TU/e - Department of Mathematics and Computer Science
Data-Mining Research Group

Philips Research - Department of Brain, Behaviour & Cognition

Bridging the Domain-Gap in Computer Vision Tasks

Master Thesis

J.G.C. Brouns

Supervisors:

Dr. D. Mocanu	TU/e
PDEng. ir. D. Lowet	Philips Research

Assessment Committee:

Dr. D. Mocanu	TU/e
PDEng. ir. D. Lowet	Philips Research
Dr. ir. A. Serebrenik	TU/e

Public Version

Eindhoven, December 2019

Abstract

Via this work, Philips Research explored the feasibility of an augmented reality driven interactive user manual for Philips consumer products. It was investigated whether a mobile app can be developed for consumers which is able to estimate the pose of complex and texture-poor Philips consumer products based on the device's camera-feed using state-of-the-art deep learning computer vision algorithms. Subsequently, this estimated object's pose can be used to augment the camera feed with virtual 3D cues. In order to train such deep learning computer-vision models, lots of annotated data is required. The acquisition of real training-data for computer-vision algorithms is a resource-demanding (and sometimes impossible) task and hence focus was shifted to training models using only synthetic (computer generated) image data. A tool called Philips Synthetica was developed, which can generate annotated images based on a computer-aided-design (CAD) model of an object. The properties of synthetic data can be leveraged to create a theoretical unlimited amount of statistical unbiased training-data that can be completely shaped by the data-engineer. However, computer vision models trained with only synthetic data are confronted with the domain gap or reality gap problem; a discrepancy between source domain (synthetic training data) and target domain (real test data). This research investigates two techniques for bridging this domain-gap: Domain Randomization and Generative Adversarial Data Enhancement using CylceGANs. We show on various computer-vision architectures (Convolutional Pose Machines and YOLOV3) and various computer-vision tasks (keypoint prediction and object localization and classification), that models trained with our GAN-enhanced data outperform models trained with the original data. Moreover we concluded that domain randomized data is benevolent for task performance in the real domain especially when combined with (semi) photo-realistic synthetic data. Finally, a proof-of-concept mobile iOS app for the Philips use-case is presented which utilizes the best-performing pose-estimation model from the experiments.

Preface

I would like to thank my university supervisor Decebal Mocanu. Besides the excellent academic accompaniment, Decebal also provided emotional support throughout the duration of this Thesis. Decebal makes his students feel like colleagues and friends, rather than just students. Moreover, via his network and recommendation, Decebal introduced the possibility for me to join the Philips Brain, Behavior & Cognition team for my master thesis with complete academic freedom. Furthermore, I would like to thank my company supervisor Dietwig Lowet. We both shared the same interest, vision and potential of the concept and had inspiring discussions. Dietwig's enthusiasm always managed to lift me up, pursuing me to push further than what we thought was initially possible. Dietwig also provided me with plenty of freedom and space which I really appreciated. I want to thank Alexander Serebrenik for taking the time to be in my thesis assessment committee. I want to thank Simin Chen for his excellent professional advice on streamlining the thesis structure.

I would like to thank Igrid Hietbrink for the help in coordinating the search for the necessary CAD-models. Hans Rogatschnig for the detailed help and providing various versions of CAD-models for Philips consumer products.

From my personal sphere I would like to thank Anava Jain for the provided emotional support during a hard period that occurred in the last year. Last, but certainly not least, I would like to thank my parents for everything they have done for me. They never stopped believing in me and stood by me in times of prosperity and in times of hardship.

Jeroen Brouns
Eindhoven
December 2019

Contents

Contents **v**

List of Figures **viii**

List of Tables **xi**

1 Introduction **1**
- 1.1 Motivation 1
  - 1.1.1 Philips and Augmented Reality 2
  - 1.1.2 Augmented Reality 2
- 1.2 Project Scope 4
  - 1.2.1 Problem Statement as Defined By Philips 4
  - 1.2.2 Research Aim & Research Question 4
- 1.3 Previous Work at Philips 6
- 1.4 Other Use-cases 7
  - 1.4.1 Counterfeit Product Detection 7
  - 1.4.2 Assistive Surgical Device Tooling 7
  - 1.4.3 Robotics 7
- 1.5 Contributions 8
  - 1.5.1 Practical Contributions 8
  - 1.5.2 Academic Contributions 8
- 1.6 Used code-bases 9
- 1.7 A Guide for the Reader 9

2 Pose Estimation **11**
- 2.1 Introduction 11
  - 2.1.1 Formal Definition of a Pose 11
  - 2.1.2 Pose Estimation 11
  - 2.1.3 The Point-n-Perspective Algorithm 12
- 2.2 Pose Estimation - Literature Study 14
  - 2.2.1 Human Pose Estimation 14
  - 2.2.2 Object Pose Estimation 15
  - 2.2.3 Conclusions from the Literature 15

3 You Only Look Once **17**
- 3.1 Introduction 17
- 3.2 Model Architecture 18
  - 3.2.1 Residual-block 18
  - 3.2.2 Detector-layer 18
  - 3.2.3 Upsampling-layer 19
  - 3.2.4 Non-maximum Suppression 19
- 3.3 Anchor Boxes 21

3.4 Loss Function . . . 22
3.4.1 Localization Loss . . . 22
3.4.2 Classification Loss . . . 23
3.4.3 Confidence Loss . . . 23
3.4.4 Total Loss Function . . . 23
3.5 Benefits of YOLO . . . 24
3.6 Limitations of YOLO . . . 24

4 Convolutional Pose Machines 25
4.1 Introduction . . . 25
4.2 Pose Machines . . . 26
4.3 Architecture . . . 27
4.4 Long-range Spatial Dependencies . . . 27
4.5 Extending Normal Convolutional Layers . . . 27

5 Synthetic Data 29
5.1 Introduction . . . 29
5.2 Why Use Synthetic Data? . . . 30
5.2.1 No Statistical Bias . . . 30
5.2.2 Cost and Time Effectiveness . . . 30
5.2.3 Anonymity . . . 30
5.3 Why Not Use Synthetic Data? . . . 31
5.3.1 Complexity . . . 31
5.3.2 Degree of photo-realism . . . 31
5.3.3 Domain-gap . . . 31
5.4 Use-cases for Synthetic Data . . . 32
5.4.1 Autonomous Vehicles . . . 32
5.4.2 Optical Character Recognition . . . 32
5.4.3 Anomaly Detection . . . 32
5.4.4 Facial Recognition . . . 32
5.5 Conclusions from the Literature . . . 33

6 Crossing the Domain-gap 35
6.1 Introduction . . . 35
6.2 Domain Randomization . . . 36
6.3 CycleGAN . . . 38
6.3.1 Introduction . . . 38
6.3.2 Definition of CycleGAN . . . 39
6.3.3 Loss Functions . . . 39
6.3.4 Full Trainings Objective . . . 41
6.3.5 The Intuition Behind Using GANs for our Use-case . . . 41
6.4 Research Question . . . 44

7 Methods 45
7.1 Data Generation & Collection . . . 46
7.1.1 The Data Annotation Formats . . . 46
7.1.2 Synthetic Data . . . 47
7.1.3 Evaluation Data-set in the Real Domain . . . 47
7.2 Experiment 1: Generative Adversarial Data Enhancement - Method . . . 48
7.2.1 Training CylceGAN . . . 48
7.2.2 Dataset Creation Using CylceGAN . . . 48
7.2.3 Evaluation Strategy . . . 49
7.3 Experiment 2: Benchmarking the Synthetic Data Types - Method . . . 50
7.3.1 A Note on Data Augmentation . . . 50

7.3.2 Experiment 2A - Evaluation on CPM . . . 52
7.3.3 Experiment 2B - Evaluation on YOLO . . . 54

8 Experiment Results 57
8.1 Experiment 1: Generative Adversarial Data Enhancement - Results & Discussion 57
8.1.1 Handpicked Generated Images . . . 57
8.1.2 Encountered Artifacts . . . 58
8.1.3 Visual Evaluation . . . 58
8.2 Experiment 2 : Benchmarking the Synthetic Data Types - Results & Discussion . . 60
8.2.1 Results of Experiment 2A - Convolutional Pose Machines . . . 60
8.2.2 Results of Experiment 2B - YOLO . . . 61
8.2.3 Experiment 2 - Discussion . . . 62
8.3 Answering the Research Question . . . 64

9 Proof of Concept Prototype 65
9.1 Implementation . . . 65
9.1.1 Inserting the Trained CPM Model . . . 65
9.1.2 The Pose Estimation Process . . . 65
9.2 Results . . . 67
9.2.1 Demo Videos . . . 67
9.3 Discussion . . . 68

10 Conclusions 69
10.1 The Academic Side . . . 69
10.1.1 Conclusions . . . 69
10.1.2 Future Work . . . 69
10.2 The Practical Side . . . 70
10.2.1 Conclusions . . . 70
10.2.2 Future Work . . . 70

Bibliography 71

Appendix 75

A Exploration by Predecessor 75
A.1 Existing Solutions . . . 75
A.2 Deep Learning . . . 76

B Creating the Evaluation Data-set in the Real Domain 78

C Philips Synthetica 79
C.1 What is Philips Synthetica? . . . 79
C.1.1 Computer-Aided-Design . . . 80
C.2 Modeling Realistic Reflections . . . 80
C.3 Calculating the Annotations . . . 83
C.3.1 Bounding-box Annotations . . . 83
C.3.2 Key-point Annotations . . . 83

List of Figures

1.1 The by Statista prospected growth of the global Augmented Reality revenue market in 2025 in comparison with 2017 and 2018 [1]. 3
1.2 Gartner's Hype Cycle for Emerging Technologies 2018 [2]. 3
1.3 Philips Airfryer HD9650 - Example of regions that do not allow for robust unique feature extraction using SIFT (in red) and regions that are suited for robust unique feature extraction (in green) 6

2.1 Pose ambiguity for PnP where $n = 4$ 13
2.2 PnP Algorithm - Overview. Known 3D information from the object (from for example a CAD-model) is leveraged to calculate a rotation and translation vector from 2D predictions. 14

3.1 YoloV3 Architecture Overview [3] 18
3.2 YOLO architectural elements: Residual Block 19
3.3 Overview of YOLO's prediction's output tensor in a grid-cell. The tensor contains predictions for $B = 3$ bounding-boxes, objectness score and class probability scores for each class [3] 20
3.4 Overview of what happens in the prediction layer of YOLO. The input-image is divided into a S x S grid. For each grid-cell, 3 bounding-boxes and a conditional class probability score are predicted. Subsequently, by means of a threshold and non-maximum-supression, the final output prediction is determined. 20
3.5 Example of the parameters necessary to calculate box coordinates for my dog Nala. 22

4.1 Convolutional Pose Machine - Schematic Overview. 26

6.1 The Domain-gap - A Schematic Illustration 35
6.2 Domain Randomization - Demo illustration of approach from original authors where low-fidelity rendered images with random camera positions, lightning conditions, object positions and non-realistic textures were created in order to train a detector. 36
6.3 Domain Randomization - Sensitivity on real images of the number of training samples used during training 37
6.4 Domain Randomization - Number of unique textures 38
6.5 Pix2PixHD - Example of their city generation use-case [4]. 38
6.6 Example inpainting results of Yu's method [5]. Missing regions are shown in white. The generator is trained to fill-in the white regions. 39
6.7 CylceGAN - Schematic Illustration of the training phase of learning the mapping $G : X \to Y$ where X is the synthetic data domain and y is the real data domain. . 40
6.8 CylceGan Cycle Consistency Loss - Visual Illustration 41
6.9 The Inverse Transform Method where an inverse CDF defines a mapping from a uniform distribution over [0,1] to a standard Gaussian distribution. 42
6.10 Style Transfer using CylceGAN - A mapping $G : X \to Y$ is learned to match the distribution of the synthetic input image with the distribution of the real data domain. 44

7.1 Example visualisation of the chosen 14 key-points 46
7.2 Example visualisation of a 2D bounding-box annotation 46
7.3 Randomly selected samples of $Dataset_{BASE}$ 47
7.4 Randomly selected samples of $Dataset_{DR}$ 47
7.5 Randomly selected (cropped) samples of $Dataset_{REAL}$ 48
7.6 Experiment Pipeline Setup - From data generation to evaluation. 53
7.7 Illustration of the Intersection over Union (IoU) performance metric on the ground-truth and predicted bounding-box around my Belgian Malinois "Nala". IoU takes a range $< 0, 1 >$, where 1 is an ideal match where both boxes are completely overlapping. 55

8.1 Comparison of handpicked synthetic input data (left) with generated data by the CycleGAN of the same synthetic data (middle) and handpicked unpaired real data (right) for reference. 57
8.2 Results for the different instances of Convolutional Pose Machines trained with different types of data. The mean Euclidean distance error of all real test-images is depicted per synthetic data-type category. 60
8.3 Violin-plot of the experiment results grouped per synthetic data category. Each data-point represents the mean average Euclidean distance for a specific test-image in the test data-set (which is the mean euclidean distance between the ground truth and predicted coordinates of all the 14 key-points in that image). 61
8.4 Results for the different instances of YOLO models trained with different types of data. The mAP score is depicted per synthetic data type category for data-sets of 20k and 5k samples. A test split of 10% was used. 61

9.1 Proof of concept prototype - Schematic overview of the application's pose-estimation process. Every frame, the image buffer is read from the device's camera. From this image, the MLModel predicts beliefmaps which are converted to (x,y) coordinates after which a pose is estimated using the point-n-perspecive algorithm. Subsequently, the pose can be used to update the GUI elements. 66
9.2 Practical Prototype - Augmenting the real airfryer object with an overlay of its 3D CAD model. 67
9.3 Proof of concept prototype - The user is asked to turn on the airfryer by pressing the ON/OFF button (left). Despite the heavy distortion of the camera feed, because the user is changing its position, the augmented virtual cue (the 3D arrow) remains in the correct position at all times (right). 67
9.4 Proof of concept prototype - Turning the air-fryer's main knob example. At all points, the pose estimation remains accurate so that the virtual cues can be augmented in the correct position and rotation with respect to the object. 68

A.1 Apple ARKIT - Creating a reference database by scanning features that can be used for matching during inference. 76
A.2 Philips Airfryer HD9650 - Example of regions that do not allow for robust unique feature extraction (in red) and regions that are suited for robust unique feature extraction (in green) 76

C.1 Philips Synthetica - Screenshot of an airfryer image being generated by the tool . . 79
C.2 Philips Synthetica - Wall that gets assigned random images from the VOC2012 datasets. The air fryer object can now reflect the images on this picture wall by means of a reflection probe. Note that in the actual tool, this picture wall has 6 surfaces, completely enclosing the object. 81
C.3 Philips Synthetica - Before (top) and after (bottom) pictures when using a reflection probe. In the bottom picture it is clearly visible that the sphere reflects its surroundings. 81

C.4 Cubemap example - The view from six directions are painted on the interior surfaces of a cube. 82
C.5 Philips Synthetica - Screenshot of Philips Synthetica generating an airfryer image. Note the reflections of the picture-wall in the air fryer object. 82

List of Tables

1.1 Moscow Requirements that define the scope of the practical part of this thesis. . . 5

7.1 The used hyper-parameter settings for the Convolutional Pose Machine architecture that were used in experiment 2A. 52
7.2 The used hyper-parameter settings for the YOLOv3 architecture that was used in experiment 2B. 54

8.1 Descriptive statistics of the mean Euclidean distance score per synthetic data category for the CPM experiment 60

Chapter 1

Introduction

1.1 Motivation

Imagine unwrapping that freshly bought Philips Airfryer that you have always wanted. You can finally start eating healthy and leave that greasy traditional deep-fryer on the shelf! There is one problem though; you are gazing at all the separate parts that are supposed to be somehow build into a working Airfryer and you have no idea where to start. Its a complex product, at least, so you have been told by that saleswoman in the electronics store; it can cook vegetables, fry fries, roast chicken and much more! So where to start? There is a 'quick-start' guide in the box, which really doesn't give you much info. A bit overwhelmed and visibly disappointed you reach out to the extensive user manual, which of course seems to come in every language except yours! At points like these, you wished that there was this person or magical device that could tell you exactly what to do and help you in every step along the way. That saleswoman in the store sure seemed to know everything about the device and in the store she made it look so easy. Imagine how easy it would be if our phones could teach us all about our new device? Imagine we could point our phones at devices that we want to learn more about and let them guide us step-by-step through interactive tutorials using audio and visual cues. That sure beats reading a text manual! The phone can be pointed at the air fryer and it automatically recognizes, which type of air fryer we are dealing with. Augmented reality is being used to show information about the device in real-time. Very convenient, since those illustrations in textual manuals are often quite unclear to you and let's face it: "you never had good three-dimensional spatial awareness". *"Place the container in the base by pulling down the 2 little levers on both sides"*, it sounds from the phone's speakers while two little 3D arrows show you exactly which levers are supposed to be pulled down. The levers are very small and barely visible, but luckily the phone is showing you exactly where they are. *"Now turn the knob to set the desired cooking mode to 'steaming' and press 'ready'"*. You did it! You just learned to air-fry your first broccoli and you had fun doing so. You chuck away that big printed manual in the nearest trash-bin, you definitely will not be using that anymore.

Already in 1980, Caroll et. al. formulated in their theory of the Active User Paradox that people don't read manuals, but dive right into tasks until they get stuck or finish through trial and error [6]. Besides textual manuals being tedious for the consumer to read, these manuals also bring high resource costs and an increased CO_2 footprint. Even some car manufacturers do not provide printed copies of the car's manual anymore but make use of online manuals. The in this thesis proposed concept is a counteract measure to people misusing the product because they have not properly read the manual and answers an observed trend of digitized product manuals. Furthermore, augmented reality can provide a great toolset for engaging the user and offering (premium) unique value to the customer in comparison with competitors. Hence, the concept could positively increase Philips' brand perception. The concept of augmented-reality interactive user-manuals does not have to stay limited to business-to-consumer solutions. Within business-to-business solutions and use-cases, augmented-reality is being employed for educating and training employees in a more (cost)effective way. Imagine a factory worker that needs to be instructed to work with a certain machine. An interactive augmented-reality education system could engage the scholar by providing a rapid and unique learning experience.

1.1.1 Philips and Augmented Reality

[redacted]

1.1.2 Augmented Reality

Besides increasing the value of existing Philips products and services, occasionally the group tries to investigate ways to create new customer and business value. A potential tool for this was identified to be Augmented Reality. It was estimated by Statista [1] that the market for Augmented Reality will increase from roughly 3.5 billion to more than 198 billion U.S. dollars; an astounding growth factor of 56 (see Figure 1.1). When we look at Gartner's hype cycle [2] (see figure Figure 1.2), we can observe that Augmented Reality has surpassed its Innovation Trigger, Peak of Inflated Expectations and is now in its Trough of Disillusionment. A technology breakthrough was made which got boasted with early publicity and several success stories. Some companies took action, but many did not. Currently, as Augmented Reality is in its Trough of Disillusionment, interest wanes as experiments and implementations fail to deliver. Producers of the technology shake out or fail. Investments continue only if the surviving providers improve their products to the satisfaction of early adopters. Initially, pioneers and enthusiasts thought that augmented reality-driven glasses or contact-lenses would take over the role over our smartphone, yet it did not. Experiments such as Google Glass failed due to friction with social constructions, privacy concerns, no clear customer-benefit and a poor market-strategy [7]. Since then, the use-cases for Augmented Reality were redefined and currently focus on B2B e-learning and customer service. Specific use-cases still remain for B2C such as IKEA's virtual catalog, yet these uses-cases seem to be limited to games and user-engagement. Philips tries to continuously position itself as a pioneer of new emerging technologies and augmented reality is such a technology that Philips wishes to explore.

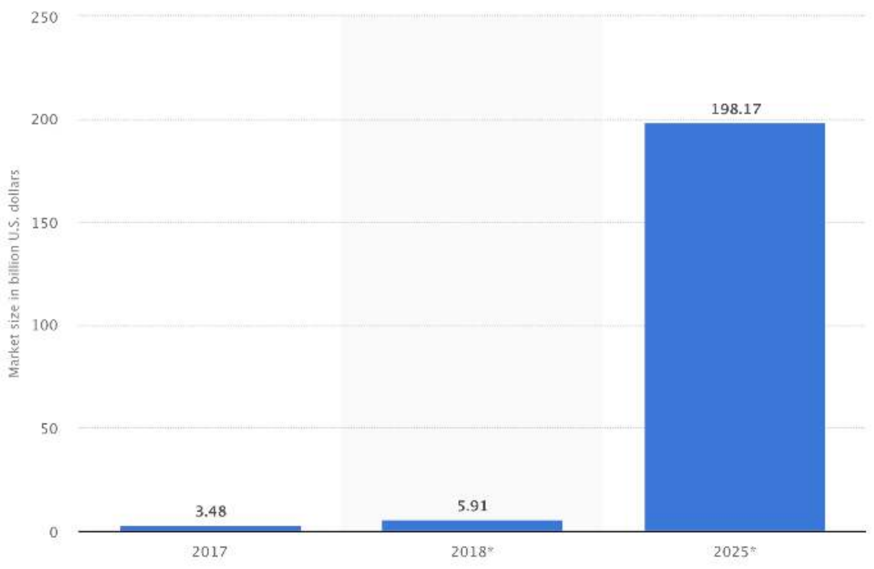


Figure 1.1: The by Statista prospected growth of the global Augmented Reality revenue market in 2025 in comparison with 2017 and 2018 [1].

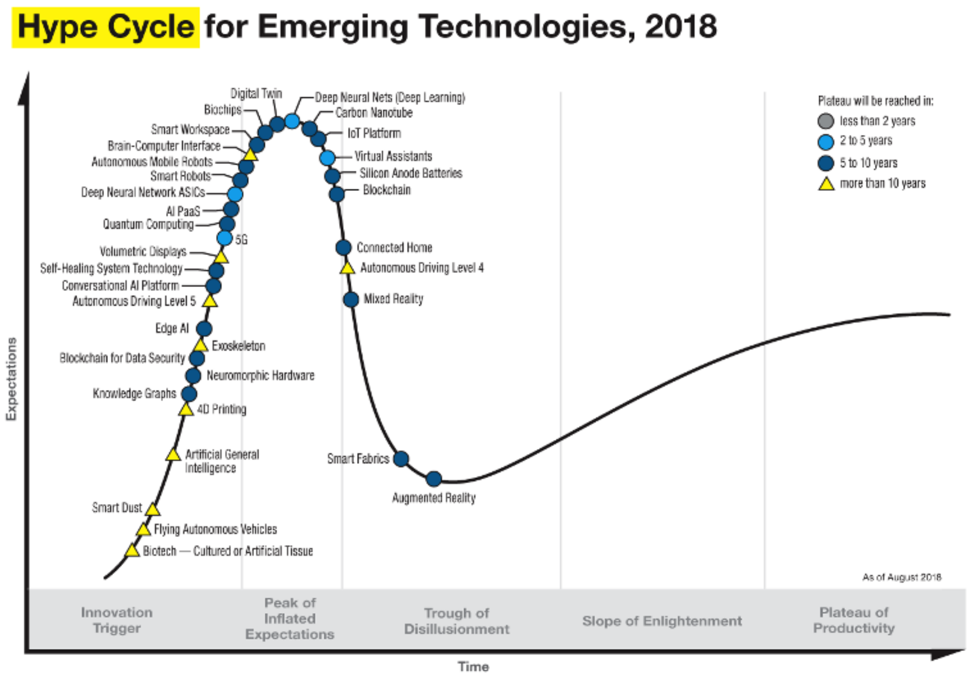


Figure 1.2: Gartner's Hype Cycle for Emerging Technologies 2018 [2].

1.2 Project Scope

For clarity, the problem statement of this thesis is split up into 2 parts: a practical side and an academic side. The practical side entails the exploration towards a solution to the concept: an augmented reality interactive user-manual. The academic side answers a theoretical research question that was explored during the creation of the practical prototype which supports the solution's practical scalability. First, the problem statement and a requirements overview for the practical deliverable for Philips is given. Subsequently, the academic research question is stated.

1.2.1 Problem Statement as Defined By Philips

As stated before, the goal of the concept is to create a digital manual on a mobile phone that uses augmented reality to provide the user with virtual audio(visual) instruction cues. The mobile device's camera is used to continuously capture a video-stream of the 'to-be-detected' object. This video-stream needs to be processed and analyzed in (preferably) real-time. Besides detecting and identifying a specific object in its entirety, individual sub-components of this object also need to be identified (e.g. we want to detect an Airfryer, identify the model's type and then detect individual subcomponents such as the buttons, air-vent, food-container, LED-screen so that we can assign 3D virtual cues such as arrows or tooltips to the locations of these sub-components).

Taking into consideration the project objectives and motivations, an analysis of results of the predecessor and the results and conclusions of an initial brief exploration phase (discussed in more detail in section 1.3), a practical problem statement can be defined as follows:

P1 ***Does there exists (or can we develop) a technology that can detect and identify objects and it's sub-components based on an RGB camera feed so that we can augment the video stream with virtual (audio)visual cues in a 3D virtual augmented world?***

In Table 1.1, an overview can be found of project requirements that were created using the MoSCoW method [8] (a method commonly used to help key stakeholders understand the significance of initiatives in a specific release).

