially designed for manipulation tasks. , S. 4, 23, 26, 27

SOAR
Acronym for State Operator Apply Result, is an explitcit cognitive architecture that aims at human cognition, S. 4, 133