1.2.2 Research Aim & Research Question

Most (un)supervised machine-learning algorithms need annotated training-data. Collecting and annotating this data is often done by hand, which is resource and cost-intensive. Philips has quite a few consumer products and collecting and annotating training-data for all these products by hand is simply 'not a scalable option'. A possible solution could be training the same deep-learning models with synthetic data: *"data that is generated instead of obtained by direct measurement"*. Synthetic data has much potential since a theoretically unlimited amount of (automatically annotated) training data can be generated on-demand. Unfortunately, the use of synthetic data often yields to lower performance on of the models on real data, due to the dissimilar distributions between the real and the synthetic domain (called a domain-gap). Two techniques are identified that could help minimize this domain-gap, effectively improving model performance in the real domain. These techniques are: domain randomization and generative adversarial data enhancement. The main research question is defined as follows:

R1 **Can we enhance synthetic image training data using generative adversarial and domain randomisation methods to improve the performance of deep learning models on computer vision tasks in the real domain?**

ID	**Type**	***Description***
REQ0	Must have	The developed detection algorithm should be able to detect reflective and texture-poor objects such as the Philips Airfryer HD9650.
REQ1	Must have	The detection algorithm uses a deep learning strategy
REQ2	Must have	The detection algorithm can be trained using only synthetic data
REQ3	Must have	The detection algorithm can reach accurate enough pose estimation performance for use in a production setting
REQ4	Must have	Inference can be performed in real-time ($> 24fps$) in a desktop-environment using a high-end GPU (*NVIDIA 1080ti and above*)
REQ5	Should have	Inference can be performed in real-time ($> 24fps$) on flagship iOS mobile devices (*iPads and iPhones produced in 2018 and above*)
REQ6	Must have	The solution contains one Philips consumer product.
REQ7	Could have	The solution contains multiple Philips consumer products.
REQ8	Should have	The solution is accurate enough to create a virtual 3D overlay of the object on top of the physical object.
REQ9	Should have	The solution can distinguish multiple similar looking Philips consumer products.
REQ10	Should have	The solution can be deployed for the Google Android operating system
REQ11	Should have	The solution can be deployed for the Apple IOS operating system

Table 1.1: Moscow Requirements that define the scope of the practical part of this thesis.

1.3 Previous Work at Philips

Philips started to explore the road-map to developing a solution for such an augmented reality interactive manual. One senior FTE was tasked with investigating potential development methods and technologies. The research was conducted into both existing solutions offered by third parties and custom solutions that had to be developed in-house. Some conclusions were drafted, but no proper final solution was created. Existing traditional methods make use of Scale-Invariant-Feature-Transform (SIFT) [9]. These are cheap, fast and theoretically robust computer vision algorithms that extract traditional features (such as edges, blobs, and corners) out of images that subsequently can be used for object detection through feature-matching against a database. SIFT algorithms require texture-rich objects since otherwise no features can be extracted. Unfortunately, many Phillips products are reflective and texture-poor (see Figure 1.3), which resulted in non-functioning prototypes. Because of this, attention was shifted to deep-learning techniques which are nowadays successfully dominating computer vision tasks [10][11][12]. Convolutional architectures are known to be able to recognize even the hardest objects. The predecessor trained a 2D convolutional object detector to detect various Philips consumer products. From this, it was concluded that 2D object detection is not sufficient for our goals since it requires each subcomponent of the object that we are interested in to be detected individually. However, if instead of 2D bounding-box predictions, an object's 'pose' could be estimated, it could tell us not only where an object is located, but also how this object is positioned in 3D space (rotation and translation). This, in turn, could help us deduce information about each sub-component (by leveraging known knowledge about the object's geometry) without needing individual inference. Hence the pose-estimation task was selected to be explored. Please refer to Appendix A for a more detailed analysis of the previous work and its conclusions. chapter 2 provides a formal definition of a pose.

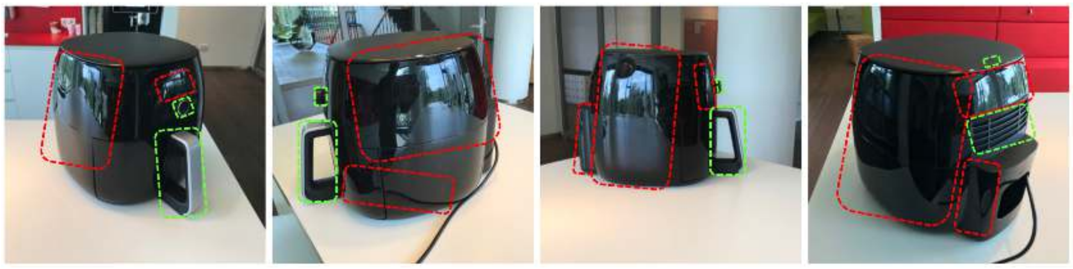


Figure 1.3: Philips Airfryer HD9650 - Example of regions that do not allow for robust unique feature extraction using SIFT (in red) and regions that are suited for robust unique feature extraction (in green)

1.4 Other Use-cases

Besides the main use-case for the technologies that were developed during this thesis, other use-cases were identified were the technology can be applied.

1.4.1 Counterfeit Product Detection

[redacted]

1.4.2 Assistive Surgical Device Tooling

[redacted]

1.4.3 Robotics

Within the field of robotics, pose-estimation is a popular and well-explored task. Since robots often need to be able to interact with their physical environment, they need to learn as much as possible about their environment. A concrete example of where pose-estimation in combination with Augmented Reality could be used is in Philips smart robot-vacuums. These robots autonomously navigate through the house, but certain areas are prohibited for these vacuums. A mobile phone could be used to mark these areas using pose-estimation and augmented-reality.

1.5 Contributions

This thesis attempts to create a solution for a concept-idea that involves a working prototype/demo of the concept and a set of tools that supported the development of this prototype. This latter 'set of tools' consists of a software-engineered pipeline for data creation. The academic contributions mainly involve insights in techniques that enable the practical solution to be highly scalable without experiencing exploding resource costs.

1.5.1 Practical Contributions

- **Philips Synthetica** - A unity-engine based photo-realistic synthetic data generator that can generate automatically annotated rendered images of an object in a scene utilizing a Computer-Aided Design (CAD) model. The tool supports various annotation output formats among which are individual keypoint annotation and 2D bounding-box annotation formats. The tool can be used to render photo-realistic materials that feature reflections and specific specular properties. Please refer to section C.1 for a more detailed description of Philips Synthetica.
- **Helper Scripts** - A collection of PYTHON conversion scripts that help with preprocessing tasks such as converting between data-formats, visualizing results and benchmarks, etc.
- **Working Prototype** - A native mobile iOS application that contains a proof-of-concept demo for the Philips air fryer HD9650 using the trained deep-learning models. Moreover, it features a test-mode, outlier detection and continuous averaging of the predictions and a data-gathering mode.

1.5.2 Academic Contributions

- A proven and elaborated use-case where a computer vision model was trained with only synthetic data and managed to obtain high inference accuracies in the real domain. Other works often require the presence of at least 10% of real data, while our results show that even in occluded scenarios, high performance can be obtained with only synthetic data.
- New insights on improving synthetic data through Generative Adversarial Networks without the need for strictly paired data. In contrast to existing work, this use-case is unique and novel in the sense that it focuses on an extremely complex object that contains noisy reflections.
- By means of an objective benchmark on different types of synthetic data, more insights were obtained on the recently popular domain randomization technique. Our work suggest that the domain randomization technique is most useful when combined with other (semi) photo-realistic data.
- A new publishable data-set that has annotations for both 2D bounding-boxes and keypoints. This data-set could be used in future work to further investigate the minimization of the domain-gap between the synthetic and the real domain. The dataset could be published as a benchmark dataset for a pose-estimation task in a crowd-sourcing competition.

1.6 Used code-bases

The following code base was used for training the You Only Look Once models:

- **A Keras implementation of YOLOv3 (Tensorflow backend)**
  `https://github.com/qqwweee/keras-yolo3`
  Author: qqwweee

The following code-base was used for training the CPM models on a desktop and testing the CPM models on mobile-devices:

- **Real-time single person pose estimation for Android and iOS**
  `https://github.com/edvardHua/PoseEstimationForMobile`
  Authors: edvardHua, tucan

Inspiration for the proof-of-concept was taken from the following code-base:

- **Real-time Mobile Car Pose Estimation with CoreML**
  `https://github.com/laanlabs/CarPoseDemo`
  Authors: laanlabs

1.7 A Guide for the Reader

As mentioned in the project scope in section 1.2, this thesis is split up into a practical part and an academic part.

For the practical side of this thesis, the focus lies on solving a pose-estimation task. chapter 2 provides a formal definition of a pose, a detailed elaboration of a popular post-processing algorithm and a literature study on pose-estimation. Note that for the academic side it is not required to read the pose-estimation chapter. Chapter 9 presents implementation details and demo media for the proof-of-concept prototype that was developed for our practical use-case as stated in subsection 1.2.1.

For the academic part, two popular computer-vision architectures (YOLO and Convolutional Pose Machines) and their coherent computer-vision tasks will be used to evaluate different types of synthetic data. The YOLO algorithm is described in full detail in chapter 3 and the Convolutional Pose Machine algorithm is described in full detail in chapter 4. Since the focus lies on evaluating the different types of synthetic data, it is not necessary to have a complete and thorough understanding of both these algorithms and the computer-vision tasks that they are solving. Hence, chapter 3 and chapter 4 are optional. The sections that are of importance for the academic part start at chapter 5, which provides a definition of synthetic data and describes the pros and cons and uses-cases of synthetic data. Chapter 6 introduces 2 methods to overcome the domain-gap problem that occurs with synthetic data after which a hypothesis is linked to the research questions. Subsequently, chapter 7 describes the research method and chapter 8 presents the results and discussion. For the academic part, chapter 9 is optional, since it describes the practical prototype. Finally, conclusions and future work are presented in chapter 10 for of both the academic experiments and the practical side.

Chapter 2

Pose Estimation

2.1 Introduction

In this section, a formal definition of a pose will be given. In the field of computer-vision, estimating the pose of objects or humans in image-data is a wildly explored task. Pose-estimation methods can be divided into several categories, which will be briefly described. Most pose-estimation methods convert some kind of two-dimensional predictions into a pose by means of the so-called "Point-n-Perspective" (PnP) algorithm. Because of this, the PnP algorithm will be elaborated in detail. Finally, a brief literature study on the most influential pose-estimation methods will be provided.

2.1.1 Formal Definition of a Pose

A pose describes the position and orientation relative to some coordinate system. A pose has 6 degrees of freedom, 3 for the axes of rotation: roll, pitch, yaw and 3 for the axes of translation: x-position, y-position, z-position. Note the importance of relativity to 'some coordinate system'. The pose's position and rotation are relative to a reference point; an origin pose with its coherent reference coordinate system.

2.1.2 Pose Estimation

Pose estimation is referred to as the task of determining the pose of an object or person in a (sequence of) RGB(D) (stereo) image(s). The pose estimation problem can be solved with different methods depending on the image sensor configuration and choice of methodology. Four classes of methodologies can be distinguished:

1. **Analytic and Geometric Methods -** Given the 3D geometry of the object, the mapping from 3D points in the world to 2D points and the image sensor calibration of the camera, the object's projection in an image is a well-known function of the object's pose. Specific points on an object (typically corners or rich feature points) are selected to serve as a reference. Via a combination of linear algebra equations its is now possible to estimate the pose of the 3D points with respect to its 3D reference points.

2. **Genetic algorithm methods -** A more robust, yet computationally more expensive approach that is useful in scenarios where the sensor calibration is of poor quality. The pose represents the genetic representation. The fitness function becomes the error between projected points in the image and its object reference points.

3. **Learning-based methods -** Methods that use artificial learning-based systems that directly learn the mapping of the pose to a 2D image.

4. **Hybrid approach -** A combination of the above methodologies

2.1.3 The Point-n-Perspective Algorithm

The Point-n-Perspective (PnP) algorithm tries to solve the problem of estimating the pose of a calibrated camera given a set of n 3D points P in the world-space and their corresponding 2D projections P' in an image. The PnP algorithm is a vital component of many pose estimation methods used in both robotics [14] and augmented reality [15] [16] [17].

The n in point-n-perspective stands for the amount of used input reference points. In its minimal form, solutions for PnP exists for $n >= 3$ (P3P). However, with just three points of reference, P3P can yield op to four real geometrically feasible solutions. Increasing the number of correspondence points (that preferably lay on different orientation axis) can optimize the solution by removing translation and rotation ambiguity. An example of an incorrect pose estimation due to lack of orientation information is depicted in Figure 2.1.

Formal problem definition: Given a set of correspondences between 3D points p_i expressed in a world reference frame, and their 2D projections p'_i onto an image plane, we attempt to retrieve the pose (a rotation vector R and a translation vector t) of the camera w.r.t. the real world coordinate-system and the focal length f. Figure 2.2 provides an abstract-level overview of the PnP problem definition.

Given all the above, we yield the perspective projection model for a pinhole-camera:

$$s\,p_c = K\,[\,R\,|\,T\,]\,p_w$$

where:

- s represent the Z coordinate of the 3D point in the camera coordinate system and can be used as a scale factor. When transforming between the camera coordinate system to the image-plane coordinate system, a dimension is lost. The scale factor can be used to determine whether the object is a small object viewed from a small distance or a big object viewed from a big distance.

- $p_w = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ is the homogeneous world point where x, y, z represent the position coordinate of a 3D point on their respective axis.

- $p_c = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ is the corresponding homogeneous image point where u and v respectively represent the x and y coordinates of a 2D point position in pixel coordinates on the image plane.

- R and t are the 3D rotation and 3D translation of the camera (extrinsic parameters) that are being calculated. R and t define the position of the camera center and the camera's heading in world coordinates.

  $\begin{bmatrix} R_{3x3} & T_{3x1} \end{bmatrix}_{3x4}$

  Note that T is not the direct position of the camera, but rather the position of the origin of the world coordinate system expressed in coordinates of the camera-centered coordinate system. The position C of the camera can be expressed in world coordinates by: $C = -R^{-1}T = -R^TT$

- K is the intrinsic camera matrix that describes the projective mapping from world coordinates to pixel coordinates:

  $$K = \begin{bmatrix} x & \gamma & c_x & 0 \\ 0 & y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where:

- f_x is the focal length x-coordinate
- f_y is the focal length y-coordinate
- c_x is the optical-center (the principal-point) x-coordinate
- c_y is the optical-center (the principal-point) y-coordinate
- γ is the skew coefficient between the x and y axis

All the above yields us the following equation:

$$sp_c = s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

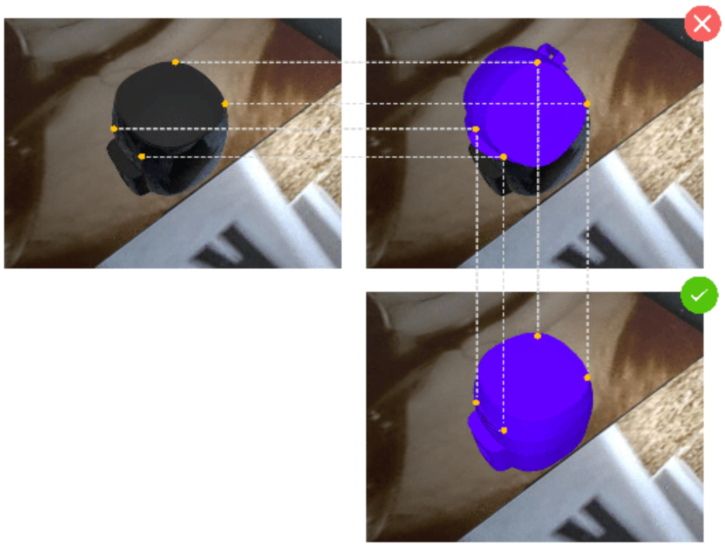


Figure 2.1: Pose ambiguity for PnP where $n = 4$
. The bottom-right image shows the correct pose. The top-right image shows how the same input can also yield other solutions due to rotational ambiguity.

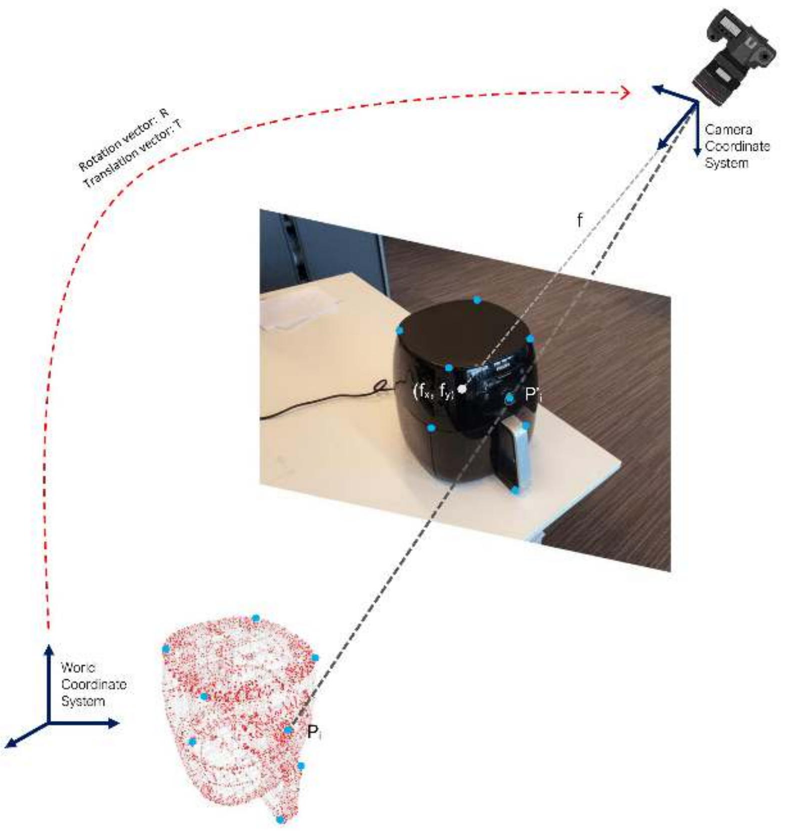


Figure 2.2: PnP Algorithm - Overview. Known 3D information from the object (from for example a CAD-model) is leveraged to calculate a rotation and translation vector from 2D predictions.

2.2 Pose Estimation - Literature Study

Within the pose estimation field, there are two main sub-fields: human pose estimation and object pose estimation. What follows is a brief discussion of the most novel and influential academic publications in both these sub-fields.

2.2.1 Human Pose Estimation

Classical Approaches

These methods utilize hand-crafted features and static graphical models. An example of this is the pictorial structures framework introduced by Fischler et. al. [18], which was made practical by Felzenszwalb et. al. [19]. An object (or human) is represented by a collection of "parts", which are arranged in a dynamic and deformable configuration. In 2011, Yang et. al. introduced a method that uses a deformable part model [20]. Rather than modeling articulation using a family of warped (rotated and foreshortened) templates, Yang introduced the use of a mixture of small, non-oriented parts. Articulations are modeled well in this approach, however at the cost of limited expressiveness and not being able to take the global context into account. Moreover, a limitation of these methods is that such a pose estimation model cannot depend on image data.

Deep Learning Approaches

With the introduction of "DeepPose" by Toshev [21] et al, research on human pose estimation began to shift from classic approaches to Deep Learning. Toshev defined pose estimation to be a

CNN-based regression problem on human body joints. Moreover, they proposed a refinement of the predictions using cascaded regressors. In 2015, Tompson[22] et. al. introduced the use of generated heatmaps (instead of directly regressing on body-parts) that represent pixel-wise prediction confidence. Tompson et. al. ran images through multiple resolution banks in parallel to simultaneously capture features at a variety of scales. In 2016, Wei[23] et. al. introduced Convolutional Pose Machines (CPM), which consists of a sequential architecture composed of convolutional networks that directly operate on belief maps from previous stages, producing increasingly refined estimates for part locations, without the need for explicit graphical model-style. CPMs are good at learning long-range spatial relationships by using sequentially increasing receptive fields. In 2016, Newell[24] et. al. introduced the ”Stacked Hourglass Network”. Their network consists of pooling and upsampling layers that resemble an hourglass when stacked together. In 2018, Xiao[25] et. al. argued that simpler networks performed as well as CPM and the Stacked Hourglass Network. Their network consisted of ResNet and a couple of deconvolutional layers (instead of skip connections that preserve information for each resolution). In 2019, Sun[26] et. al. showed that their network maintains a high-resolution representation instead of the popular high-to-low or low-to-high resolutions. Their network architecture starts from a high-resolution subnetwork, and gradually adds high-to-low resolution subnetworks and connects the multi-resolution subnetworks in parallel. Repeated multi-scale fusions are conducted by exchanging information across parallel multi-resolution subnetworks over and over through the whole process. Their method eliminates the need for intermediary heatmap supervision.

2.2.2 Object Pose Estimation

Classical Approaches

Classical approaches to 3D Pose Estimation of objects often involve local key-points and feature matching by means of local descriptors [9][27]. These methods are often fast and robust, however only in the case of texture-rich objects in high resolution images[28]. Just like classical approaches for human pose estimation, classical approaches for object pose estimation often also rely on graphical models [29].

Deep Learning Approaches

In 2017, Xiang[30] et. al. published PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. PoseCNN estimates the 3D translation of an object by localizing its center in the image and predicting its distance from the camera. The 3D rotation of the object is estimated by regressing to a quaternion representation. Moreover, a novel loss-function was introduced that enables for handling symmetric objects. Tremblay[31] et. al. from NVIDIA Research published a deep object pose estimation technology (2018) that was trained with only synthetic data. They explored the domain-gap and managed to create a one-shot network that is able to perform competitively against state-of-the-art networks. Tekin[32] et. al. proposes a Single Shot Pose Estimation method (2018) based on the popular You Only Look Once (YOLO) architecture. Although more accurate methods exist, Tekin’s method is one of the fastest algorithms, because it uses YOLO as its backbone architecture. In 2019, Li[33] et. al. published DeepIM: a novel deep neural network for 6D pose matching. Given an initial pose estimation, DeepIM is able to iteratively refine the pose by matching the rendered image against the observed image.

2.2.3 Conclusions from the Literature

Many proposed methods require (often expensive) post-processing steps that refine the solution, effectively making the solution ’multi-staged’. Ideally, the chose method should not be dependent on such post-processing steps, but rather be able to directly regress to an estimated pose and hence be ’single-stage’. Moreover, our solution requires detections to be fast; prediction needs to happen in real-time ($\geq$24 fps). Because of this requirement, Tekin’s YOLO-based method was initially chosen to be explored for our solution. If decently optimized, YOLO (and its variants) can probably be deployed on mobile devices with decent prediction performance. However, YOLO was not initially designed for the pose-estimation task, but rather for the much less complex task of 2D bounding-box regression and classification. In the 2D bounding-box regression task, it is more important to classify the detected objects rather than to predict a precise localization for the objects. YOLO is known to be very fast at the expense of accurate localization predictions, compared with other state-of-the-art methods. However, in the task of pose-prediction, localization accuracy is very important. Therefore Convolutional Pose Machines were selected as a second method. This work originally was designed for the human pose estimation task, but since CPM excels at learning long-ranged dependencies between variables, CPM would probably also work well in the case of object pose estimation. CPM is a fully convolutional network, which gives the possibility to swap out the slow classical convolutional layers for optimized convolutional modules such as MobileNet, ShuffleNet or EffNet, allowing inference to be fast and deployable on the edge.

Chapter 3

You Only Look Once

3.1 Introduction

You Only Look Once (YOLO) is a state-of-the-art, real-time object detection system. It can detect instances of semantic objects of a certain class (such as cars, bottles, or humans) in digital images and videos. Besides classifying the detected objects, YOLO determines where the object is located by predicting a 2D bounding-box. YOLO thanks its name to the methodology behind its architecture: "You Only Look Once"; the model looks only once at an image, whereas other detection algorithms (such as SSD[11] and RetinaNet[34]) go over an image multiple times at different scales. These latter 'other' detection systems repurpose classifiers and/or localizers to perform detection while YOLO directly regresses to a solution. Input images get divided into cells that each predict bounding boxes and class probabilities, which are subsequently weighed such that only high confidence predictions remain. Only having to look at the image allows for high detection speeds, while maintaining high accuracies. Yolov3 is more than 1000x faster than R-CNN and 100x faster than Fast R-CNN [10]. On a Pascal Titan X YOLO processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev [10].

What follows is a more detailed elaboration of YOLO's architecture, trainings-procedure, and performance. Subsequently, experiment results will be presented and discussed. Since YOLO's first appearance in 2015, improved versions and variants were released. To name a few:: YOLOv2, YOLOv3, YOLO9000, YOLOLite, and TinyYOLO. We will be speaking about YOLOv3 unless mentioned explicitly.

3.2 Model Architecture

YOLO features only convolutional layers (54 layers to be exact), making it a Fully Convolutional Network (FCN). An overview of YOLO's architecture is presented in Figure 3.1 It's model architecture consists of the following 3 components: residual-blocks, detection-layers, and upsample-layers. These 3 components will be elaborated below.

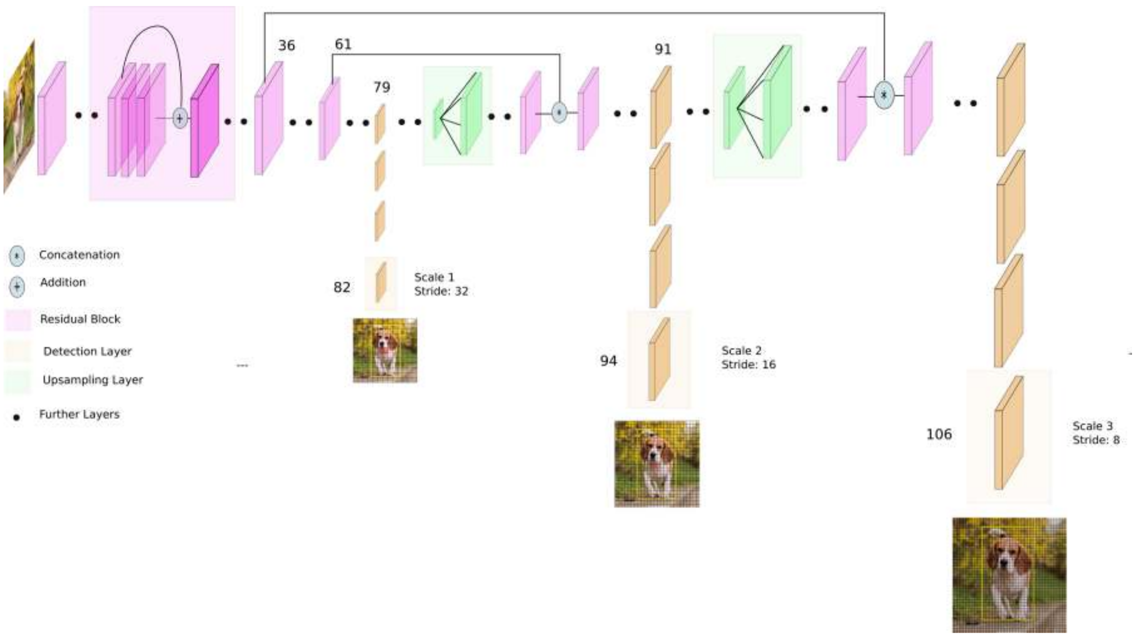


Figure 3.1: YoloV3 Architecture Overview [3]

3.2.1 Residual-block

Also called identity block, contains convolutional layers that apply a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activation called a feature map, indicating the locations and strength of a detected feature in an input, such as an image. In a residual block, the activation of one layer is fast-forwarded to another deeper layer, which is done through a 'skip-connection' (or 'shortcut')[35]. See Figure 3.2 for a schematic-overview. Instead of learning unreferenced functions, the model can now learn residual functions with reference to the layer's input, resulting in a gain in accuracy from considerably increased depth while being easier to optimize. Theoretically, training-error should gradually decrease with every layer that is added, yet in practice, a saturation point will be reached where training error starts increasing. Deeper models are more prone to vanishing or exploding gradients. Residual blocks help overcome these vanishing or exploding gradients.

3.2.2 Detector-layer

After the image traveled through various residual-blocks where convolutions are applied to its corresponding vector, the image vector gets fed to the so called detection layer. The input image is divided into a grid of S x S cells, where $S = 13$. For each grid cell c $\in S$ x S, 2 components are predicted:

1. B bounding boxes ($B = 3$ as chosen by the original authors), which each have 5 parameters:
    - x: the x-coordinate of the center of the box (or in some implementations, the bounding-box origin coordinates) relative to the origin of the grid cell.

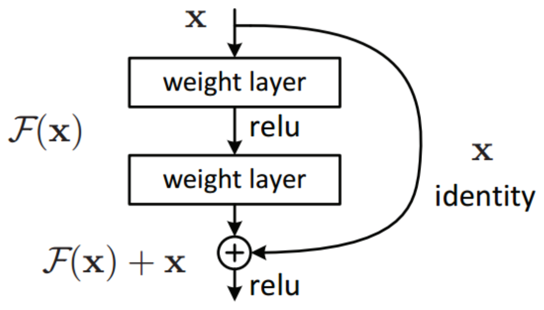


Figure 3.2: YOLO architectural elements: Residual Block

- y: the y-coordinate of the center of the box (or in some implementations, the bounding-box origin coordinates) relative to the origin of the grid cell.
- w: width of the bounding-box relative to the whole image
- h: height of the bounding-box relative to the whole image
- c: confidence value that represent the Intersection over Union (IoU) between the predicted box and any ground truth box

2. C conditional class probabilities: $Pr(Class_i|Object)$. Regardless of the number of boxes, only one set of class probabilities is predicted for that specific grid-cell. Note that since this prediction happens for every cell c in the grid S x S, it does not mean that high class probabilities for e.g. 'dog' mean that there actually is a dog in that cell present, just that IF there is an object present, it is most likely a dog. In the final step, the conditional probabilities (whether there is an object present or not) and the individual bounding-box predictions are multiplied, which yields class-specific confidence scores for each box.

Figure 3.3 depicts a schematic overview of YOLO's output tensor of a prediction in a specific grid-cell.

3.2.3 Upsampling-layer

After the detection-layer, the resulting feature map is up-sampled by a factor 2. A feature-map from earlier on in the network is also taken and concatenated with the upsampled feature-map to obtain fine-grained information from previous feature-extractions (a typical encoder-decoder architecture). Upsampling allows for obtaining more meaningful semantic information from the features.

3.2.4 Non-maximum Suppression

A method that is used to ensure that an object is only identified once. Consider we are trying to detect the dog in Figure 3.5. This dog lays in multiple cells of the 13 by 13 grid. Non-maximum Suppression ensures the optimal identification of the cell among all cell candidates to which the dog belongs. First, all candidates with probability of the object being present that are lower than some threshold (usually 0.6) are discarded. Subsequently, the cell with the largest probability of that specific object is selected and we discard any remaining cells with intersection over union greater than some threshold value (usually 0.5) with the prediction cell.

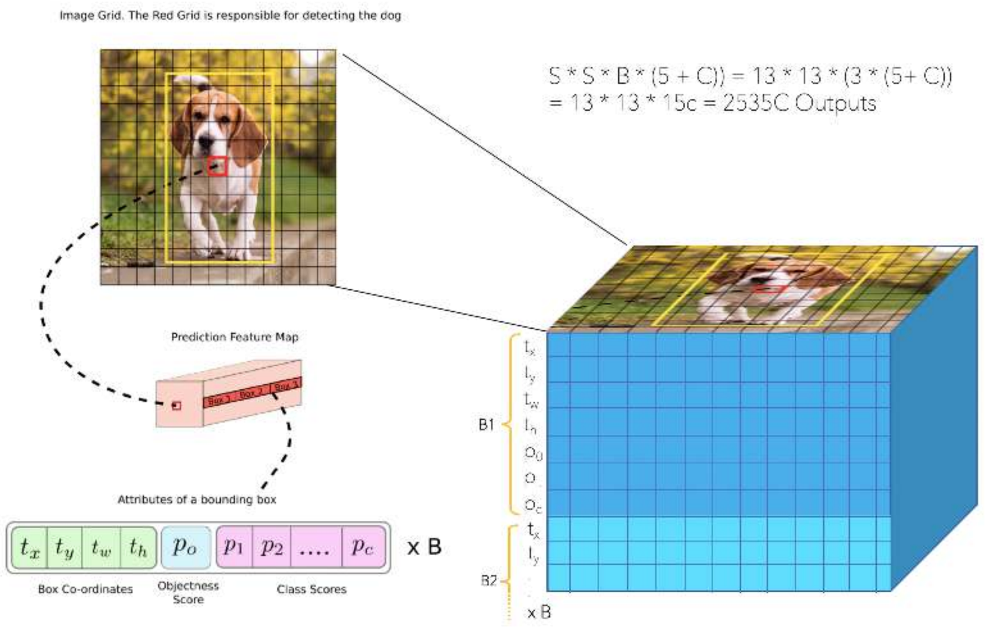


Image Grid. The Red Grid is responsible for detecting the dog
S * S * B * (5 + C)) = 13 * 13 * (3 * (5+ C))
= 13 * 13 * 15c = 2535C Outputs
Prediction Feature Map
Box 1
Box 2
Box 3
Attributes of a bounding box
t_x t_y t_w t_h p_o p_1 p_2 .... p_c x B
Box Co-ordinates
Objectness Score
Class Scores
B1
t_x
t_y
t_w
t_h
O_0
O
O_c
B2
t_x
t_y
x B

Figure 3.3: Overview of YOLO's prediction's output tensor in a grid-cell. The tensor contains predictions for $B = 3$ bounding-boxes, objectness score and class probability scores for each class [3]

.

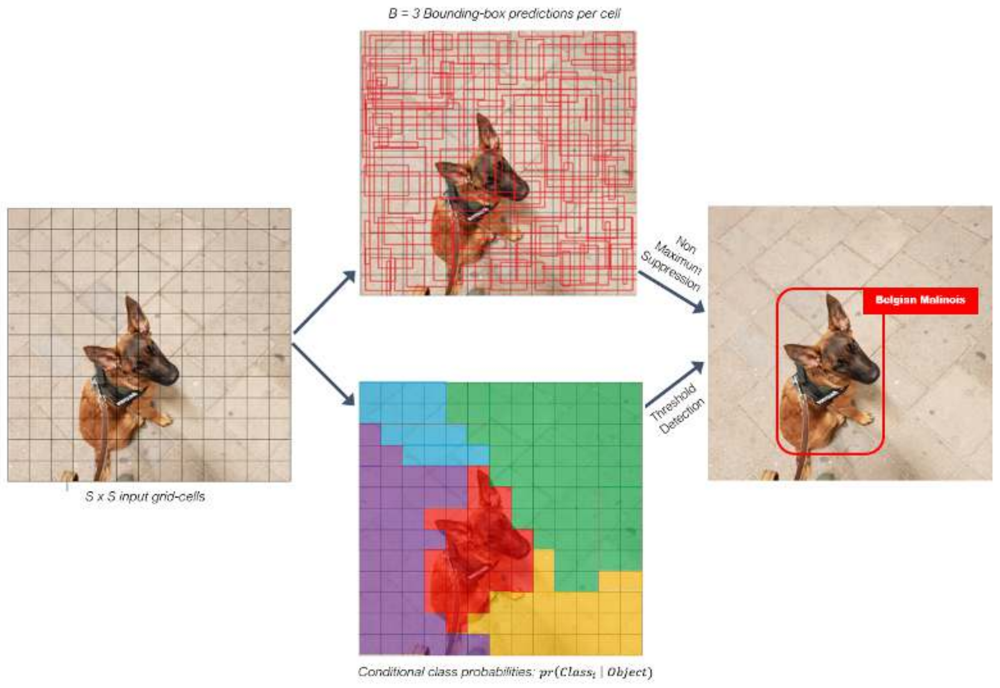

B = 3 Bounding-box predictions per cell
Non Maximum Suppression
Belgian Malinois
S x S input grid-cells
Threshold Detection
Conditional class probabilities: $pr(Class_i \mid Object)$

Figure 3.4: Overview of what happens in the prediction layer of YOLO. The input-image is divided into a S x S grid. For each grid-cell, 3 bounding-boxes and a conditional class probability score are predicted. Subsequently, by means of a threshold and non-maximum-supression, the final output prediction is determined.

3.3 Anchor Boxes

Usually, object detectors regress directly to the width and height of the bounding-box. However, in practice, this often leads to unstable gradients during training. Therefore, modern object detector methods (like YOLO) offset to pre-defined default bounding boxes called anchors or predict log-space transformed coordinates.

The original authors set the number of anchors for YOLOv3 to be 3. Anchor boxes can be seen as templates for the dimensions of the to be predicted bounding-box and are calculated a priori by means of the k-means clustering algorithm that clusters the ground-truth bounding-box annotations for the respective training-set. The network's output is transformed to obtain bounding box predictions using the following formulas:

$$b_x = \sigma(t_x) + c_x \tag{3.1}$$

$$b_y = \sigma(t_y) + c_c \tag{3.2}$$

$$b_w = p_w e^{t_w} \tag{3.3}$$

$$b_h = p_h e^{t_h} \tag{3.4}$$

Where:

- tx, ty, tw, th are outputs of the network.
- b_x and b_y are the center coordinate (x, y) (or the origin in some implementations) of the predicted bounding-box, b_h and b_w are the height and width of the predicted bounding-box.
- YOLO does not predict the absolute coordinates of the bounding box's center, but rather it predicts relative coordinates to that specific grid-cell. An offset is used to calculate the absolute position of the bounding-box with respect to the full-image. c_y and c_x are the (x,y) coordinates of the grid cell's origin (top-left), which are used as offset.
- p_w and p_h are the width and height of the anchor-box.

Figure 3.5 shows an example of the bounding-box parameters, model-output and anchor-box.

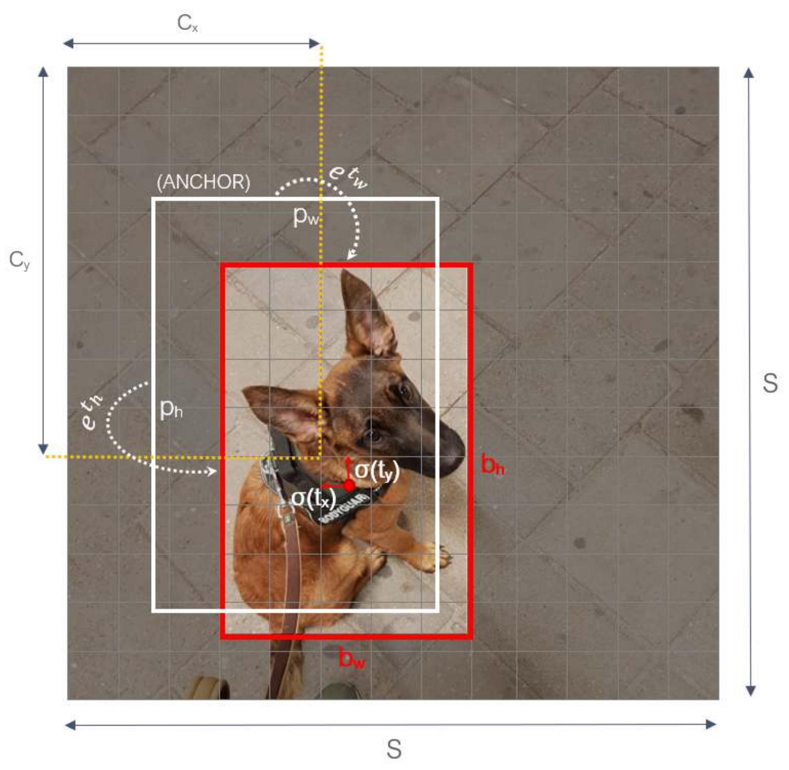


Figure 3.5: Example of the parameters necessary to calculate box coordinates for my dog Nala.

3.4 Loss Function

The loss function contains 3 components, which will be described in the following sections.

3.4.1 Localization Loss

This loss penalizes location and dimension errors of the predicted bounding-box. Only the grid-cell that is responsible for the current prediction is considered. The square root error is used, since it puts more emphasizes on bigger errors.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{3.5}$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{3.6}$$

where:

- λ_{coord} is a hyper-parameter that can be used to put more emphasize on the particular loss.
- $1_{ij}^{obj} = 1$ if the jth bounding-box in cell i is responsible for detecting the object. Otherwise this variable takes 0

3.4.2 Classification Loss

If an object is detected, the squared error of the class conditional probability is taken for each class to penalize wrong classifications.

$$+\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{3.7}$$

where:

- $1_i^{obj} = 1$ if an object appears in cell i, otherwise 0.
- $\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

3.4.3 Confidence Loss

This loss penalizes the wrong objectness of a grid-cell prediction and can be split in 2 scenarios:

False positive

An object was detected, but there is actually no object present:

$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{3.8}$$

where:

- $\hat{C}_i$ is the box confidence score of in cell i.
- $1_{ij}^{obj} = 1$ if the jth boundary box in cell i is responsbile for detecting the object, otherwise 0.

False negative:

An object was not detected, but there is acutally an object present:

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \tag{3.9}$$

where:

- $\hat{C}_i$ is the box confidence score of in cell i.
- 1_{ij}^{noobj} is the complement of 1_{ij}^{obj}.

3.4.4 Total Loss Function

The total loss function is an addition of the above three mentioned loss function components:

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
&+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{3.10}$$

Mutually exclusive classes are assumed when using softmax; an object can only belong to one class. YOLOV3 also introduces a variant for multilabel classification using independent logistic classifiers. In this variant, no Softmax is used, but rather Sigmoid. During training, binary cross-entropy is used as a loss function for the class predictions. The authors claim that no noticeable loss in performance occurs because of this.

3.5 Benefits of YOLO

- YOLO allows for real-time processing, while still maintaining good accuracies. Because of this reason, YOLO is one of the most popular computer vision detection algorithms.
- YOLO has relatively good generalization abilities, meaning it can handle unseen data when switching domains.
- Predictions are made with a single network that processes an image only once and hence can be trained in an end-to-end manner to improve accuracy
- YOLO's grid-based design enforces spacial diversity in the bounding-box predictions; a predicted bounding-box is always assigned to a clear grid-cell. In the communication domain, spatial diversity means that *"in the same space, multiple antennas are used to offer reliability"*. In the case of these bounding-box predictions, multiple bounding-box predictions are merged (offering reliability) using non-maximum suppression and assigned to the best grid-cell.

3.6 Limitations of YOLO

- In comparison with region-proposal based methods, YOLO has problems with detecting small objects and groups of objects that appear close together. Each grid-cell only makes 3 predictions and hence many objects that are grouped together won't all be detected. Small objects are often not detected because YOLO's feature maps are of a relatively low resolution when compared with other state-of-the-art architectures.
- Although YOLO generalizes well to unseen features, YOLO has trouble generalizing to unseen bounding-box configurations (dimensions and sizes), because of its anchor-based regression method.
- YOLO is not able to achieve state-of-the-art performance.

Chapter 4

Convolutional Pose Machines

4.1 Introduction

The concept of Pose Machines (PM) was first discussed by Ramakrishna et. al. [36]. Wei et. al. [23] extended this work by adding deep convolutional layers for feature extraction into each of the stages of the pose-machines. In this 2016 CVPR paper with over 700 citations, it was shown how the prediction and image feature computation modules of the traditional pose machine can be replaced by a deep convolutional architecture allowing for both image and contextual feature representations to be learned directly from data.

These 'Convolutional' Pose Machines (CPM) define a sequential prediction framework for directly learning rich implicit spatial models, without having to adhere to some graphical model. To achieve this, sequentially stacked convolutional networks operate directly on regressed belief maps from previous stages, producing increasingly refined estimates for keypoint locations. The author claims to counteract the vanishing gradient problem by its staged approach that forces intermediary supervision, thereby replenishing back-propagated gradients and conditioning the learning procedure. State-of-the-art performance was achieved on standard benchmarks including MPII, LSP and FLIC datasets.

4.2 Pose Machines

- Let $p \in P$ be a keypoint in the set of keypoints that we are interested in.
- Lets define an image I to be an array of pixels. This array of pixels is a 2D-matrix of I_{width} columns I_{height} rows.
- To refer to a specific pixel within the image matrix, we define its coordinate at x and y as $I[x][y]$

Our ultimate goal: predicting the location (pixel-coordinates) of all our keypoints in the image:

$$\forall p \in P : I[x_p][y_p]$$

A pose machine [36] consists of a sequence of multi-class predictors $g_t()$ that are trained to predict the location of each part in each level of the hierarchy. In every stage $s \in \{1, ..., S\}$, for every part $p \in P$, the classifier g_t assigns a believe value b to every pixel z in the image:

$$g_1(f_z, \psi(z, b_{t-1})) \rightarrow \{b_s^p(z)\} \qquad \forall s \in \{1, ..., S\}, \forall p \in P$$

based on:

1. Features f_z extracted from the image at location z
2. Contextual information of the neighbourhood of coordinate z that was estimated by the preceding stage's classifier g_{t-1}.

where:

- ψ is a mapping from the beliefmap b_{b-1} to context features.

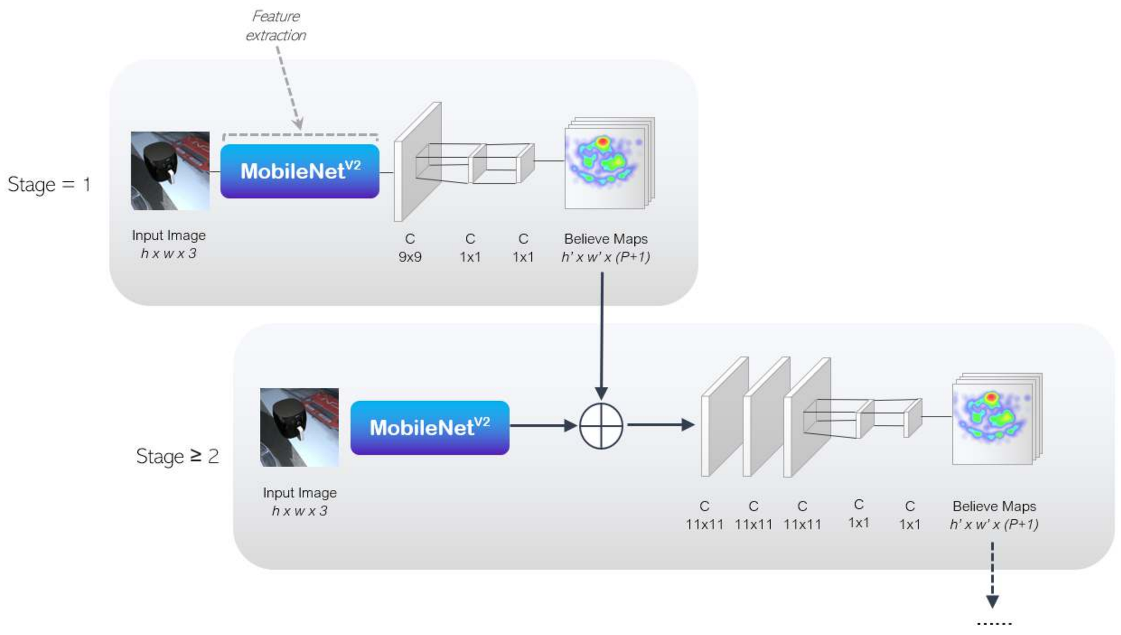


Figure 4.1: Convolutional Pose Machine - Schematic Overview.

4.3 Architecture

The original Pose Machine by Ramakrishna et. al. [36] uses boosted random forest for prediction and static hand-engineered features to capture spatial context across stages. As an improvement on the traditional Pose Machines, prediction and image feature computation modules of a pose machine can be replaced by a deep convolutional architecture. A fully convolutional architecture was used that consists out of: 5 convolutional layers + pooling, finalized by two 1x1 convolutional layers.

4.4 Long-range Spatial Dependencies

Within the field of Pose Estimation, whether this is Human Pose Estimation or Object Pose Estimation, it is of value to implicitly model long-range dependencies between variables in structured prediction tasks. For example, knowing the location of a human's torso allows for making assumptions about the location of other body parts, because the pose has to adhere to the anatomy of a human body; we know that a human foot never can be connected to a human head. The same holds for pose estimation for objects: if we know the position of one of the wheels of a car, then we know that the car's side-mirror is most likely somewhere above this wheel. Hence, we are dealing with a modeled long-range dependency between different components.

In Convolutional Pose Machines, the predictors of stages $s > 1$ can use regional spatial contextual information $\psi_{s>1}()$ from previous stages to improve predictions, by leveraging the fact that parts occur in consistent geometric configurations (anatomy of a human, the rigid-body of a car). ψ serves as a function to encode the landscape of the belief maps from the previous stage in a spatial region around the location z for all the different keypoints. No explicit function exists that calculates context features so we define ψ to be the receptive field to which the predictor is restricted to access the believe-maps from previous stages.

The receptive field is defined as the region in the input space that a particular CNN's feature is looking at (and therefore affected at). A receptive field of a feature can be described by its centroid location and radius (or size). A pixel that is closer to a receptive field's centroid, contribute more to the calculation of the output feature. The belief-maps that are generated by the first stage in the CPM are generated by a network that examines the image with small receptive fields. In subsequent stages, this receptive field is increased to allow for sequential prediction with learned spatial context features. Increasing the receptive field can be done in the following ways:

1. Adding pooling layers at the expense of model prediction precision.
2. Increasing the convolutional kernel-size at the expense of increasing the model's number of parameters.
3. Increasing the number of convolutional layers at the risk of encountering vanishing gradients during the training of the model.

The original author chose to apply method (1), adding pooling layers and method (3), increasing the number of convolutional layers.

4.5 Extending Normal Convolutional Layers

For our use-case, real-time inference on mobile devices is a requirement. However, traditional convolutional layers, such as the ones used by the original authors of CPM, are slow and often not optimized for deployment on the edge. Because of this, these traditional convolutional layers were replaced with a class of efficient models called MobileNets. MobileNets are based on a streamlined architecture that uses depth-wise separable convolutions to build light-weight deep neural networks. MobileNets introduce two simple global hyper-parameters that efficiently trade off between latency and accuracy and can be chosen by the engineer based on the constraints of the problem at hand. Please refer to the work of the original authors [37][38] for a detailed explanation of MobileNetv1 and MobileNetv2, since this falls outside of the scope of this thesis.

Chapter 5

Synthetic Data

"The world's most valuable resource is no longer oil, but data"
— The Economist [39].

5.1 Introduction

As the quote stated above might be slightly exaggerated, it does hold some parts of the truth. In the last decade, data has become a currency that only a few big players possess. Commonly these big players are referred to as the GAFAM-companies: Google, Amazon, Facebook, Apple, and Microsoft. For Google and Facebook, data is even the main driver of their core business model: collecting it, analyzing it and leveraging it. Due to the network-effect, first discussed by Carl Shapiro and Hal R. Varian [40], the value of a product or service increases with the number of others using it. Google and Facebook gain unsurpassable advantages in their data-gathering processes, due to the sheer popularity of their products. The more people use their products and services, the more data gets created, the more value it can give to its users, the more revenues can be generated.

"If you are not paying for it, you're not the customer; you're the product being sold"
— Unknown

Recent developments in open-source machine-library frameworks lowered the barrier for starting with algorithm-design. In the sense of crowd-sourcing, co-creation, and co-innovation, the GAFAM companies even open-source vast amounts of their knowledge (state-of-the-art algorithms, toy data-sets, research) for other companies and individuals to start working with. However, the accessibility problem of high-value data-sets to the public domain remains; It is because of this, that it has become vastly impossible for data-based tech start-ups to acquire a market-share alongside the big giants.

But what if data could be generated instead of obtained from the real world? What if we could just flip the switch on a little robot that generates our own data that is tailored to our specific needs and requirements? The concept of "synthetic data" can be defined as follows according to the McGraw-Hill Dictionary of Scientific and Technical Terms: *"any production data applicable to a given situation that is not obtained by direct measurement"*. In the field of computer vision, synthetic data is often generated by means of three-dimensional Computer-Aided-Design models (3D CAD models) or in the field of Natural-Language-Processing, text could be generated to train the desired classifier with. In an ideal world, this ”synthetic data” could solve all our data-problems. But unfortunately, we do not live in a perfect world just yet. The next sections will discuss the advantages and pitfalls of synthetic data.

5.2 Why Use Synthetic Data?

5.2.1 No Statistical Bias

The properties of synthetic data could be leveraged in such a way that we could circumvent any statistical bias that is often present in real data-sets. Incorrect samples that contain labeling offsets, measurement or labeling errors can be avoided when using synthetic data. Moreover, we can adapt our training-data in such a way that it is optimized for the task at hand, whether this is anomaly detection, object detection, pose-estimation, etc. The synthetic environment can be adapted in such a way that the model's performance is optimized for the required task.

5.2.2 Cost and Time Effectiveness

Deep learning models such as YOLO and SSD require large amounts of annotated training-data before a model's acceptable performance is reached [10] [11]. Annotating real data that was obtained through direct measurement is costly and time-consuming. However, In the case of synthetic data, after the data-generation-pipeline is created, a theoretically unlimited amount of perfectly accurate annotated training-data can be created. In contrast to real data, synthetic data does not require (expensive) additional pre-processing steps; the data is already clean and consistent since it was generated directly as per the requirements.

5.2.3 Anonymity

In various fields, synthetic data is being used as a filter for information (name, IP address, social security number, credit card number, etc.) that would otherwise compromise the confidentiality of its corresponding data entities. Since synthetic data is generated ad-hoc, by definition it does not contain any sensitive meta-data.

Intermezzo - Synthetic data example: autonomous vehicles

Let's say we want to train a deep object detector model for autonomous vehicles. Our model needs to detect cars and pedestrians. The firm hired a team of engineers who developed a special set of cameras and sensors that can be mounted on a car, which will drive around the city to collect data. Moreover, the firm hired a team of 8 data-collectors and rented 4 cars that were set out to collect data for a whole week in July in the city of Eindhoven.

In hindsight, it turned out to be a pretty expensive endeavor:

- Since the task at hand was detecting cars and pedestrians, all these entities had to be manually annotated a posteriori, which was very time and cost consuming.
- Another expensive post-processing step had to be performed: license-plate numbers of the cars and faces of persons that were present in the data-set had to be blurred manually to comply with confidentiality regulations.
- Although the performance of the trained object detector was great on the validation-set (which was a subset of the collected data), it turned out that the trained model did not perform well in cases where it was snowing or raining. Since the data that the model was trained with was collected in a hot week in July, the collected data did apparently not contain any snow or rain.

Statistical biases as described above can be avoided when using synthetic data since we could generate data featuring all possible weather-conditions, all types of car attributes such as color, shape, brands, etc. We could generate vast amounts of annotated data without spending many resources on it. Moreover, synthetic data can be generated in such a way that it does not include any privacy-sensitive information such as number-plates, faces of persons, etc.

5.3 Why Not Use Synthetic Data?

5.3.1 Complexity

Producing quality synthetic data is complicated because the more complex the system, the more difficult it is to keep track of all the features that need to be similar to real data. One is able to completely control the simulation environment but might face challenges introducing all (feature) variants into the dataset.

5.3.2 Degree of photo-realism

Creating synthetic-data with the same photo-realism degree as the real data is virtually impossible (or too time-consuming). With the recent advancements in computer graphics (for example, the dedicated NVIDIA RTX ray-cast enabled cards) more advancements might be achieved. Yet modeling all these photo-realistic properties is time-consuming and remains an approximation of the real world.

5.3.3 Domain-gap

By far the most considerable disadvantage of using synthetic data is the domain-gap phenomenon. When a model is trained on synthetic data, its performance on real data is lower. This will be discussed in more detail in chapter 6.

5.4 Use-cases for Synthetic Data

Nowadays, more and more researchers and companies are exploring the uses of synthetic data. Especially in domains where manual data gathering and annotation is difficult or not possible. What will follow are a few selected use-cases where synthetic data is already being successfully used.

5.4.1 Autonomous Vehicles

Addressing the fact that a sufficient amount of diverse images with class annotations is necessary for object-detection related tasks, Ros et al. [41] introduced the SYNTHIA dataset, a collection of 213,400 diverse urban computer-generated images with automatically generated class annotations. Images are generated simulating different seasons, weather and illumination conditions from multiple viewpoints. Frames include pixel-level semantic annotations and depth. Ros et. al. showed that SYNTHIA was good enough to produce accurate segmentation results and in combination with real data outperformed existing state-of-the-art solutions. NVIDIA showed that it is possible to produce a network with compelling performance using only non-artistically-generated synthetic data [42]. With additional fine-tuning on real data, their network yielded better performance than using real data alone. Others have opted for the idea of using video games such as Grand Theft Auto as the base of a pipeline for creating synthetic data by leveraging the information that is already coded into these virtual environments [43] [44]. Richter et al. even state that at a 10:1 ratio between synthetic and real-world data, the performance results are comparable to using real-world data alone.

5.4.2 Optical Character Recognition

Gupta et al. [45] presented an engine to generate synthetic images of text in clutter. Unlike previous work [46][47] , the generated text was overlayed on to existing backgrounds in a natural way, accounting for the local 3D scene geometry. Subsequently, the generated synthetic images were used to train a Fully-Convolutional Regression Network (FCRN), which was able to efficiently perform text detection and bounding-box regression. They claimed that their method significantly outperformed other methods.

5.4.3 Anomaly Detection

In the field of anomaly detection, an often occurring problem is "extremely imbalanced data".In this problem. the amount of abnormal data is so small that one cannot obtain adequate information to analyze it. Luo et al. generates artificial abnormal samples with the "Synthetic Minority Oversampling Technique", where random synthetic examples are generated in selected line segments [48]. Lopez-Rojas et al. showed that their tool "PAYSIM" was able to generate mobile money transaction data based on an original dataset [49]. With technology frameworks such as Agent-Based simulation techniques and the application of mathematical statistics, they showed that the simulated data can be as prudent as the original dataset.

5.4.4 Facial Recognition

Kortylewski et al. [50] showed that synthetic data can be used to decrease the number of real-world images needed for training deep face recognition systems. A 3D morphable face model was used for the generation of images with arbitrary amounts of facial identities and with full control over image variations, such as pose, illumination, and background.

5.5 Conclusions from the Literature

The use of synthetic data is still fairly limited and the possibilities remain far unexplored, yet some general consensuses can be established when comparing experimental results and conclusions from the literature:

1. Combining real-world data with synthetic data leads to increased performance.
2. The use of synthetic data reduces the need for real-world training-data significantly
3. Pretraining with synthetic data followed by fine-tuning with real-world data can help close the virtual-to-real domain-gap.
4. Models trained with only synthetic data that contain samples with highly randomized (sub)properties (such as strong variations in pose, illumination, background, scale, etc.), perform well across different data-sets without using domain-adaption.

A general conclusion throughout the literature is that synthetic data is not (yet) the holy grail of data-scientists, but has great potential. Synthetic data alone seems to rarely outperform real-world data, which naturally always is to be expected; a (semi)photo-realistic approximation of a real-world scene will most likely not contain the same features as the real data. The next chapter will address the issue that arises from this. Subsequently, two methods are introduced that potentially can increase the performance of computer-vision algorithms by improving the synthetic training data.

Chapter 6

Crossing the Domain-gap

6.1 Introduction

Deep learning networks have been shown to be one of the milestones in the road-map to creating the perfect machine perception system. Networks such as YOLO[10] or SSD [51] showed great learning and generalization abilities, yet suffer from poor transferability. The latter can be referred to as the challenge of domain-shift: a shift in the relationship between data collected across different domains (e.g., computer-generated vs real annotated data). Let's say we train two miscellaneous deep neural networks A and B. We keep the training-task and network architecture the same, but network A will be trained with computer-generated data, while network B will be trained with data taken from the real world. Effectively we now have 2 trained models that reside in different domains: the simulated environment $D_{simulation}$ and the real world D_{real}.

What we can observe is the domain-shift phenomenon; a domain-gap between the performance of model A and B arises, when deploying the models in their opposite domains (see Figure 6.1).

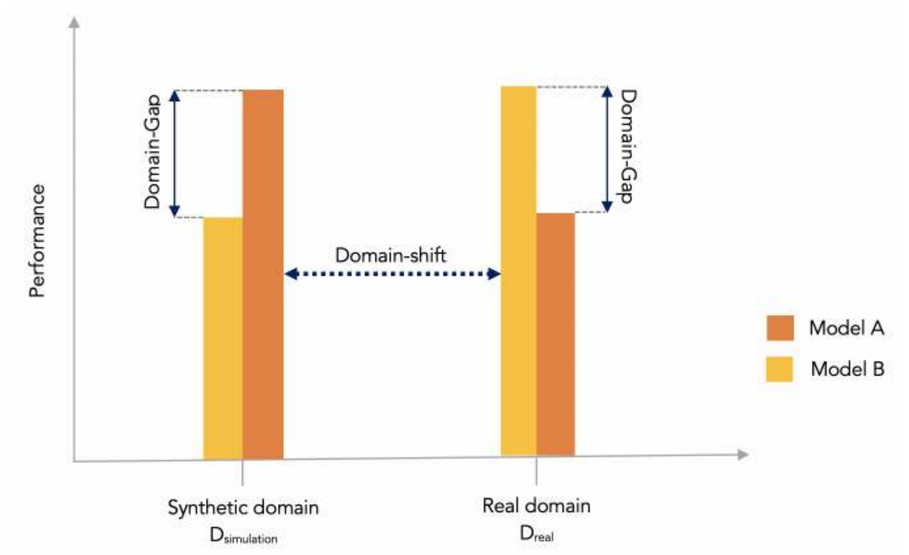


Figure 6.1: The Domain-gap - A Schematic Illustration

Various techniques exist that help with bridging the domain-gap between synthetic and real-world data, which will be elaborated in the following sections.

6.2 Domain Randomization

A technique to bridge the sim-2-real domain-gap that gained quickly in popularity is called ”Domain Randomization”, a concept first introduced by Tobin et. al. [52] in 2017. Domain Randomization attempts to create a rich distribution of training variations through data availability by introducing randomness into the data. When the synthetic data contains enough variability, the real world may appear to the model as yet another variant of what it has seen during training. Domain Randomization basically forces a model to generalize to real-world data. Within a domain, various attributes can be parametrically randomized:

- Textures of the object that is being regressed on.
- Position and orientation of the camera around the object or position and orientation of the object in 3-dimensional space
- Number and shape of distractor objects
- Position, orientation and specular characteristics of scenic illumination
- Degree of photo-realism
- Post-processing effects such as random noise and filters (e.g. lines, bloom, blur)
- Backgrounds

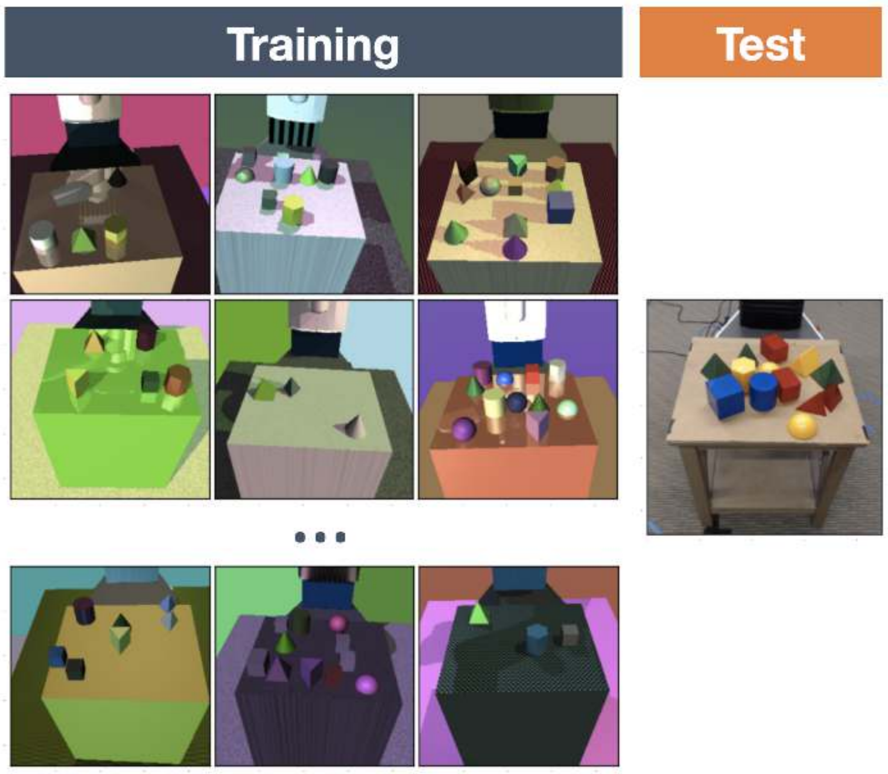


Figure 6.2: Domain Randomization - Demo illustration of approach from original authors where low-fidelity rendered images with random camera positions, lightning conditions, object positions and non-realistic textures were created in order to train a detector.

In their ablation study, the original authors report and discuss which factors are most influential to the sensitivity of the algorithm. They reported that their method is at least somewhat sensitive to all of the factors except for the use of random noise. Two important experiments and their conclusions stand out:

Experiment 1: What is the effect of the amount of domain-randomized training-images to accuracy tested on real data?

Experiment setup:

- Dependent variable: amount of domain-randomized training images
- Between 0 and 10 distractor objects were randomly placed in the scene.
- Lighting and all textures were randomized in every sample.
- Two model variants were compared: pre-trained with ImageNet images and trained from scratch.

Conclusions:

- From 5000 domain-randomized training images onward, relatively accurate real-world detection performance was achieved. Performance increased up till the use of 50.000 domain-randomized images
- Random weight initialization can achieve the same performance on real images as using pre-trained weights if provided with enough domain-randomized training-data.

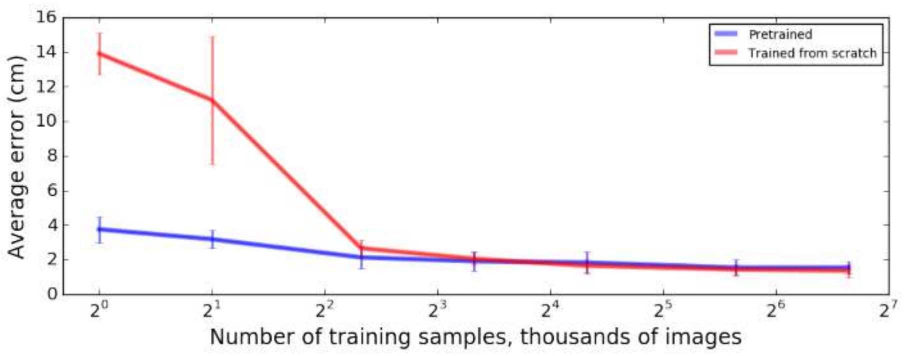


Figure 6.3: Domain Randomization - Sensitivity on real images of the number of training samples used during training

Experiment 2: What is the effect of the amount of unique textures to the accuracy tested on real data?

Experiment setup:

- Dependent variable: the amount of unique textures
- A fixed number of training-images were used: 10.000 samples

Conclusions:

- Performance degrades significantly when fewer than 1000 textures are used. Hence using a large number of textures is necessary to establish a successful transfer.
- When using a low amount of training data, texture randomization is more important than randomizing object positions and rotations: performance of using 1000 random textures during training is equal when using 1000 and 10.000 amount of training images.

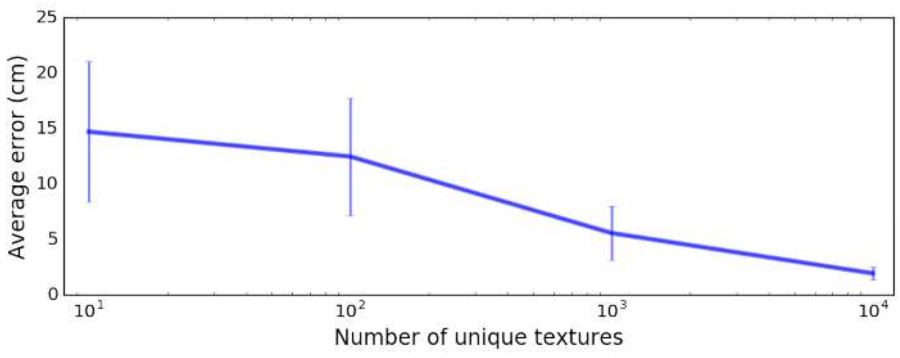


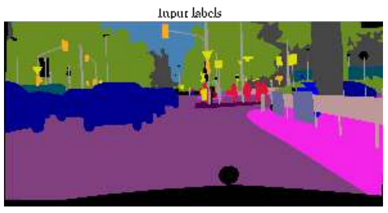
Figure 6.4: Domain Randomization - Number of unique textures

6.3 CycleGAN

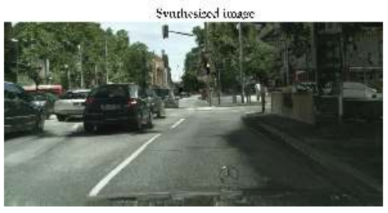
6.3.1 Introduction

A Generative Adversarial Network (GAN) is a class of machine learning systems first introduces by Ian Goodfellow et. al. in 2014 [53]. In a classical GAN, two (or more) models are being trained simultaneously in a zero-sum game: a generative model G attempts to capture the data distribution, and a discriminative model D that estimates the probability of a sample coming from the training-data rather than G. Then generative network G is trained to maximize the probability of D making a mistake, while D is trained to estimate the probability that a sample (generated by G) is real or fake.

Many exciting use-cases for GANs exist in, among others, the gaming industry, science, fashion, and advertisement. GANs are used for image enhancing with super-resolution, anomaly detection, speech-synthesis and more. GANs can be used to generate photo-realistic media which appears to be superficially authentic to a human observer. A good example of this is Pix2PixHD [4]; a framework for turning semantic label maps into photo-realistic images or synthesizing portraits from face label maps (see Figure 6.5). Yu et. al. [5] showcased the use of GANs for inpainting by using Contextual Attention (see Figure 6.6 for an example). Within the field of computer vision, Image-to-image translation is a problem that aims to learn a mapping between an input image and an output image using a training set of aligned image pairs. This form of style-transfer can be used to shift domains; e.g. 'winter to summer', 'satellite-view to maps-view', 'zebras to horses', etc. However, most GANs require paired training-data (a one-to-one mapping from source to target domain) in order to create a usable loss-function. This 'paired data' is often hard to find, since it is resource-intensive to create and in most scenarios even impossible to obtain.



(a) Annotated input data



(b) Generated Photo-realistic output

Figure 6.5: Pix2PixHD - Example of their city generation use-case [4].

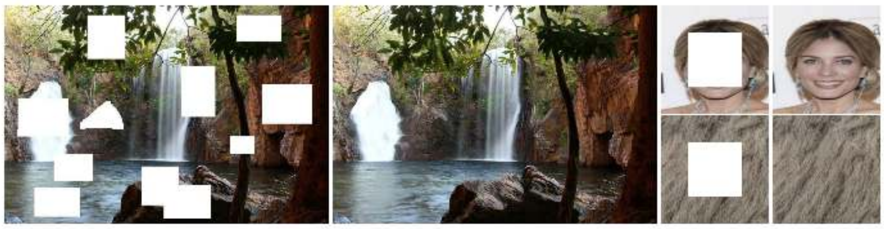


Figure 6.6: Example inpainting results of Yu's method [5]. Missing regions are shown in white. The generator is trained to fill-in the white regions.

6.3.2 Definition of CycleGAN

However, in 2018, Zhu et. al. introduced CycleGan [54]; an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired training-data. Formally the goal can be defined as:

Goal: Learn mapping functions $G : X \to Y$ and $F : Y \to X$
between two domains X and Y.

where:

- We have training samples:
  - $\{x_i\}_{i=1}^{n}$ where $x_i \in X$
  - $\{y_i\}_{j=1}^{m}$ where $y_j \in Y$
- We have 2 adversarial discriminators D_X and D_Y *where:*
  - D_X aims to distinguish between images $\{x\}$ and translated images $\{F(y)\}$.
  - D_Y aims to distinguish between images $\{y\}$ and translated images $\{G(x)\}$.

The discriminator returns a probability, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

A schematic illustration of the training phase of CycleGan to learn the mapping $G : X \to Y$ is depicted in figure Figure 6.7

6.3.3 Loss Functions

The objective task contains two types of losses:

1. **Adversarial Losses - Changes the style**
   Used to match the distribution of generated images to the data distribution of the target. Both mapping functions G and F have their own adversarial loss.

   For $G : X \to Y$ and its discriminator D_Y, the adversarial loss is defined as follows:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \frac{1}{n}\sum_{i=1}^{n}[log(1 - D_Y(G(x_i)))] + \frac{1}{m}\sum_{j=1}^{m}[log(D_Y(y_j))]$$

   where:

   - G tries to generate images $G(x)$ that look similar to images from domain Y.

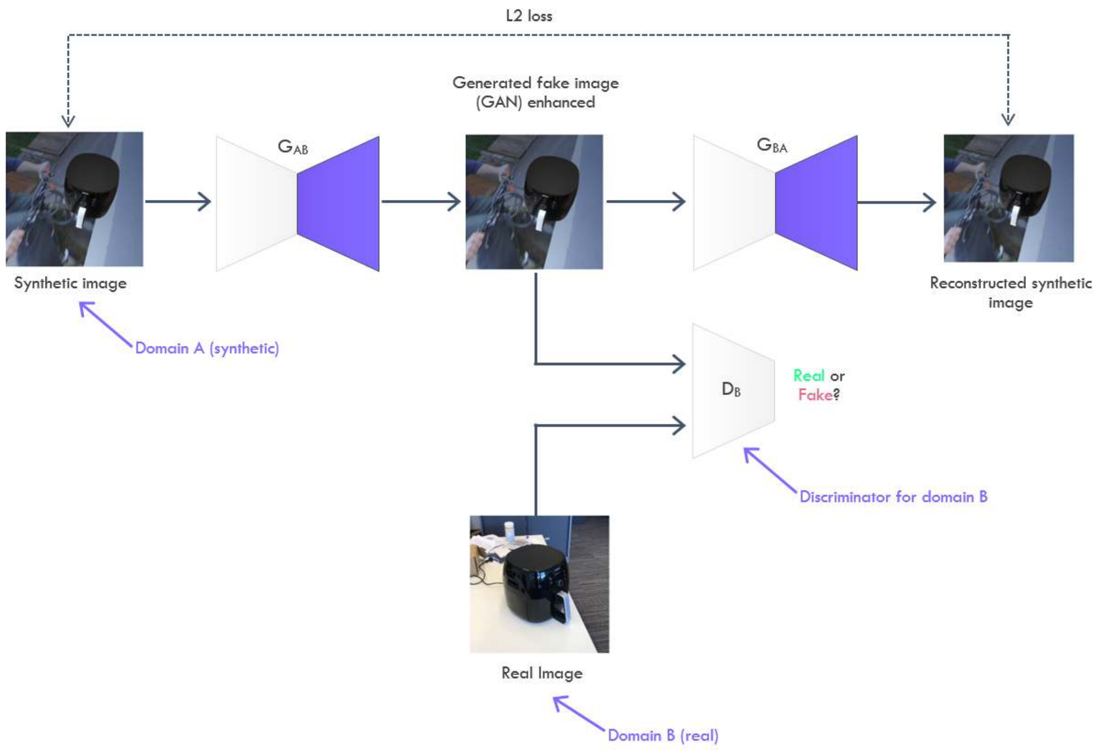

L2 loss
Generated fake image (GAN) enhanced
GAB
GBA
Synthetic image
Reconstructed synthetic image
Domain A (synthetic)
DB
Real or Fake?
Discriminator for domain B
Real Image
Domain B (real)

Figure 6.7: CylceGAN - Schematic Illustration of the training phase of learning the mapping $G : X \rightarrow Y$ where X is the synthetic data domain and y is the real data domain.

- D_y aims to distinguish between translated samples $G(x)$ and real samples y.
- G is minimized against an adversary D that maximizes $\mathcal{L}_{GAN}$:

$$\min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y)$$

A similar adversarial loss function is introduced for the function $F : Y \rightarrow X$ and its discriminator D_x: $\min_F \max_{D_X} \mathcal{L}_{GAN}(F, D_X, Y, X)$

2. **Cycle Consistency Losses - Preserve the content**
Used to prevent the learned mappings G and F from contradicting each other. It was argued by the original author that adversarial losses are not enough to guarantee that the learned functions can map an individual input x_i to a desired output y_i, since the model would be 'under-constrained'. Adversarial loss enforces the network's generated output to be f the appropriate domain but does not enforce that the input image and the generated image are still the same. A large enough network can learn incorrect mappings that yield an output distribution that matches the target distribution. The Cycle Consistency Loss addresses this. Conceptually, if the network converts an image to another domain and then back to its original domain (a cycle), the yielded output should be similar to the input. Figure 6.8 shows a schematic illustration of this loss function.

$$\mathcal{L}_{cyc}(G, F, X, Y) = \frac{1}{n}\sum_{i=1}^{n}[F(G(x_i)) - x_i] + \frac{1}{m}\sum_{j=1}^{m}[G(F(y_j)) - y_j]$$

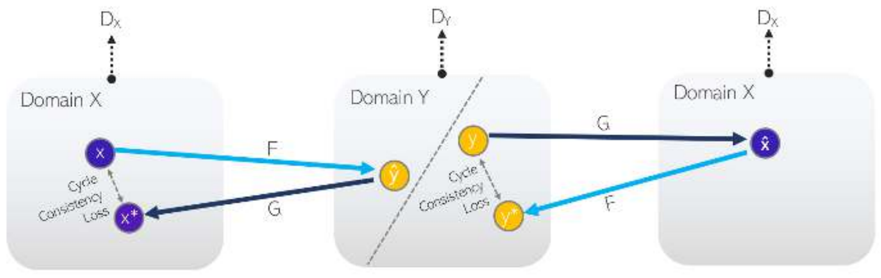


Figure 6.8: CylceGan Cycle Consistency Loss - Visual Illustration

6.3.4 Full Trainings Objective

Our full objective is defined as the sum of the cycle-consistency loss between both domains and the adversarial loss for both mapping-functions and their discriminators:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) \, + \\ \mathcal{L}_{GAN}(F, D_X, Y, X) \, + \\ \lambda \mathcal{L}_{cyc}(G, F, X, Y)\end{aligned}$$

where λ is an introduced hyper-parameter that can be used to control the relative importance of the objectives. It is suggested by the original authors to take $\lambda = 10$. We aim to solve:

$$G^*, F^* = \arg \underset{G,F}{min} \underset{D_x, D_y}{max} \mathcal{L}(G, F, D_X, D_Y)$$

6.3.5 The Intuition Behind Using GANs for our Use-case

Generating random numbers

Computers are fundamentally deterministic; it is deemed impossible for a computer to create true randomness (although the underlying question remains "what really is true randomness?"). In contrast, it is possible to design pseudo-random algorithms; the properties of their generated sequences of numbers are inherently very close to that of theoretical random number sequences. Random variables can be expressed as the result of an operation or a process. To name a few: the Inverse Transform Method, Rejection Sampling and the Metropolis-Hasting algorithm. Let's consider the Inverse Transform Method as a tool for understanding GANs and why they would work for our use-case where we are introducing photo-realistic features in our generated synthetic data.

Intermezzo - Inverse Transform Method

Goal: generate random numbers from any probability distribution by using its inverse cumulative distribution.

Let us define:

- U to be a uniform random variable over $[0, 1]$
- X to be a complex random variable

Recall that the definition of the CDF of a random variable is as follows:
A function from the originating domain of the random variable to the interval [0,1]:

$$CDF_X(x) = P(X \leq x) \qquad \in [0, 1]$$

In case of our uinform random random variable U, we have:

$$CDF_U(u) = P(U \leq u) \qquad \forall u \in [0, 1]$$

The CDF function has an inverse:

$$CDF_X^{-1}(x)$$

Lets define:

$$y = CDF_X^{-1}(U)$$

So we can proof that X and Y have the same CDF function:

$$CDF_Y(y) = P(Y \leq y) = P(CDF_X^{-1}(U) < y) = P(U \leq CDF_X(y)) = CDF_X(y)$$

By defining Y as a function of a uniform random variable, we can define a random variable with the targeted distribution of X. Inverse transform method is a method of generating random variables that follow a given distribution by making a uniform random variable go through a designed ”transform function” (see Figure 6.9 for a graphical illustration). In this case, this is the inverse CDF. The goal of this transform function is to transfer the initial probability distribution to the target distribution.

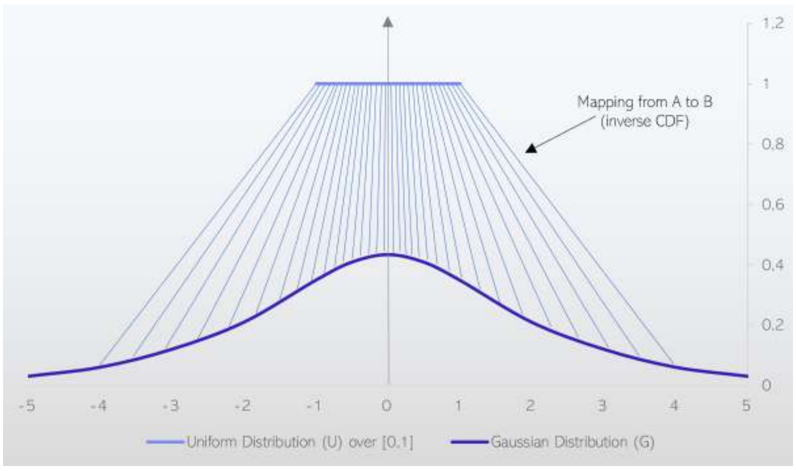


Figure 6.9: The Inverse Transform Method where an inverse CDF defines a mapping from a uniform distribution over [0,1] to a standard Gaussian distribution.

Back to GANs

Just like the Inverse Transform Method, Generative Adversarial Networks also try to tackle a set of random variable generation problems; approximating number sequences that follow a certain target distribution. In other words, matching a target distribution with a source distribution.

For the sake of simplicity, let's take a black-and-white image of a cat (a black and white image only has 1 channel, in contrast to a color-image which has 3 channels for red green and blue, which are super-positioned on top of each other). This cat image is, in essence, a single 2-dimensional pixel-matrix of size $x \times y$. We can reshape this matrix into a 1-dimensional vector that represents the cat's image.

Let's take a collection of such cat images and take their representing 1-dimensional vectors. All these n-dimensional cat vectors most likely follow a very specific probability distribution over the entire 1-dimensional vector space. In this same space, other distributions exist for birds, dogs, etc. Now the problem of generating new cats on the fly becomes the problem of creating a vector that follows the cat probability distribution over the chosen 1-dimensional space. Hence we have a random variable generation problem concerning some specific probability distribution. Two observations can be made:

1. The cat-distribution is a very complex distribution over a large space.
2. Assuming the existence of this distribution, it remains unknown how to express this distribution.

Modeling such complex functions could be achieved with neural networks where the input is a vector and the output is a transformed vector. 2 training regimes exist for such generative models:

1. **Direct:** Comparing the true and the generated probability distributions and backpropagating the loss through the network. This is done in Generative Matching Networks (GMNs).
2. **Indirect:** A generative network is trained by a downstream task (a discrimination task between true and generated samples) that enforces the generated distribution to be estimated. This is used in Generate Adversarial Networks.

Within a generative adversarial architecture, the generator is a network that models the transform function and the discriminator is another network that models a discriminative function. Doing so, parametrizes the model and its solution space. Any yielded solutions are non-optimal and bound by the capacity of these networks.

Hence, Intuitively, when switching domains using GANs, we are matching distributions of different domains. In the horses-to-zebras example, a mapping is learned to transform the distribution of vectors of horses into the distribution of vectors of zebras. The same holds for the orange-to-apple example or the winter-to-summer example. Important to note is that the existence of such distributions is assumed. Not all domains are transferable by transforming a source to a target distribution; domains need to be somewhat similar to ensure the existence of the to-be-mapped distributions [54]. In our scenario, 2 domains are defined: the synthetic domain and the real domain. It is now of interest to learn a mapping from the distribution of synthetic data to the distribution of real data (see Figure 6.10). By doing this, we conceptually will be forcing features that are present in the synthetic data to look more like the features that are present in the real data.

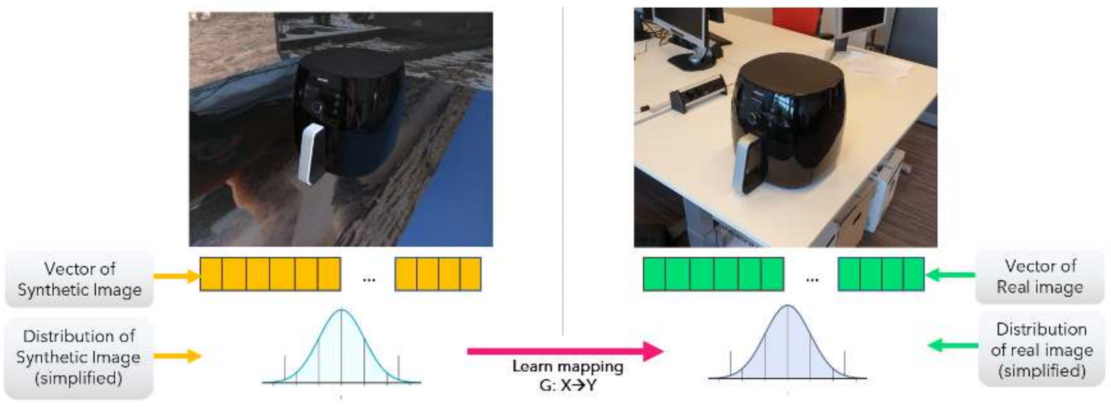

Vector of Synthetic Image
Distribution of Synthetic Image (simplified)
...
Learn mapping G: X→Y
Vector of Real image
Distribution of real image (simplified)

Figure 6.10: Style Transfer using CylceGAN - A mapping $G : X \rightarrow Y$ is learned to match the distribution of the synthetic input image with the distribution of the real data domain.

6.4 Research Question

In this section, hypothesis are stated for our research questions which are designed by a solution-driven process: a prototype concept needs to be developed of an interactive augmented reality user manual. This prototype needs to contain deep learning models that estimate the pose of various Philips consumer products. To support scalability of the solution, all computer-vision models need to be trained with only synthetic data. Synthetic data has much-unexplored potential in scenarios where labeling real data is too time and cost consuming or simply impossible due to extrinsic or intrinsic constraints and requirements. However, models trained with only synthetic data perform worse on real data due to the domain-gap phenomenon. Two techniques that could potentially positively be used to minimize this domain-gap were stated in previous sections. The aim of this research is to benchmark the effects of these two methods on the model performance in the real domain.

Recall our research question from subsection 1.2.2:

R1 **Can we enhance synthetic image training data using generative adversarial and domain randomisation methods to improve the performance of deep learning models on computer vision tasks in the real domain?**

The following hypothesis is assigned to this research question:

H1 *Enhanced synthetic training-data created with our generative adversarial method, in combination with domain randomised data, yields better performing models on computer vision tasks in the real domain.*

The next section will propose a methodology to investigate whether the hypothesis holds.

Chapter 7

Methods

This chapter will describe the strategy to answer the research question raised in subsection 1.2.2:

R1 ***Can we enhance synthetic image training data using generative adversarial and domain randomisation methods to improve the performance of deep learning models on computer vision tasks in the real domain?***

The proposed method can be split up in 3 parts:

1. **Data Generation & Collection**
   In this step, the creation of the baseline data-set ($DATASET_{BASE}$) and the domain-randomized dataset ($DATASET_{DR}$) will be discussed. A data-set consisting of manually annotated real samples ($DATASET_{REAL}$) will be introduced, which will serve as our evaluation data-set.

2. **Experiment 1 - Generative Adversarial Data Enhancement**
   In this experiment, the base synthetic data-set ($DATASET_{BASE}$) will be enhanced with the proposed generative adversarial method. First, the proposed method will be elaborated in detail. The generated dataset from this experiment will be called $DATASET_{GAN}$ and will be used in experiment 2 as one of the synthetic data types that will be benchmarked. By means of a subjective visual evaluation, it will be determined whether this generated dataset indeed became more photo-realistic.

3. **Experiment 2 - Benchmarking The Synthetic Data Types**
   In this experiment, the effects of the different types of synthetic data on model performance in the real domain will be evaluated using the following computer vision tasks and architectures:

   (a) **"Experiment 2A - Evaluation on Convolutional Pose Machines" (CPM)**
   A neural network architecture that solves a keypoint-regression task.

   (b) **"Experiment 2B - Evaluation on You Only Look Once" (YOLO)**
   A neural network architecture that solves a 2D bounding-box localization and classification task.

What will follow is a detailed description of the three method parts as introduced above.

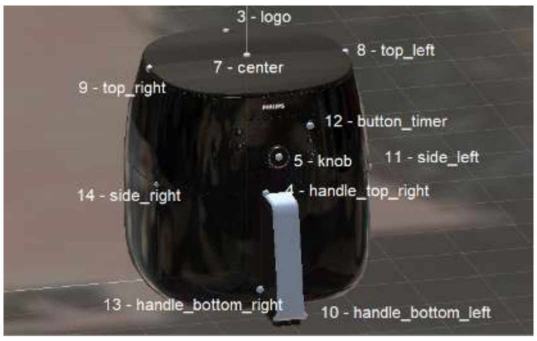
7.1 Data Generation & Collection

7.1.1 The Data Annotation Formats

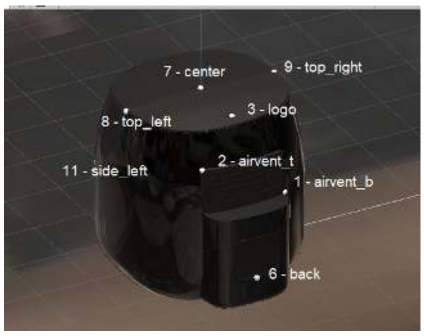
The benchmark experiment (experiment 2) will test the effect on the performance of each dataset on two different computer vision tasks. Since both tasks are inherently different, so are their annotation formats. Hence, each of the introduced data-sets have 2 different types of annotation formats:

1. **Keypoint-coordinate annotation format (CPM)**
   14 key-points (x,y) coordinates that serve as ground truth for the Convolutional Pose Machines in experiment 2A

2. **2D bounding-box for the whole air fryer object (YOLO)**
   An input-format for the YOLO algorithm in experiment 2B, which annotates a the top-left coordinate that defines the bounding-box origin and the width and height of the bounding-box.



(a) Front-view



(b) Back-view

Figure 7.1: Example visualisation of the chosen 14 key-points

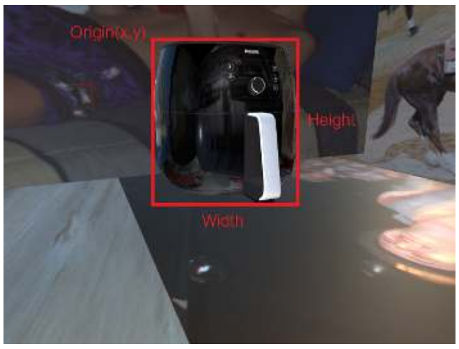


Figure 7.2: Example visualisation of a 2D bounding-box annotation

7.1.2 Synthetic Data

A data generation pipe-line was created (Philips Synthetica), which automatically outputs annotated renderings based on a computer-aided design (CAD) model. More information about Philips Synthetica can be found in appendix section C.1. The object for the computer-vision tasks was chosen to be the flagship Philips Avanche Airfryer HD9650. The following two types of synthetic data were generated:

- $Dataset_{BASE}$ - Consists of 20.000 generated images with a resolution of 256x256px. The air fryer object is modeled in a (semi-)photo-realistic way that tries to approximate the real object's material properties in terms of reflections, texture and lighting (as can be seen in Figure 7.3). This data-set is directly generated with Philips Synthetica without any post-processing steps and is being referred to as "BASE", since its the most standard form of synthetic data that will be tested. For every image: camera angle, scale and lightning-settings were randomized.

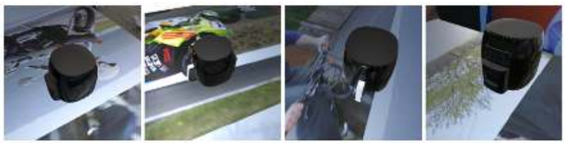


Figure 7.3: Randomly selected samples of $Dataset_{BASE}$

- $Dataset_{DR}$ - Consists of 20.000 generated images with a resolution of 256x256px. Besides the randomized camera angle, scale and lightning, now also textures were randomized according to Tobin et. al. [52]. Note that similar materials are grouped; the same type of materials used in the actual air fryer also get assigned the same randomized texture color. Please refer to Figure 7.4 for some selected samples of this dataset.

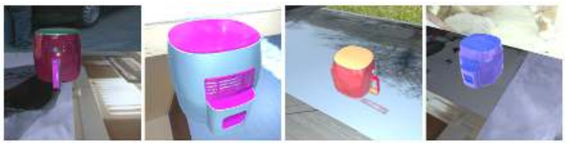


Figure 7.4: Randomly selected samples of $Dataset_{DR}$

7.1.3 Evaluation Data-set in the Real Domain

A dataset of real images of the Philips Airfryer HD9650 was created and will be used to measure the inference performance of the different models in the real domain. This dataset will be referred to as $Dataset_{REAL}$ and consists of 217 real images with a resolution of 4046x4032 px. The pictures were taken in different background settings on the Eindhoven High Tech Campus using a 2018 iPad Pro 12.9 and subsequently annotated by hand. Recall that this dataset will be used to evaluate two different computer-vision tasks with inherently different annotation formats and hence labeling had to be done twice (for each annotation format). Please refer to Appendix B for more information on how this dataset was created and annotated. See Figure 7.3 for randomly selected samples of $Dataset_{REAL}$.

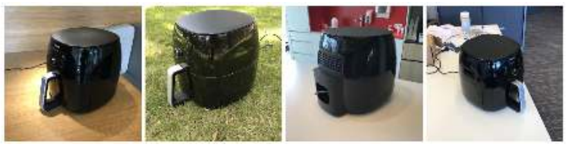


Figure 7.5: Randomly selected (cropped) samples of $Dataset_{REAL}$

7.2 Experiment 1 - Method

Generative Adversarial Data Enhancement

This section will describe the generative adversarial method for enhancing the baseline dataset ($DATASET_{BASE}$) to look more photo-realistic. A CylceGAN is trained to transfer data between the synthetic and the real domain and visa-versa. Subsequently, the trained CylceGAN is used to generate the enhanced data ($DATASET_{GAN}$). Finally, an evaluation strategy is discussed in order to determine whether the photo-realism of the generated data has increased.

7.2.1 Training CylceGAN

In order to train the CylceGAN, two unlabeled and unpaired training data-sets were created:

1. $Dataset_{BASE4GAN}$ - A data-set that consists of 950 generated synthetic images of normal BASE synthetic data (which is described in more detail in subsection 7.1.2).

2. $Dataset_{REAL4GAN}$ - A data-set that consists of 914 real pictures of the physical Philips Airfyer HD950 that were taken around the High Tech Campus Eindhoven with an iPad Pro 2018 12.9. The pictures are taken in different background settings and lighting conditions. The original images were taken at a resolution of 4046x4032px but were down-scaled to 640x480px to be able to fit into memory during training.

A CylceGAN was trained using the above training data-sets. The default network architecture and training settings were used as the 'horse2zebra' case-study, as first presented in the paper by the original authors [54]. Please refer to the original paper for more details on the used implementation of the network architecture and training details. Note that samples were generated in a fairly low resolution (256x256px), due to hardware limitation constraints. CylceGAN is a relative big architecture since it requires 4 networks (2 generators and 2 discriminators) and thus increasing the input/output resolution of the image-data results in an increased memory footprint for the GPU.

7.2.2 Dataset Creation Using CylceGAN

After training the CycleGAN and having yielded a converged model after 200 epochs, the trained model can now be used for inference. Synthetic data can now be enhanced by transforming their respective domains according to the learned distribution mapping. CycleGAN generates samples in both directions (*"Synthetic to Real"* and *Real to Synthetic*), yet we are only interested in the mapping from the synthetic to the real domain. These generated images are selected and a post-processing step is applied where a mask mapping of the object is used to cut-out only the object itself and paste this object onto a new randomly selected background from the VOC2012 dataset. This effectively changes the background of the image so that our background does not contain any transformations that are introduced by the CycleGAN (we only want the object itself to be transformed, not the background). Because of the cycle-consistency enforced by the CycleGAN, the object is still in the same position and the same pose so hence our initial annotations are still valid.

7.2.3 Evaluation Strategy

The generated dataset $DATASET_{GAN}$ will be subjected to a mere subjective visual evaluation of handpicked samples since no suitable other performance metrics exist. It will be determined if the GAN-enhanced data contains more realistic features and hence provides an overall more realistic impression than the original synthetic images.

7.3 Experiment 2 - Method

Benchmarking the Synthetic Data Types

This section describes the method used for evaluating the the effects of different types of synthetic data on the performance in the real domain. The main to be tested data-sets were introduced in section 7.1 and are repeated for completeness:

- $DATASET_{BASE}$ - Standard photo-realistic 3D renderings that serve as baseline data
- $DATASET_{DR}$ - A data-set that contains domain randomized images
- $DATASET_{GAN}$ - Enhanced $DATASET_{BASE}$ with our generative adversarial methodx

Besides training and evaluating models with data from only one synthetic data type as above, 2 additional models will be trained and evaluated with combined synthetic data types:

- $DATASET_{BASE+DR}$ - The first 10.000 samples of $DATASET_{BASE}$ and the first 10.000 samples of $DATASET_{DR}$.
- $DATASET_{GAN+DR}$ - The first 10.000 samples of $DATASET_{GAN}$ and the first 10.000 samples of $DATASET_{DR}$.

To strengthen potential findings and exclude spurious cause-effect relationships, it was chosen to investigate the validity of our hypothesis in multiple computer vision tasks and architectures, namely:

1. **"Convolutional Pose Machines" (CPM)**
   A neural network architecture that solves a keypoint-regression task.
2. **"You Only Look Once" (YOLO)**
   A neural network architecture that solves a 2D bounding-box localization and classification task.

For both of the above network architectures and coherent training-task, a quantitative method will be proposed that allows for the determination of a potential cause-effect relationship between the properties of certain selected types of synthetic data and the performance in the real domain of models trained with such synthetic data. Both experiments follow the same methodology but use different evaluation strategies. Note both experiments will not be directly compared with each other, but rather will be used to investigate the same hypothesis in different settings.

Experiment 2A involves the performance evaluation of the different types of synthetic data on the Convolutional Pose Machine architecture and training-task. Experiment 2B involves the performance evaluation of the different types of synthetic data on the YOLO architecture and training-task.

7.3.1 A Note on Data Augmentation

Augmenting image-data is a way of increasing the amount of data that can be used for training and validation by adding sample variations. Initially, a rotation, scale, crop, hue-saturation, color-balance, and flip data augmentation were applied. Augmenting data makes sense when using a relatively small training-set that consists of real data, however when using synthetic data, a theoretical amount of training-data can already be generated in a simulation environment where the engineer or domain-specialist has maximum control over the data parameters such as colors, lights, angles and positions of the objects, etc. Therefore, augmenting this data again in a post-processing step does not make much sense. Moreover, the flip-augmentation can introduce symmetrical ambiguity into the model, since the labels are flipped as well. Among researchers, the consensus is to use data augmentation during training, but considering our argumentation in the scenario of synthetic data, it was decided to turn off all post-processing image augmentation steps for both experiment 2A and 2B.

7.3.2 Experiment 2A - Evaluation on CPM

Experiment Setup

For each synthetic data-set as defined above (and combinations of these data-sets), a separate CPM model was trained using the same architecture, training-task and hyper-parameter settings (see Table 7.1). This in order to exclude other potential factors, which are not related to the synthetic data properties yet could still influence the results.

Subsequently, these independent models trained with different types of synthetic data ran inference on the manually annotated data-set of real images of the Philips Airfryer HD9650 ($DATASET_{REAL}$), effectively outputting their 2D keypoint predictions. The results of the different models then were compared with each other using the Euclidean distance metric as described in section 7.3.2. By individually comparing these different models by evaluating their performance on data in the real domain, we are effectively measure the effect of that type of synthetic data on the domain-gap. A schematic overview of the experiment method, from data-generation to evaluation, can be found in Figure 7.6.

amount_of_training_samples	20.000	Learing Rate	0.001
$input_{width}$	256	Decay Rate	0.95
$input_{height}$	256	Number of keypoints	14
scale	2	CPM Stages	6
batch_size	16		

Table 7.1: The used hyper-parameter settings for the Convolutional Pose Machine architecture that were used in experiment 2A.

Evaluation Metric

As an evaluation metric, the Euclidean distance is taken. The Euclidean distance between points a and b is the length of the line segment connecting them ($\overline{ab}$). In Cartesian coordinates, if $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n)$ are two points in Euclidean n-space, then the distance (d) from a to b, or from b to a is given by the Pythagorean formula:

$$d(\mathbf{a},\mathbf{b}) = d(\mathbf{b},\mathbf{a}) = \sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\cdots+(a_n-b_n)^2} = \sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}$$

We are predicting the (x, y) position in an image, which is a 2-dimensional plane. Hence, $n = 2$ (euclidean 2-space). In this image-plane, we have for every key-point a ground truth (x, y) pixel coordinate g and a predicted (x, y) pixel coordinate p. The Euclidean distance between this "predicted pixel coordinate" and "ground-truth pixel coordinate" can be interpreted as to how well our model is performing on the task at hand. In our scenario, we define the Euclidean distance between two 2-dimensional coordinates as follows:

$$d(\mathbf{g},\mathbf{p}) = d(\mathbf{g},\mathbf{p}) = \sqrt{(g_x-p_x)^2+(g_y-p_y)^2} = \sqrt{\sum_{d=1}^{2}(g_d-p_d)^2}$$

The mean is taken over all euclidean distances for every key-points in an image as a relative score for that image. For every image, we can then calculate the mean euclidean distance of the mean of all images, like so:

$$\frac{\sum_{i=1}^{I}\left(\frac{\sum_{k=1}^{K} d(\mathbf{g_k^i},\mathbf{p_k^i})}{K}\right)}{I}$$

where I is the number of images and K is the number of key-points. g_k^i refers to the ground-truth keypoint coordinate for the k$^{\text{th}}$ keypoint in the i$^{\text{th}}$ image and p_k^i refers to the predicted-truth keypoint coordinate for the k$^{\text{th}}$ key-point in the i$^{\text{th}}$ image.

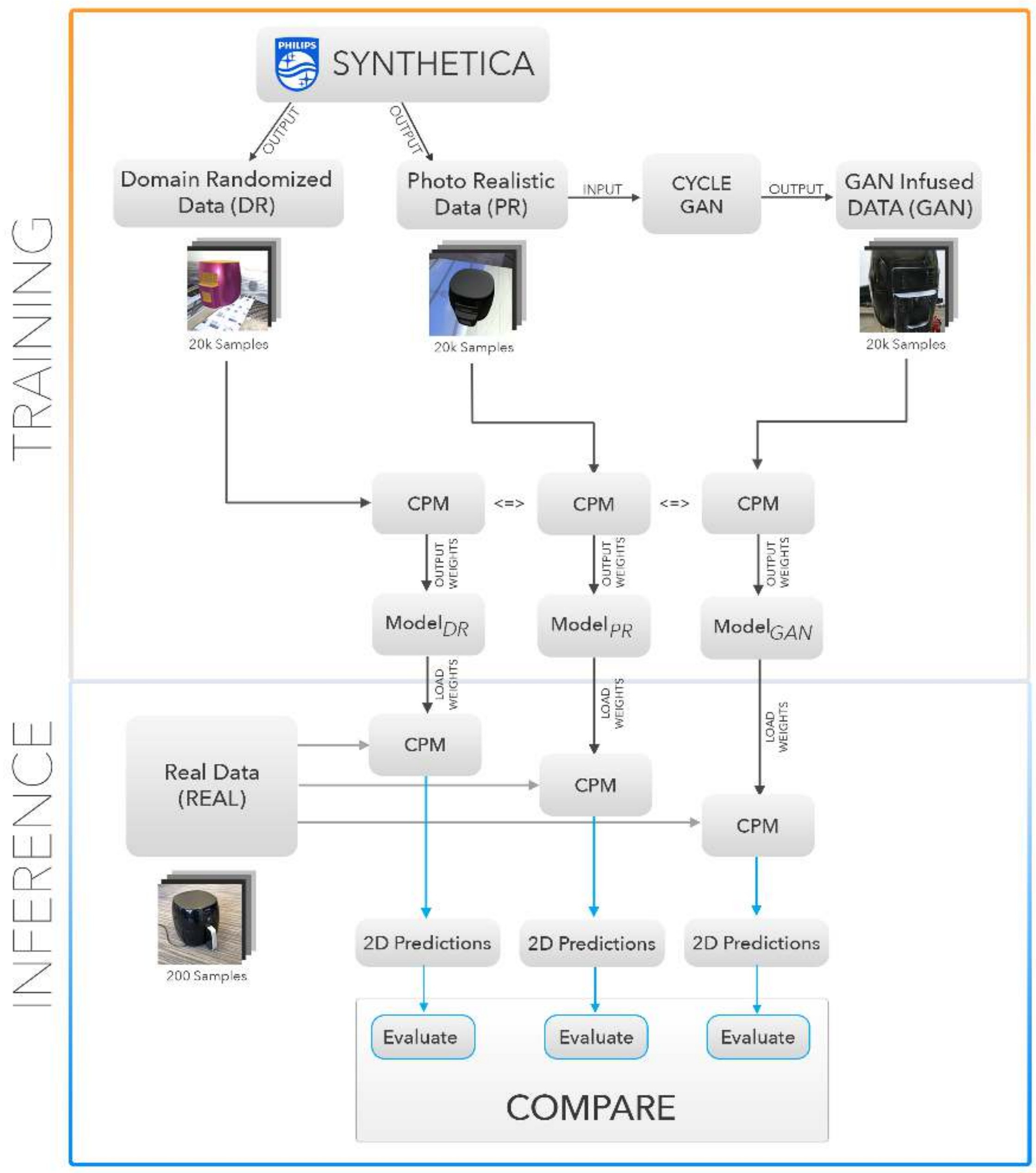


Figure 7.6: Experiment Pipeline Setup - From data generation to evaluation.

7.3.3 Experiment 2B - Evaluation on YOLO

Experiment Setup

A similar experiment setup is used as in experiment II: a separate YOLO model will be trained for each synthetic data category. Since it has been shown that YOLO can perform well even on small amounts of training-data, 2 different versions will be trained for each synthetic data category: on the full data-set (20.000 samples) and a sub-set of the data-set (5.000 samples). In experiment 2A, only one version was trained for the full data-set (20.000 samples), since CPM requires large amounts of training-data. The choice of hyper-parameters is adopted from the original authors of YOLOv3 as proposed in their original paper [10]. A 2-staged training-regime was used. For the first 50 epochs, the *darknet*53 body layers are frozen except for the 3 output layers. This is done to obtain a somewhat stable loss. For the last 50 epochs, the full body is unfrozen and the weights in all layers can now be updated. The neural network architecture, training-task, and hyper-parameter settings are kept equal for each trained model in order to exclude other factors that could potentially affect the experiment outcome. Table 7.2 shows an overview of the used hyper-parameters settings.

$input_{width}$	256	Number of classes	1
$input_{height}$	256	Validation-split	0.1
batch-size stage I	32	Learning Rate	0.001
batch-size stage II	4	Decay Rate	0.95
epochs/stage	50		

Table 7.2: The used hyper-parameter settings for the YOLOv3 architecture that was used in experiment 2B.

Evaluation Metric

During inference on the test images, the YOLO model outputs a predicted bounding-box (a corner-coordinate and bounding-box height and width) and classifies it to belong to some class $c \in C$. This predicted bounding-box can be compared with the ground-truth bounding-box via the Intersection over Union (or Jaccard) metric, which measures the similarity between finite samples sets and is defined as the size of the intersection divided by the size of the union of the sample sets. Let us define A to be the set of pixels that belong to the predicted bounding-box and let B be the set of pixels that belong to the ground-truth bounding-box. Formally, IoU is defined as:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

We define a threshold $\lambda = 0.5$ (adapted from the COCO paper [55]) and define a prediction to be positive when $IoU > \lambda$. A positively predicted bounding-box can be:

- True Positive TP(c): a proposal was made for an object of class c and there actually is an object of class c.
- False Positive FP(c): a proposal was made for an object of class c, but there is no object of class c.

Over a total data-set, we can now take the average precision of a class c:

$$AP(c) = \frac{TP(c)}{TP(c) + FP(c)}$$

The final metric that will be used to evaluate the model's performance is the mean average precision (mAP) over all queries Q (in this case, Q is the number of classes):

$$mAP = \frac{\sum_{Q}^{q=1} AP(q)}{Q}$$

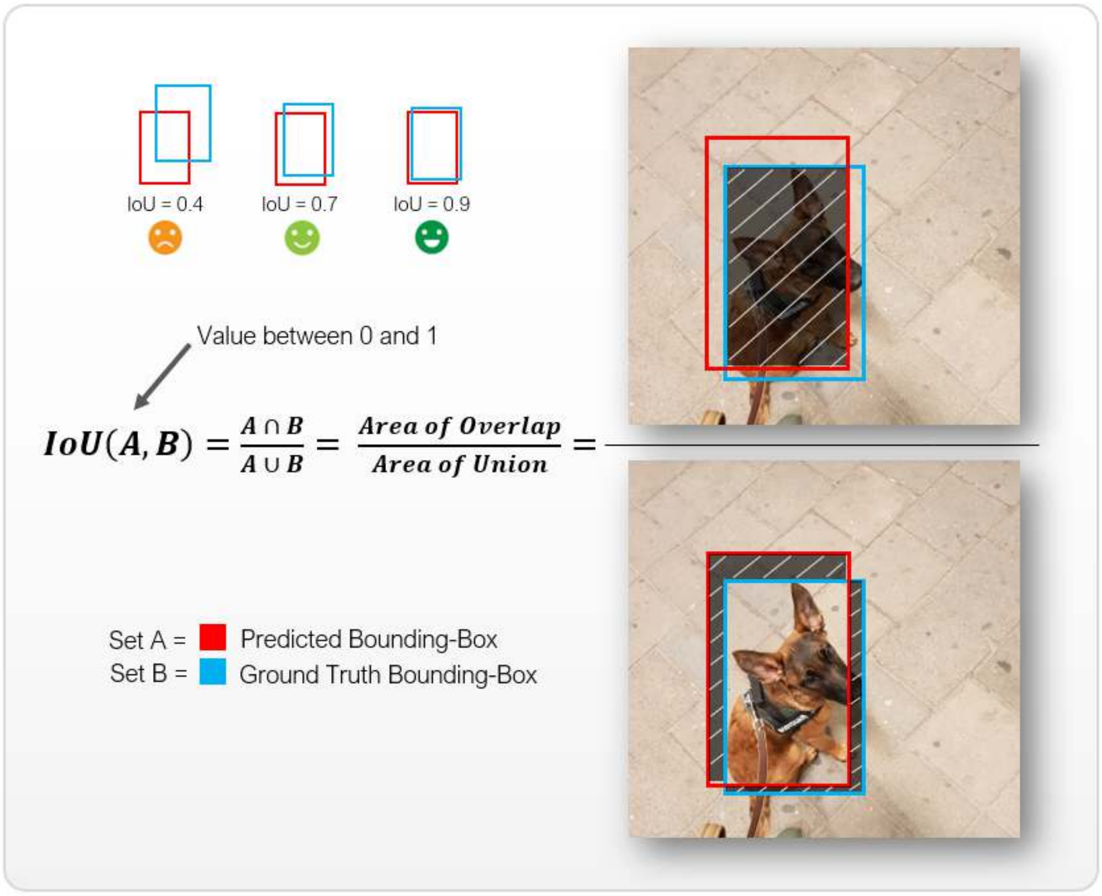

IoU = 0.4
IoU = 0.7
IoU = 0.9
Value between 0 and 1
IoU(A, B) = A ∩ B / A ∪ B = Area of Overlap / Area of Union =
Set A = 🟥 Predicted Bounding-Box
Set B = 🟦 Ground Truth Bounding-Box

Figure 7.7: Illustration of the Intersection over Union (IoU) performance metric on the ground-truth and predicted bounding-box around my Belgian Malinois "Nala". IoU takes a range $< 0, 1 >$, where 1 is an ideal match where both boxes are completely overlapping.

Chapter 8

Experiment Results

8.1 Experiment 1 - Results & Discussion

Generative Adversarial Data Enhancement

This section will present the results of experiment 1. First, handpicked samples will be presented. Subsequently, some encountered artifacts in the generated images will be discussed. Finally, a visual analysis will be made of elements in the generated images that became noticeably more photo-realistic.

8.1.1 Handpicked Generated Images

Figure 8.1 shows some randomly selected samples of the original synthetic data (left) and the same synthetic data but then improved by the CycleGAN (middle). The most right grid are handpicked real samples that somewhat match the pose of the object in the synthetic data.

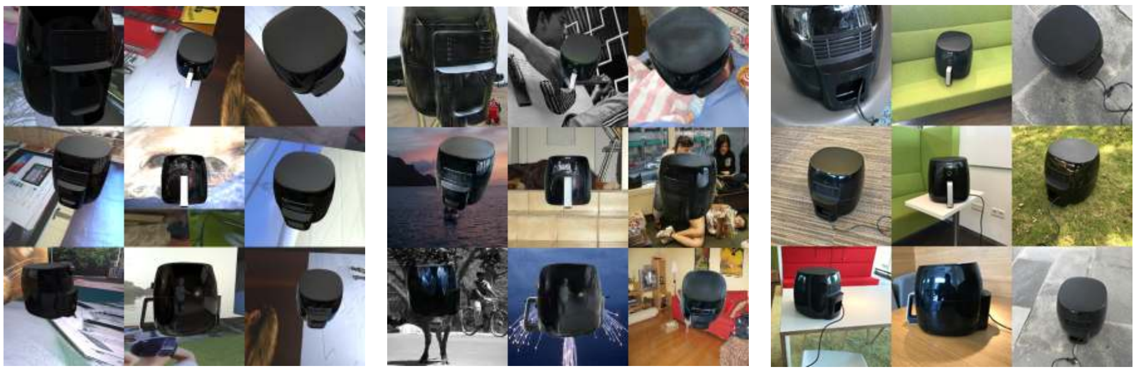


Figure 8.1: Comparison of handpicked synthetic input data (left) with generated data by the CycleGAN of the same synthetic data (middle) and handpicked unpaired real data (right) for reference.

8.1.2 Encountered Artifacts

As training-data is unpaired, it was expected to encounter artifacts introduced by the generator of the CycleGAN. Two common artifacts that were encountered are the following:

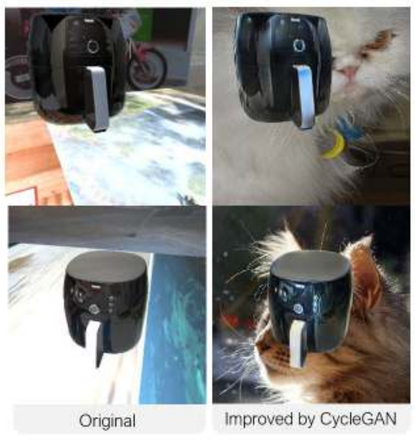
- ***Blurry edges around the object***
  Since data is unpaired, it is harder to learn a distribution mapping that focuses on the correct segment of the image (which is the object that is present in the image). If an input image has features in the background that can also be found in the object itself, edges of the object can become blurry and indistinct. This could be caused by the training-data ($Dataset_{REAL4GAN}$) not having enough distinct backgrounds and hence the learned mapping is not optimally cycle-consistent during inference. This problem is somewhat counteracted with the mask-cutting technique which was described in the method-section.

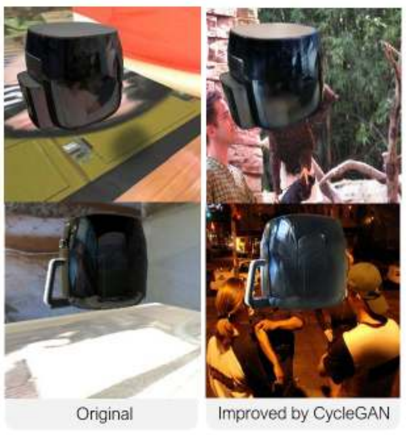
- ***Merged backgrounds***
  Since the training-set consisting of real images ($Dataset_{REAL4GAN}$) most likely does not have enough background variation (multiple training-images are taken in the same environment and hence have the same background), unwanted artifacts sometimes are introduced in the output by the generator. A clear example of this is the occasional introduction of artificial grass in the generated image. Since there are many real training-images where the object is placed on grass, CylceGAN has interpreted that the grass is part of the to be learned distribution mapping, which is unwanted behavior. Adding more variation in the CycleGAN training-set will most likely solve this issue.

8.1.3 Visual Evaluation

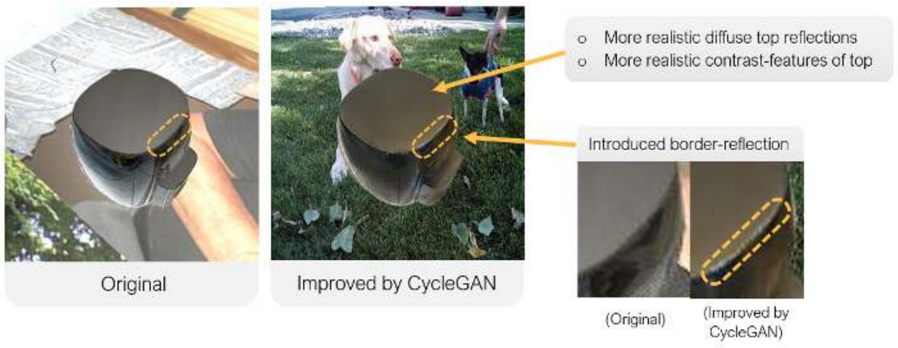
Three main observations can be made about the generated samples in terms of 'increased photo-realism levels':

1. ***Spectral reflections become more vivid and bright***.
   In general, the reflections that are simulated in the synthetic data by means of a reflection probe that ray-casts the surrounding picture wall, are too dull. In normal day light-conditions, reflections from light-sources in the real air fryer object behave more similar to a mirror; not much spectral brilliance is lost in reflections. Note that these reflections can be considered to be random noise/entropy that the computer-vision algorithms is not supposed to use in its detection process. Any computer-vision models that will be tasked with detecting the air fryer object should not learn to detect the air fryer based on the features that are present in the reflections, but rather use other features that are more consistent.

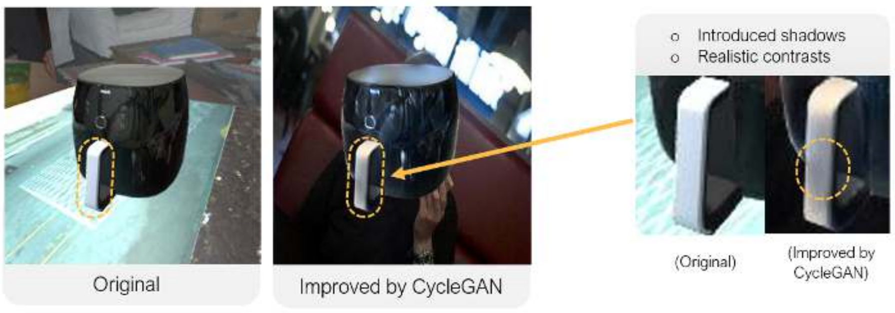




2. ***The top of the air fryer becomes notably more realistic.***
   The top of the real air fryer has diffuse light-reflective characteristics, which are not quite modeled in the original synthetic data. The CycleGAN seemingly introduces more realistic contrasts. Moreover, the CycleGAN introduces a border-reflection which is present in the real object, but not modeled in the synthetic environment.


More realistic diffuse top reflections
More realistic contrast-features of top
Introduced border-reflection
Original
Improved by CycleGAN
(Original)
(Improved by CycleGAN)

3. ***The front-handle of the air fryer becomes more contrast rich.***
   (Self-)cast shadows are turned off in the Philips Synthetica simulation environment, hence the original synthetic data does not contain any form of contrast changes due to cast shadows. It appears that contrast changes are introduced by the CycleGAN that approximate the look of cast shadows around the borders of the handle. CycleGAN appears to emphasize the photo-reflectively of the handle by adding contrast differences that emulate diffuse light characteristics that the handle of the real air fryer contains.



This subjective visual analysis implies that the GAN enhanced data indeed became more photo-realistic. Although no objective measurement for this exists, it can be argued that the photo-realism level visibly increases when the distribution of the original synthetic data is transformed into the data distribution of real air fryers. Hence it makes sense to benchmark the performance of this GAN enhanced data in experiment 2A and experiment 2B on their respective computer-vision task and algorithm.

8.2 Experiment 2 - Results & Discussion

Benchmarking the Synthetic Data Types

This section will present the results and discussion of experiment 2. Recall that this experiment was split up in two parts: 2A and 2B. Experiment 2A tests the hypothesis on the CPM architecture and training-task and experiment 2B tests the hypothesis on the YOLO architecture and training-task as described in the methods section. The results of experiment 2A and experiment 2B will be presented separately. Subsequently, the results of both experiments will be discussed together with reference to the hypothesis and research question.

8.2.1 Results of Experiment 2A - Convolutional Pose Machines

Table 8.1 contains descriptive statistics of the Euclidean distance errors of each model on the evaluation data-set. Figure 8.2 shows an overview of the mean Euclidean distance error between the predicted coordinates and the ground-truth coordinates over all images in the test-set (real annotated images), grouped by synthetic data category. Figure 8.3 presents a violin-plot of the same data.

Table 8.1: Descriptive statistics of the mean Euclidean distance score per synthetic data category for the CPM experiment

	GAN + DR	GAN	DR + BASE	BASE	DR
Mean	154.837	172.763	174.437	345.595	514.200
Std. Deviation	213.067	243.084	301.741	459.345	388.580
Minimum	34.000	45.000	29.000	32.000	27.000
Maximum	1722.000	1544.000	2400.000	2588.000	1530.000

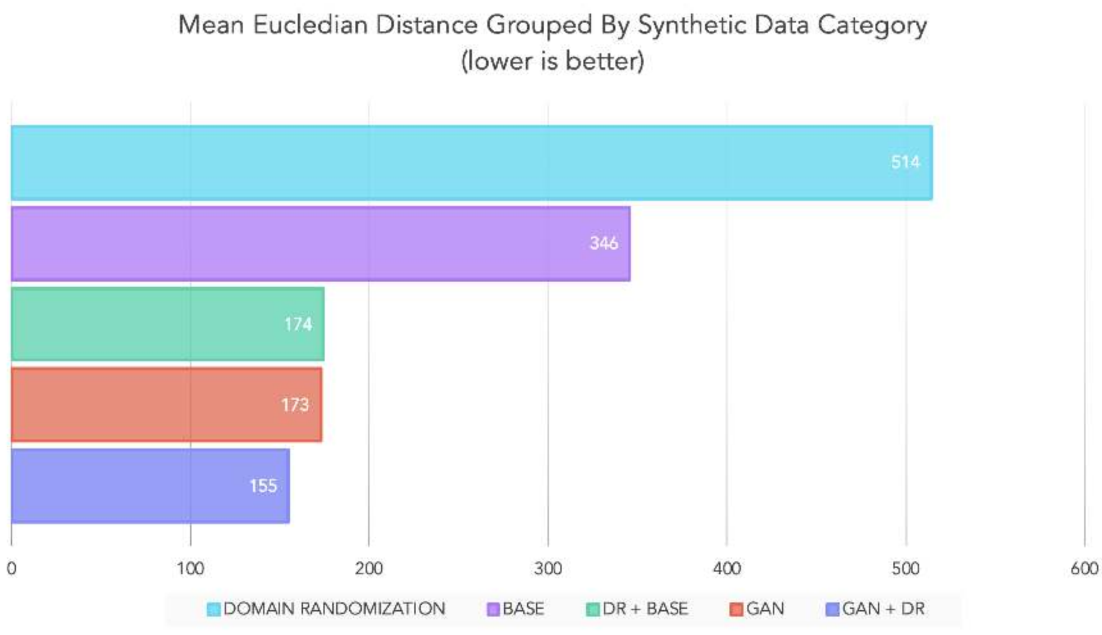


Figure 8.2: Results for the different instances of Convolutional Pose Machines trained with different types of data. The mean Euclidean distance error of all real test-images is depicted per synthetic data-type category.

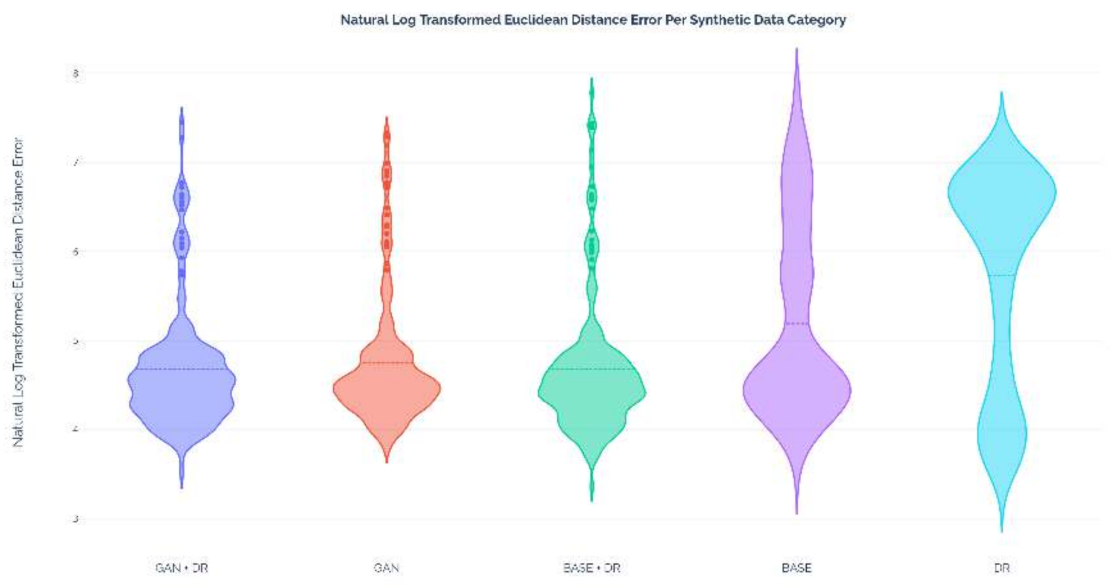


Figure 8.3: Violin-plot of the experiment results grouped per synthetic data category. Each data-point represents the mean average Euclidean distance for a specific test-image in the test data-set (which is the mean euclidean distance between the ground truth and predicted coordinates of all the 14 key-points in that image).

8.2.2 Results of Experiment 2B - YOLO

Figure 8.4 shows the mAP score of different models trained with different types of synthetic data on the test dataset that consists of real annotated images.

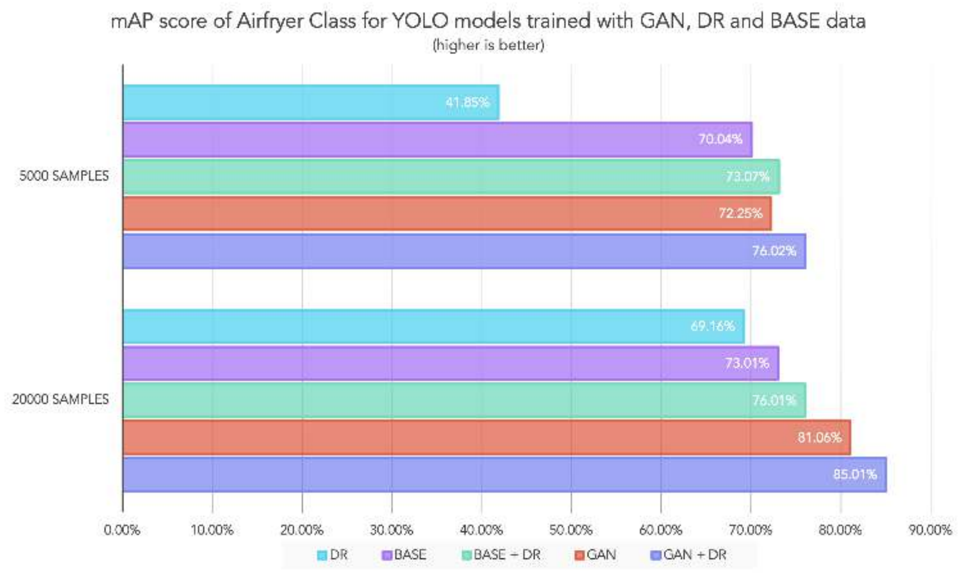


Figure 8.4: Results for the different instances of YOLO models trained with different types of data. The mAP score is depicted per synthetic data type category for data-sets of 20k and 5k samples. A test split of 10% was used.

8.2.3 Experiment 2 - Discussion

Please note that for readability reasons we sometimes refer to a specific model trained with a specific type of data as $Model_{DATATYPE}$. For example, "a model trained with only GAN-enhanced data" is written down like: "$Model_{GAN}$". Recall from subsection 7.1.2 that the BASE data-set is the standard (semi) photo-realistic synthetic data variant where it was attempted to create semi-realistic 3D renderings where environmental reflections that occur in the material of the air fryer are modeled via ray-casting. This "BASE" data-set acts like a "baseline" for the experiment.

Domain Randomized Synthetic Data

Both in the CPM as the YOLO experiment, the models trained with only domain-randomized data performed by far the worst in their respective computer vision tasks in the real domain. Nevertheless, it is still very exciting to see that computer vision algorithms can still achieve decent performance while having only seen very randomized data which is in essence quite inconsistent with how the real object looks like. Note from Table 8.1 that the CPM models that were trained with data that includes domain randomized data have a relative high Euclidean distance standard deviation when compared to models trained with BASE and GAN enhanced data. The idea behind domain randomization of training-data is enforcing the introduction of robustness and generalization capabilities by exposing the model to many variances of the actual object, effectively teaching it features that are not depending on textures. A high standard deviation of the error metric indicates that the mean error scores for individual images are spread out over a wider range, which the violin plot (Figure 8.3) indeed confirms. It can be argued that the observed significantly lower performance of the model that is trained on only domain-randomized data is a results of the data being simply 'too randomized'. It is good to introduce robustness via randomization, however, some similarity with the actual physical appearance of the object is most likely required. For example, our data-generation pipe-line assigns random colors to the materials of the object via a pseudo-random number generator. In this case, the assigned color was a randomized RGB value that effectively is made up out of 3 parameters in the range of 0 to 255 for its '**r**ed', '**g**reen' and '**b**lue' component. The RGB value for the color 'black' has values 0 for '**r**ed', '**g**reen' and '**b**lue'. The probability that a material of the air fryer is assigned a black-ish color is quite low since all 3 values of the RGB component need to approximate 0. Hence the domain-randomized dataset does not contain many black air fryers (if any) and does therefore not represent the data distribution of the real air fryer. These results imply that using only domain randomized data always performs worse than data that uses material textures that the real object actually contains.

GAN-Enhanced Synthetic Data

From the results of both the YOLO and CPM experiment, it can be observed that $Model_{GAN}$ significantly outperforms both $Model_{BASE}$ and $Model_{DR}$. In the CPM experiment, $Model_{GAN}$ has a 2 times lower mean Euclidean distance error on our evaluation dataset than $Model_{BASE}$, and a 3 times lower mean Euclidean distance error than $Model_{DR}$. In the YOLO experiment, similar results can be observed: $Model_{GAN}$ performs the best, $Model_{BASE}$ performs second-best, and $Model_{DR}$ performs the worst. These results imply that a matching distribution between real images and their synthetic counterparts can lead to increased performance on computer-vision tasks.

Combining BASE Data with Domain Randomized Data

From the results it was concluded that models trained with only domain randomized data perform worse than models trained with only BASE or only GAN-enhanced data. An argument for this was given, stating that the to be detected objects in the training-set should at least somewhat resemble the actual physical appearance of the object it will encounter during inference. As a continuation of this statement, a claim can be made that domain-randomized data needs to be combined with data that is (at least somewhat) similar to the actual data. Doing so can provide the model with, not only very randomized data, but also data that strongly resembles the actual data that will be encountered during inference. This claim is enforced by the observation that when combining domain randomized data with BASE data, significant performance increase over models trained with only BASE data can be observed. In the CPM experiment, $Model_{BASE+DR}$ performed almost twice as good as $Model_{BASE}$ and almost 3 times better than $Model_{DR}$. Similar in the YOLO experiment with 20.0000 samples, $Model_{BASE+DR}$ performed better ($76,01\%$ mAP) than $Model_{BASE}$ ($73,01\%$ mAP) and $Model_{DR}$ ($69,16\%$ mAP).

The YOLO experiment shows similar results as the CPM experiment in term of similar relative performance between the different models. An exception is $Model_{BASE+DR}$, which slightly outperforms $Model_{GAN}$ in the variant that was trained with only 5.000 images. In the variant that was trained with 20.000 training images, the $Model_{GAN}$ outperforms the $Model_{BASE+DR}$, which is consistent with the CPM experiment results. As discussed in subsection 8.1.2, the GAN enhanced data occasionally contains unwanted artifacts that are introduced by the generator of the CycleGAN. Most likely when the algorithm is provided with more data during training, noisy data (caused by unwanted artifacts) is overshadowed by the larger amount of data points.

Combining GAN-enhanced Data with Domain Randomized Data

Even higher model test performance was achieved when combining a subset of GAN-enhanced data with a subset of domain randomized data and training a new model with this combined dataset. The resulting increased performance was expected, since a performance increase was already observed when combining BASE data with domain randomized data, implying that adding domain-randomized data as a subset is benevolent to the task-performance. Since it was also observed that a model trained with GAN-enhanced data performs better than a model trained with just BASE data, combining GAN-enhanced data with domain randomized data will intuitively outperform a model trained with just GAN-enhanced data.

In other words, *we know from earlier observation that:*

- Performance($Model_{GAN}$) >> Performance($Model_{BASE}$)
- Performance($Model_{BASE+DR}$) >> Performance($Model_{BASE}$)

Therefore it was expected that:

- Performance($Model_{GAN+DR}$) >> Performance($Model_{GAN}$)

Outliers In Experiment 2A (CPM)

From the violin-plot in Figure 8.3, it can be observed that the mean Euclidean distance error for some images in the evaluation dataset yield outliers (any data point more than 1.5 interquartile ranges) for $Model_{GAN+DR}$, $Model_{GAN}$ and $Model_{BASE+DR}$. A comparison of the outliers between these 3 models is not necessarily insightful. Some images yield reoccurring outliers that occur in inference results of all the models, which could imply that these specific test-images contain (ambiguous) poses or scales that were not encountered during training. For example, the object could be placed too far away from the camera or an image contains too much noise (for example, motion-blur, out-of-focus) for any model to make accurate predictions. In the evaluation results of $Model_{BASE}$ (Figure 8.3), a thicker top tail can be observed. Samples within this tail are not considered to be outliers anymore. Since the result analysis is an inter-comparison between the various models, it was decided not to remove any consistent outliers.

8.3 Answering the Research Question

From the visual analysis in experiment I, it was concluded that increased photo-realism levels can be achieved on synthetic data with the proposed generative adversarial method; the by CylceGAN enhanced dataset was visually confirmed to be more realistic. Subsequently, the various type of synthetic data were tested on different architectures and computer-vision tasks and evaluated using different evaluation strategies. Both experiment II and III yielded similar results, showing that using the GAN enhanced data from experiment 1 as training-data indeed does yield better performing computer vision models (YOLO and CPM), than the original training-data.

Moreover we showed that adding domain randomized data improves the general model performance on data in the real domain. This is an insight that complements the work of the original authors [52], who specifically tested the effects of domain randomization in the sole presence of domain randomized data. The original authors presented experiment results that showed that *"when the number of unique textures increases, performance increases"*. Besides using domain-randomized data as a way to create a more robust model, domain-randomized data could also be used as a form of task-specific pre-training. Subsequently, real or photo-realistic synthetic data could be used to fine-tune the model.

Chapter 9

Proof of Concept Prototype

This chapter will present the developed prototype. First, some implementation details will be elaborated briefly after which the resulting prototype will be discussed by means of demo material. Finally, a conclusion will be made whether the requirements were achieved.

9.1 Implementation

9.1.1 Inserting the Trained CPM Model

From the academic experiments, the best scoring Convolutional Pose Machine model ($Model_{GAN+DR}$) was used as the pose estimation model in the practical prototype. First, the trained model is converted to a protocol buffer file (.pb). A protocol buffer file contains a complete TensorFlow program, including weights and computation. It does not require the original model building code to run, which makes it useful for sharing or deploying. The iOS ecosystem uses Core ML; Apple's machine learning framework for performing inference with integration of pre-trained machine learning models on the edge. Since deployment of the model will happen on an iOS device, another conversion of this .pb file to .MLModel is required. Just like a Protocol Buffer file, a MLModel file encapsulates a model's prediction methods, configuration, and model description.

9.1.2 The Pose Estimation Process

[redacted]

From the pose-information it can be deduced where the object is and how it is rotated in the real world. The estimated pose is now used to update the position and rotation of GUI elements such as the 3D virtual cues that belong to the activity that is currently being presented in the tutorial. For example, the user could be asked to press the ON/OFF button, and hence a 3D arrow object is spawned at the location of this ON/OFF button. Please refer to Figure 9.1 for a schematic overview of the pose estimation process in the iOS application.

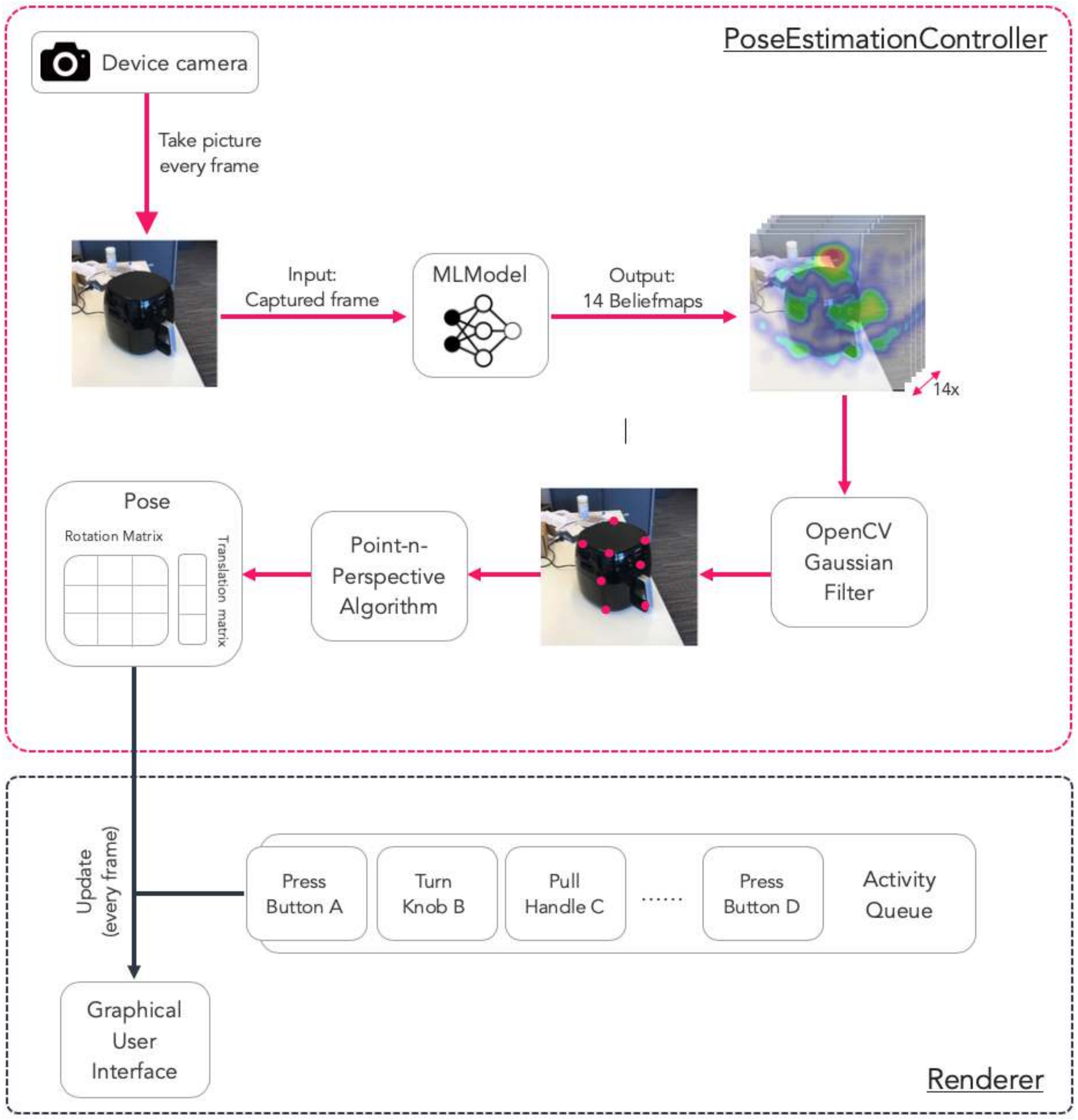


Figure 9.1: Proof of concept prototype - Schematic overview of the application's pose-estimation process. Every frame, the image buffer is read from the device's camera. From this image, the MLModel predicts beliefmaps which are converted to (x,y) coordinates after which a pose is estimated using the point-n-perspecive algorithm. Subsequently, the pose can be used to update the GUI elements.

9.2 Results

9.2.1 Demo Videos

Demo footage of the practical prototype can be found at the following links:

- **Pose Estimation Using Convolutional Pose Machines**
  The estimated pose is used to overlay a 3D CAD model over the physical object in real time on an iPad Pro 12.9' 2018. Figure 9.2 shows some snapshots taken from this application.
  `http://jeroenbrouns.com/thesis_prototype_demo_overlay.mp4`
- **Proof of concept prototype - Demo**
  Figure 9.3 shows screenshots of the application where the user is following a tutorial and is asked to turn on the airfryer by pressing the button as indicated by the 3D arrow.
  `http://jeroenbrouns.com/thesis_prototype_demo_concept.mp4`

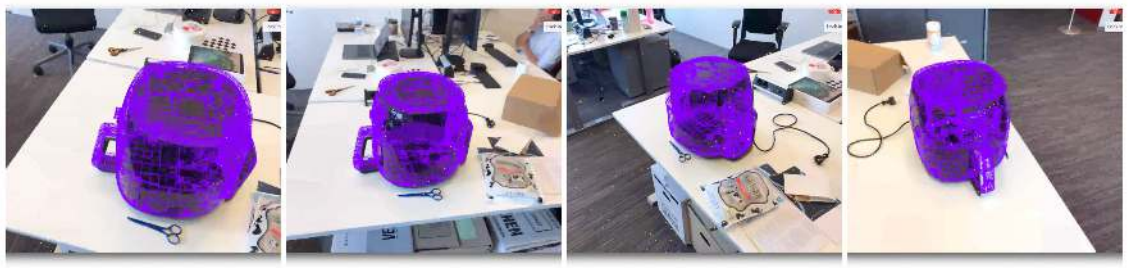
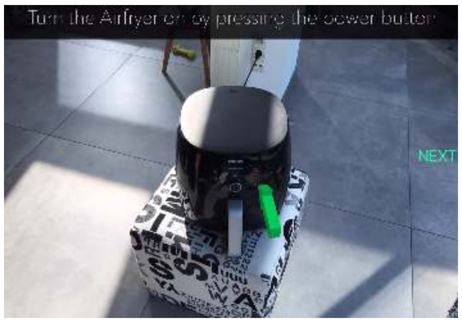
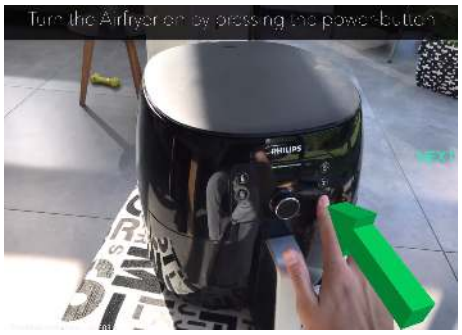


Figure 9.2: Practical Prototype - Augmenting the real airfryer object with an overlay of its 3D CAD model.



(a) User is asked to press the ON button



(b) User presses the ON button

Figure 9.3: Proof of concept prototype - The user is asked to turn on the airfryer by pressing the ON/OFF button (left). Despite the heavy distortion of the camera feed, because the user is changing its position, the augmented virtual cue (the 3D arrow) remains in the correct position at all times (right).

9.3 Discussion

A solution was developed that allows the real-time estimation of complex texture-poor objects such as the Philips air fryer HD9650. The current solution is not always accurate enough to overlay the physical object with a 3D CAD overlay, but is accurate enough to assign 3D virtual cues to certain components of the object. The most important question that needs to be answered is whether the pose-estimation model is sufficiently accurate for use in a production environment. The estimated pose needs to be correct at all times, in all camera angles and positions. Figure 9.4 shows 12 screenshots of the prototype application that were taken with a 400ms interval. The virtual 3D arrow that is used to indicate the location of the physical knob remains at a steady position. If the pose-estimation model fails to accurately estimate a pose, the last correct pose will be used. Apple's augmented reality toolkit takes care of all simultaneous localization and mapping that is required in between frames. The current prototype only supports 1 object, since the used CPM model does not solve a classification task. Multiple objects could be supported by introducing a 2-step approach: a YOLO-like algorithm that first classifies the object (step 1) after which a CPM model is loaded specifically for that object (step 2). Currently, the prototype is developed specifically for iOS, but Google's Android ecosystem provides the same functionalities for deploying deep learning models on the edge via their TensorflowLite framework.

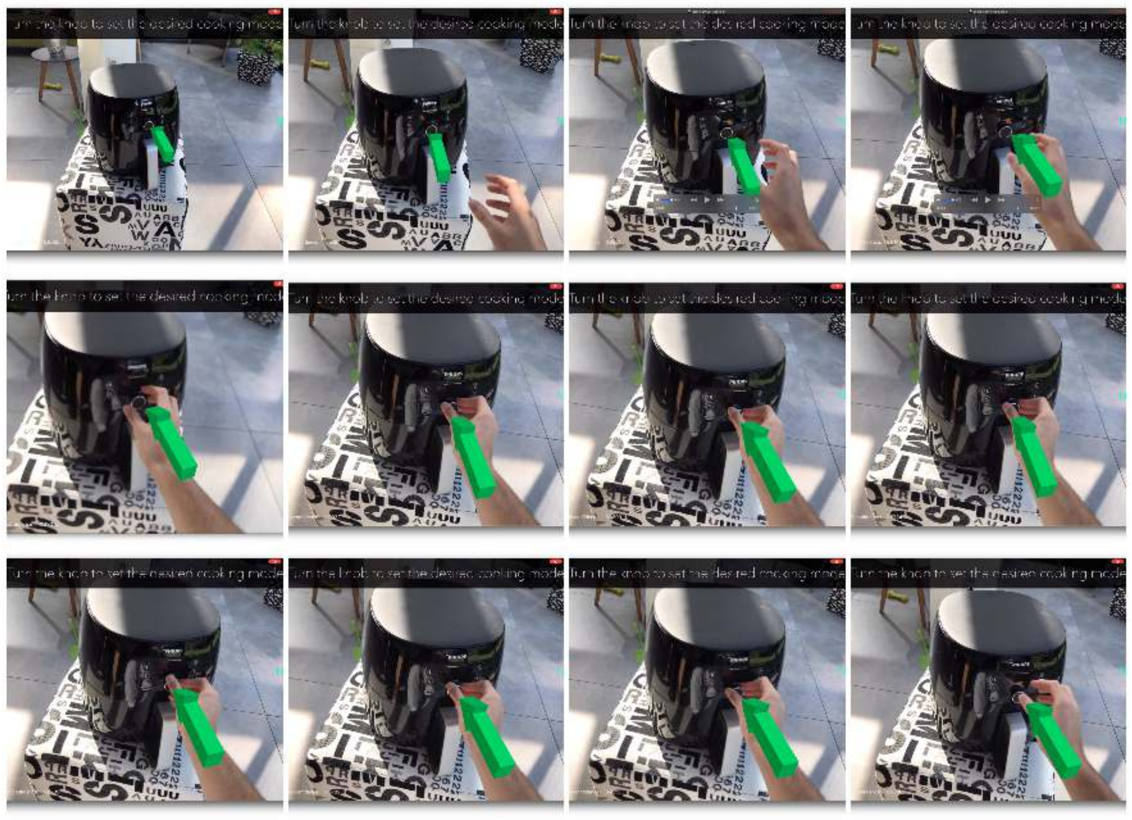


Figure 9.4: Proof of concept prototype - Turning the air-fryer's main knob example. At all points, the pose estimation remains accurate so that the virtual cues can be augmented in the correct position and rotation with respect to the object.

Chapter 10

Conclusions

10.1 The Academic Side

10.1.1 Conclusions

The research questions were designed to investigate various methods and techniques that can help to minimize the domain-gap between synthetic data and real data. Ideally, we train computer vision models with real data however collecting and annotating this data is often resource expensive or simply not possible. We showed that the generative adversarial neural network style-transfer concept can be leveraged to make synthetic data look more photo-realistic. This implies the existence of a distribution mapping from the synthetic domain to the real domain. Subsequently we showed that our GAN-enhanced data significantly outperforms the original synthetic data in multiple computer-vision tasks with different neural network architectures, which implies that a matching data distribution between different domains is benevolent for task performance. We explored the use of domain-randomized synthetic data, which yields reasonable performance, yet performs the worst in comparison with all other synthetic data types. However, when training a model with a combined data-set of ”standard photo-realistic synthetic data” with ”domain-randomized synthetic data”, a performance increase was observed which significantly surpasses the performance of models that were trained with only ”standard photo-realistic synthetic data” or only ”domain-randomized synthetic data”. When comparing this combined model with the model that was trained with GAN-enhanced data, a similar performance was observed. The best model turned out to be a combination of ”domain-randomized synthetic data” and ”GAN-enhanced synthetic data”.

We can conclude that GAN enhanced data can perform better than its regular synthetic data counterpart. However, considering that combining ”domain-randomized data” with ”standard synthetic data” can yield nearly the same results as ”GAN-enhanced data”, it might not be worth the resources that are involved with creating the GAN enhanced data. In order to use the Cylce-GAN to enhance the original synthetic data, a data-set of real images is required. This data-set of real images is unpaired and does not have to be manually annotated, however this still results in adding additional post-processing steps which bring resource overhead. Some computer-vision tasks do not require predictions to be highly accurate and consideration must be made while taking into account the requirements of the final product/solution.

10.1.2 Future Work

Our claims about generative adversarial neural network style-transfer for synthetic data to real data were tested on different computer-vision tasks and architectures, yet only one single object (the Philips air fryer HD9650) was used for testing. Ideally, the proposed method would be tested with multiple objects. Future work could test the proposed methods on multiple objects.

Moreover, the manually annotated test set of real images did not contain many samples (240 images), which could have led to statistical bias. Future work could test the method on a bigger test-dataset. Since CylceGAN uses unpaired data, learning a correct distribution mapping from synthetic images to real images is harder for more complex objects. Future work could investigate a segmentation method where different object's materials get segmented as individual images. For each such created 'material class', a separate CycleGAN could be trained, which hence could focus on learning a specialized distribution mapping for that specific object's material. In our experiments, domain randomized data was combined with other data and trained in the same stage. Future work could investigate whether pre-training with domain randomized data and subsequently fine-tuning with other data has different effects on the inference performance in the real domain.

10.2 The Practical Side

10.2.1 Conclusions

By means of a concept prototype, we showed the viability of the interactive augmented reality user manual. The solution is highly scalable, since its prediction models can be trained with only synthetic data that is generated from the object's computer-aided design model. The models trained with synthetic data perform well enough to be used in a production setting. Both Apple and Google already provide promising software development tools which allows for deploying neural architectures on the edge. Especially when replacing the resource-demanding traditional convolutional layers with optimized convolutional feature extraction modules such as MobileNet, fast inference speeds on mobile devices can be achieved which was not possible until recently. It is to be expected that in the near future, the current SIFT-based approaches that are applied in augmented reality applications will be boosted (or entirely replaced) with deep-learning approaches such as the one proposed in this thesis.

10.2.2 Future Work

In order to ensure that the solution could make predictions in real-time, tuned-down hyper-parameter settings were selected and used in the prototype. These settings could be increased and will most likely result in higher prediction accuracies. For example, the input image resolution to the CPM model was only 192x192px and could be increased to achieve better performance, at the cost of inference speed. The amount of keypoints in the prediction tasks was set to 14 and could also be increased, effectively providing the PnP-algorithm more robustness in its pose-predictions. The prediction model of the prototype regresses directly on key-points, yet future work could investigate other methods that use, for example, iterative refinement through a post-processing segmentation task [56]. Future work could also be the optimization of the convolutional extraction modules. Currently, MobileNetv2 is used, but other improved modules are already published (for example: MobileNetv3).

Bibliography

[1] Global augmented reality market size 2025 - statista. https://www.statista.com/statistics/897587/world-augmented-reality-market-value/. (Accessed on 12/11/2019).

[2] 5 trends emerge in the gartner hype cycle for emerging technologies - 2018 smarter with gartner. https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/. (Accessed on 12/11/2019).

[3] What's new in yolo v3?. towards data science. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b. (Accessed on 06/24/2019).

[4] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans, 2017.

[5] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention, 2018.

[6] John M. Carroll and Mary Beth Rosson. Interfacing thought: Cognitive aspects of human-computer interaction. chapter Paradox of the Active User, pages 80–111. MIT Press, Cambridge, MA, USA, 1987.

[7] 5 reasons why google glass was a miserable failure - business 2 community. https://www.business2community.com/tech-gadgets/5-reasons-google-glass-miserable-failure-01462398. (Accessed on 12/11/2019).

[8] Dai Clegg and Richard Barker. Case method fast-track - a rad approach. 1994.

[9] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, Sep. 1999.

[10] Yolo: Real-time object detection. https://pjreddie.com/darknet/yolo/. (Accessed on 06/24/2019).

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[13] Philips anticounterfeit program works to eliminate fraudulent xenon hid headlight bulbs. https://www.aftermarketnews.com/philips-anti-counterfeit-program-works-eliminate-fraudulent-xenon-hid-headlight-bulbs/. (Accessed on 12/11/2019).

[14] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects, 2018.

[15] Jason Rambach, Alain Pagani, Michael Schneider, Alexander Artemenko, and Didier Stricker. 6dof object tracking based on 3d scans for augmented reality remote live support. *Computers*, 7:6, 01 2018.

[16] Understanding arkit tracking and detection - wwdc 2018 - videos - apple developer. `https://developer.apple.com/videos/play/wwdc2018/610`. (Accessed on 07/15/2019).

[17] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: A hands-on survey. *IEEE Transactions on Visualization and Computer Graphics*, 22, 01 2016.

[18] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. Comput.*, 22(1):67–92, January 1973.

[19] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *Int. J. Comput. Vision*, 61(1):55–79, January 2005.

[20] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2878–2890, December 2013.

[21] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks, 2013.

[22] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient object localization using convolutional networks, 2014.

[23] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines, 2016.

[24] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *Lecture Notes in Computer Science*, page 483–499, 2016.

[25] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking, 2018.

[26] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation, 2019.

[27] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Int. J. Comput. Vision*, 66(3):231–259, March 2006.

[28] Vincent Lepetit and Pascal Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, 1(1):1–89, 2005.

[29] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[30] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2017.

[31] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects, 2018.

[32] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction, 2017.

[33] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation, 2018.

[34] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[36] Varun Ramakrishna, Daniel Munoz, Martial Hebert, J. Andrew Bagnell, and Yaser Sheikh. Pose machines: Articulated pose estimation via inference machines. In *ECCV*, 2014.

[37] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[39] The world's most valuable resource is no longer oil, but data - regulating the internet giants. `https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data`. (Accessed on 09/16/2019).

[40] Information rules : a strategic guide to the network economy : Shapiro, carl : Free download, borrow, and streaming : Internet archive. `https://archive.org/details/informationrules00shap`. (Accessed on 09/16/2019).

[41] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[42] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization, 2018.

[43] Farzan Erlik Nowruzi, Prince Kapoor, Dhanvin Kolhatkar, Fahed Al Hassanat, Robert Laganiere, and Julien Rebut. How much real data do we actually need: Analyzing object detection performance using synthetic and real data, 2019.

[44] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games, 2016.

[45] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[46] Praveen Krishnan and C. V. Jawahar. Generating synthetic data for text recognition, 2016.

[47] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition, 2014.

[48] Menghua Luo, Wang ke, Zhiping Cai, Anfeng Liu, Yangyang Li, and Chak Cheang. Using imbalanced triangle synthetic data for machine learning anomaly detection. *Computers, Materials Continua*, 58:15–26, 01 2019.

[49] Edgar Alonso Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. Paysim: a financial mobile money simulator for fraud detection. 2016.

[50] Adam Kortylewski, Andreas Schneider, Thomas Gerig, Bernhard Egger, Andreas Morel-Forster, and Thomas Vetter. Training deep face recognition systems with synthetic data, 2018.

[51] Weiwei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[52] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2017.

[53] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[54] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.

[55] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[56] Riza Alp Guler, Yuxiang Zhou, George Trigeorgis, Epameinondas Antonakos, Patrick Snape, Stefanos Zafeiriou, and Iasonas Kokkinos. Densereg: Fully convolutional dense shape regression in-the-wild, 2018.

[57] Us6711293b1 - method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image - google patents. `https://patents.google.com/patent/US6711293`. (Accessed on 09/23/2019).

[58] tzutalin/labelimg: Labelimg is a graphical image annotation tool and label object bounding boxes in images. `https://github.com/tzutalin/labelImg`. (Accessed on 12/08/2019).

[59] M. M. Sarcar. *Computer aided design and manufacturing.* ., 2008.

[60] Unity-Technologies. Materials, shaders textures. In *Unity - Manual.* 2019.

Appendix A

Exploration by Predecessor

[redacted]. Research was conducted into both existing solutions offered by third parties and custom solutions that had to be developed in-house. Some conclusions were drafted, but no proper final solution was created. The following sections elaborate the investigated approaches in further detail.

A.1 Existing Solutions

Third parties such as Apple, WikiTude and Vuforia offer Software Development Kits (SDK) that enable the detection of objects through the use of Scale Invariant Feature Transform (SIFT). This algorithm was first published by David Lowe in 1999 [9] and subsequently patented in Canada by the University of British Columbia [57]. Different open-source variants of the SIFT-algorithm exists such as BRISK, FAST and ORB. SIFT belongs to a group of algorithms within computer vision that attempts to spot and describe local features in images as key-points. These key-points are obtained from scale-space extrema of differences-of-Gaussians (DoG) within a difference-of-Gaussians pyramid. A full elaboration of SIFT is outside of the scope of this thesis.

Although the exact implementations of feature extraction and matching algorithms in SDKs such as Apple's ARKIT or Vuforia is being kept secret, a general methodology for object detection can be defined as follows:

1. SIFT key-points of objects are first extracted from a set of reference images and stored in a database (see Figure A.1).
2. At inference time, keypoint-features are extracted from the to be matched target-image.
3. The extracted keypoints from the target-image are matched against the database by comparing individual features from the new image to the features in the database. This matching can be done by taking the Euclidean distance of the feature vectors and applying a k-nearest-neighbours (kNN) algorithm. In order to speed op the kNN search, a best-bin-first (BBF) approximation for selecting point matches can be used.
4. An efficient hash table implementation of the Hough Transform is used to identify clusters of features that vote for the same object. The best solution is selected as a possible match.

SIFT is claimed to be scale, rotation, illumination and viewpoint invariant, but in practice results are poor. Non-rigid, opaque and reflective objects are the Achillesheel of SIFT features.

A solution was created using the Vuforia SDK and turned out to be of insufficient quality, due

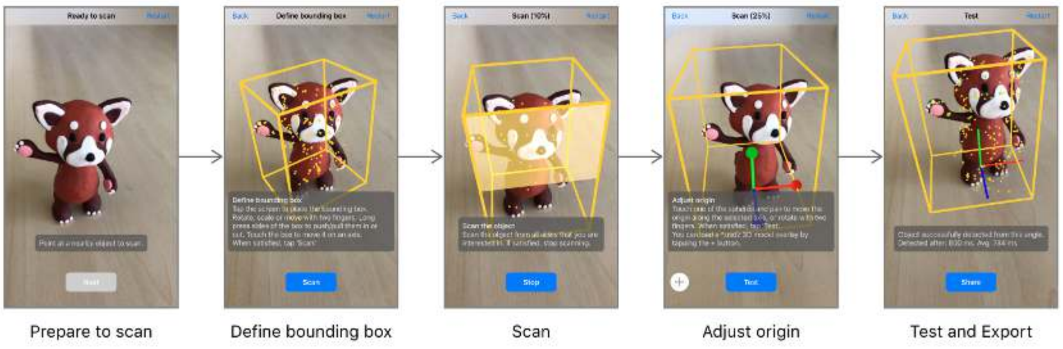


Figure A.1: Apple ARKIT - Creating a reference database by scanning features that can be used for matching during inference.

to the nature of the objects that had to be detected. Techniques that are based on SIFT require texture-rich surfaces that allow sufficient features to be extracted. Many Philips products lack high-quality features (detailed textures, colours, etc) and are translucent, incandescent or reflective in nature, which made a SIFT-based approach unsuitable (at least the solutions that were available at the time).

Lets take one of the flag-ship Philips Airfyrers, the HD9650, as an example (see Figure A.2). The HD9650, like most Philips Airfryers, mainly consists of a highly reflective black material (marked in red). Reflective properties do not allow for robust unique feature extraction, since reflections can be considered a form of random noise; reflections are not part of the actual object and hence cannot be used for feature matching. Only few other sub-components like the handle and the buttons (marked in green) can be used for feature extraction, yet these are wildly insufficient to obtain accurate results.

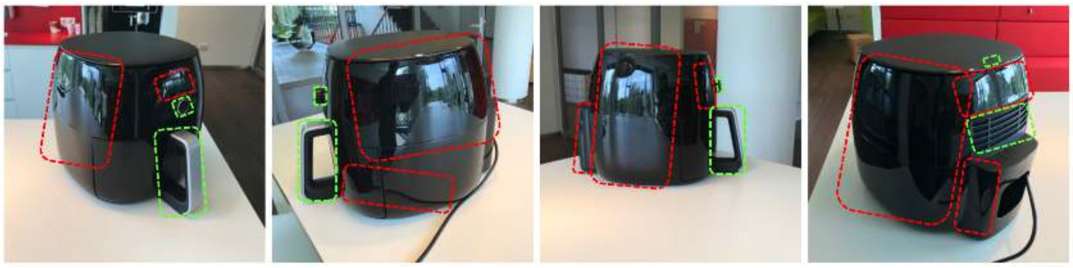


Figure A.2: Philips Airfryer HD9650 - Example of regions that do not allow for robust unique feature extraction (in red) and regions that are suited for robust unique feature extraction (in green)

A.2 Deep Learning

Since existing tools and solutions failed in the specified use-case, attention was shifted to detection approaches that include end-to-end Deep Convolutional Neural Network (CNN) architectures, which have proven to be very well suited for these types of classification tasks [12]. The initial idea was to, beside classification of objects in images, also perform localization (determining the position of the classified object in the image). Real data was annotated by hand: objects in training-images were assigned a label in the form of a ground-truth 2D bounding box to locate the specific object in the image. This was done for 3 specific Philips Consumer products: an air-fryer, an air-purifier and a toothbrush. Subsequently, a deep object detector model called "You Only

Look Once" (YOLO) was trained. YOLO was selected, because of its unparalleled inference speed in comparison with competing methods, while still maintaining high accuracy. Conclusions from this exploratory project were as follows:

- The trained models performed 'alright'. Localization of the objects was robust (there was almost always a prediction present), yet not always accurate locations for these objects were predicted.
- Inference was runs on an NVIDIA 1080ti graphics card at approximately 30fps. Considering that the solution has to run on mobile devices, the YOLO algorithm still is far too slow. However optimizations can be made here: reducing model parameters, using bottleneck features etc.
- It was very cost and resouce expensive to gather and manually annotate the neccesary trainingdata. This data-gathering process considered to not be scalable in the case of training models with many objects (more than 50, instead of just 3 Philips products). Perhaps synthetic data could be of use here; automatically annotated data could be generated on-demand based on Computer-Aided-Design (CAD) Models of their respective consumer product. This generated data can then be used for training a detector.
- Just performing classification and predicting 2D location of full objects in their entirety is not enough to adequately solve our use case. We are mainly interested in the sub-components of the full object. For example when we consider the airfryer, we would like to know where its buttons are located, its handle, airvent etc. The final product should be able to, for example, "point the user to press a specific button on the airfryer" or "open the airfryer by pulling this or that handle". The solution should be able to direct the user through all features and components of the object. Three possible solutions to this (nested) multi-class problem were identified:

    1. A possible solution is to **treat object's sub-components as individual classes.** Instead of parent-classes like "Airfryer model XXX" or "Toothbrush model XXX", we would get "Airfryer model XXX power button", "Airfryer model XXX handle", etc. This would occur in 2D, which is disadvantageous. No information about a 3D environment would be known and thus no sub-components can be detected that are occluded or placed on the back-side of the object. The solution preferable can guide a user to a subcomponent of the object that is not currently visible.
    2. Possible, **a hierarchical tree** to relate the classes and sub-classes could be used just like the one in YOLO9000. In YOLO9000, a general class is assigned (for example: dog), after which a specification is made according to the hierarchy tree (the type of dog is a Belgian Malinois). A variant of this mechanism could be used to first detect the Airfryer and subsequently its parts.
    3. **Leveraging the spacial and geometrical information of an object's pose** (the rotation and translation of an object with respect to its origin). Pose Estimation is often done through regressing on the locations of specific hand-picked feature-points instead of regressing to bounding-boxes. A post-processing algorithm known as Point-n-perspecive (PnP) is then used to estimate the pose of the object. If we estimate the pose of an object, we can also deduce the relative locations of its sub-components, since a 3D overlay can now be placed over the image.

       Option 1 was briefly explored, but not further elaborated in this thesis. Option 3 is the main method that was explored and developed in this thesis.

Option (1) was briefly explored, and yielded good results but is however not further discussed in this document. Option (3) is the main method that was explored and developed in this thesis.

Appendix B

Creating the Evaluation Data-set in the Real Domain

A dataset of real images of the Philips Airfryer HD9650 $Dataset_{REAL}$ was created and will be used to measure the inference performance of the different models in the real domain. Pictures were taken in different background settings on the Eindhoven High Tech Campus using a 2018 iPad Pro 12.9 and subsequently annotated by hand. Recall that this dataset will be used to evaluate two different computer-vision tasks with inherently different annotation formats and hence labeling had to be done twice (for each annotation format):

- **Keypoint-coordinate annotation format (CPM)**
  Two tools were developed to annotate real data according to our requirements. The first tool was an iOS application that made use of a briefly trained CPM model that provides a rough estimate of the pose of the object. Subsequently, the physical object could be overlayed with a 3D virtual object using the estimated pose-information. This rough pose-estimate could then be refined by hand using GUI-controls. The native Augmented Reality Simultaneous Localization and Mapping (SLAM) techniques were used to continuously map the environment so that the 3D overlay would remain correctly positioned when the user moves around in the real world. Because of this, pictures could be taken continuously while moving around the object and could be automatically annotated by comparing the user's movement data with the initial pose of the object. An ideal data-gathering technique, however, not accurate enough; the augmented reality tracking was insufficient to preserve an accurate location when moving around the object. It was decided to abandon this method since no additional factors (such as keypoints with a measurement offset) should be introduced into the experiment. Method number two involved the development of a C .NET application, which allowed for manually selecting a subset of the 14 keypoints that were not occluded in the images. This method was taken to create a fairly accurate dataset of 217 annotated real images. More information about both methods can be found in appendix (XXX).

- **2D bounding-box for the whole air fryer object (YOLO)**
  The MIT-licensed labelIMG [58], an open-source python application, was used for labeling images in the required keras-yolov3 format.

  - **LabelImg: a graphical image annotation tool**
    `https://github.com/tzutalin/labelImg`
    Authors: tzutalin

Appendix C

Philips Synthetica

C.1 What is Philips Synthetica?

Philips Synthetica is a custom synthetic data generation pipeline written for this thesis (see Figure C.1 for a screenshot). The tool is build with the Unity 3D Engine and written in C#. Philips Synthetica supports multiple annotation formats among which the COCO format, singleshot/linemod format and keras-yolov3 format. It supports domain-randomization of (grouped) textures, domain-randomization of lights, domain-randomization of backgrounds, mask-image generation and post-processing noise filters.

A video of Philips Synthetica generating data in action can be found at:

- **Philips Synthetica Demo - Airfryer**
  `http://jeroenbrouns.com/philips_synthetica_demo.mp4`

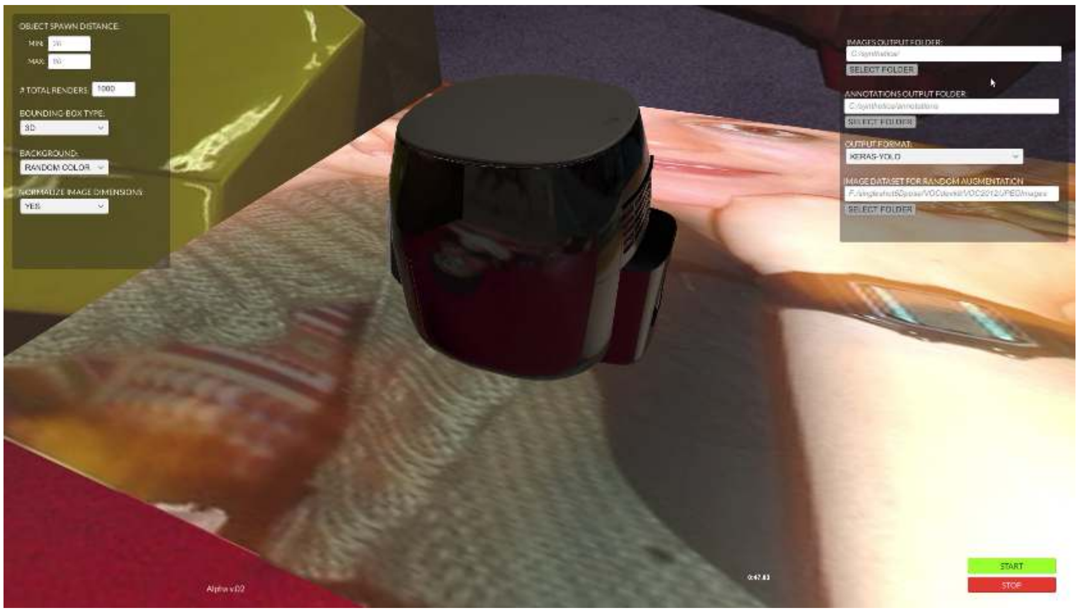

OBJECT SPAWN DISTANCE:
MIN: 30
MAX: 50
TOTAL RENDERS: 1000
BOUNDING-BOX TYPE: 3D
BACKGROUND: RANDOM COLOR
NORMALIZE IMAGE DIMENSIONS: YES
IMAGES OUTPUT FOLDER: C:/synthetica/
ANNOTATIONS OUTPUT FOLDER: C:/synthetica/annotations
OUTPUT FORMAT: KERAS-YOLO
IMAGE DATASET FOR RANDOM AUGMENTATION: F:/singleshot6Dpose/VOCdevkit/VOC2012/JPEGImages

Figure C.1: Philips Synthetica - Screenshot of an airfryer image being generated by the tool

C.1.1 Computer-Aided-Design

Computer-aided design (CAD) is the use of computers (or workstations) to aid in the creation, modification, analysis, or optimization of a design [59]. The user can input such a CAD design in the formats .obj, .dae, or .fbx, assign materials and shaders to the design and finally use the 3D model to create annotated renderings in high volume. A graphical user interface allows the user to change settings such as output-directory, amount of images that needs to be rendered, annotation format etc.

Background Information: Unity's Material Rendering

Rendering in Unity utilizes 3 components [60] which are closely linked to each other: materials, shaders and textures.

- **Materials:** definition of how a surface should be rendered. In other words, it contains the details of how to map a 2D texture onto a 3D model via tiling information, color tints and more; the available options for a Material depend on which Shader the Material is using. A material defines surface properties such as its basic color (Albedo), reflectively and roughness.

  A material utilizes the following sub-components:

  - **Shaders:** small scripts that contain the mathematical calculations and algorithms for calculating the color of each pixel rendered, based on the lighting input information and the Material configuration.
  - **Textures:** a bitmap image (mapping from some domain to bits. For example: a range of integers to bits). A Material often contains a reference to a texture, which the Material's shader can use to calculate for example surface color.

C.2 Modeling Realistic Reflections

The main use-case as described in subsection 1.2.1 involves creating pose-estimation and detection models of complex texture-poor reflective objects. Because of this, Philips Synthetica offers functionality to model realistic reflections in the object via ray-casting a surrounding picture-wall (see image Figure C.2) by means of a reflection probe. A Reflection Probe captures a spherical view of its surroundings in all directions (see Figure C.3). The captured image is then stored as a cubemap (see Figure C.4), which conceptually is a box with flat images of the view from six directions painted on its interior surfaces. The object where the reflections are supposed to be modeled use a shader that accesses the vector and color information of the corresponding side of the cube map.

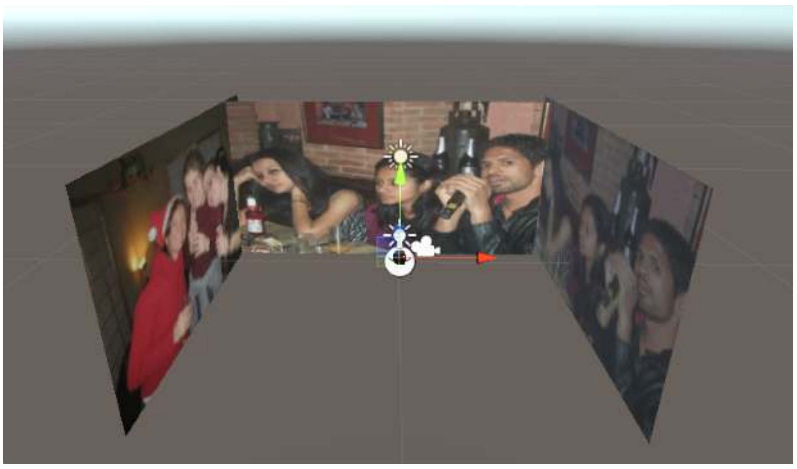


Figure C.2: Philips Synthetica - Wall that gets assigned random images from the VOC2012 datasets. The air fryer object can now reflect the images on this picture wall by means of a reflection probe. Note that in the actual tool, this picture wall has 6 surfaces, completely enclosing the object.

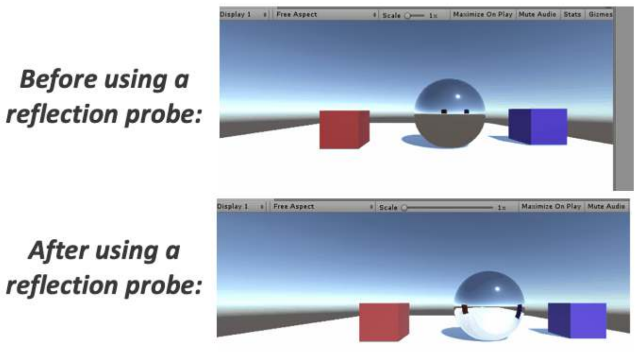

Before using a reflection probe:
After using a reflection probe:
Display 1
Free Aspect
Scale
Maximize On Play
Mute Audio
Stats
Gizmos

Figure C.3: Philips Synthetica - Before (top) and after (bottom) pictures when using a reflection probe. In the bottom picture it is clearly visible that the sphere reflects its surroundings.

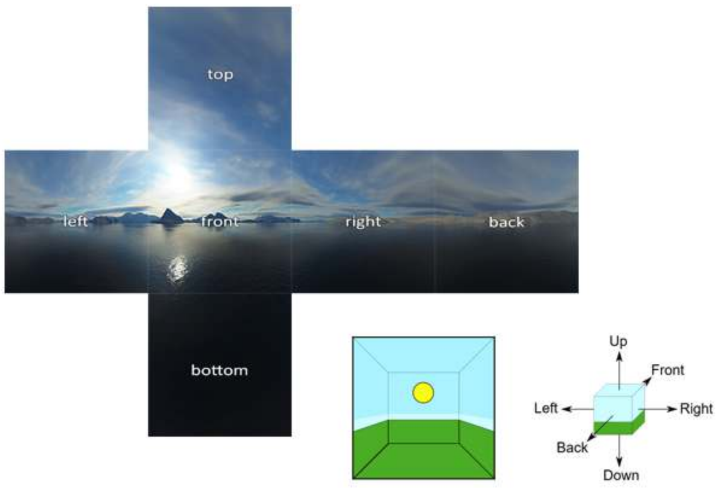


Figure C.4: Cubemap example - The view from six directions are painted on the interior surfaces of a cube.

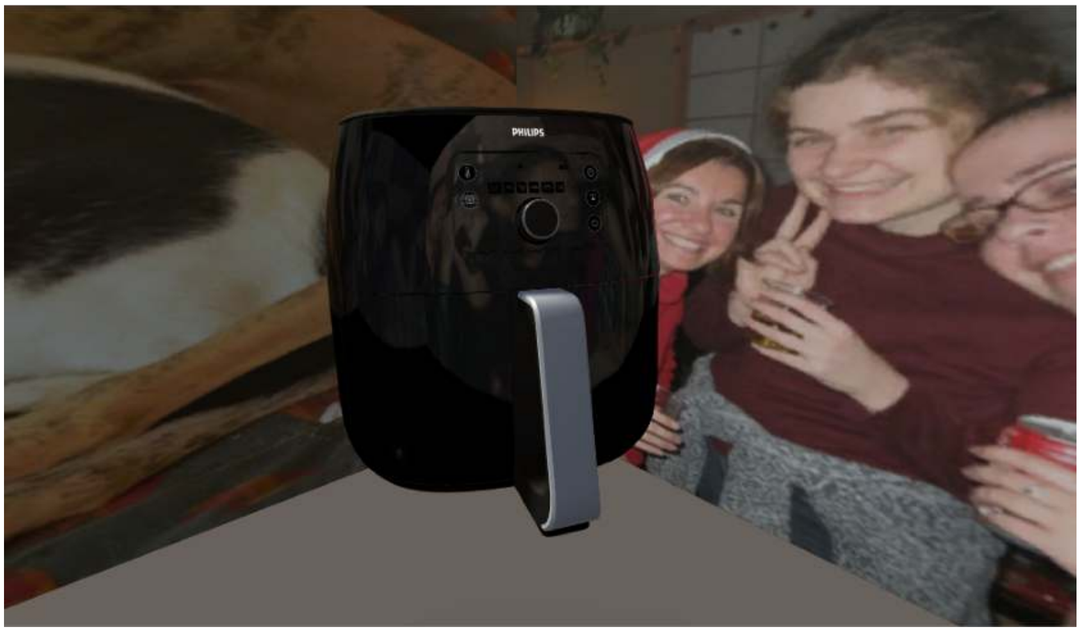


Figure C.5: Philips Synthetica - Screenshot of Philips Synthetica generating an airfryer image. Note the reflections of the picture-wall in the air fryer object.

C.3 Calculating the Annotations

C.3.1 Bounding-box Annotations

A bounding-box annotation is created for each component of the 3D model that was flagged to be a wanted annotated class. Child-meshes of each such component are merged and a new mesh is calculated. The 3-dimensional bound-coordinates of this new parent mesh are now projected via shooting rays onto a 2-dimensional plane that origins in the virtual camera. These obtained 2D coordinates are then normalized for the camera-dimensions and written to their respective label file.

C.3.2 Key-point Annotations

In the 3D model, tagged helper objects are placed that tell the data generator what coordinates to export as keypoints. This 3D keypoint is converted to a 2D coordinate via ray-casting using Unity's WorldToScreenPoint functionality and subsequently written into their respective label file.