Universität
Bremen

Faculty 3: Mathematics and Computer Science

I Cannot Believe All this Dirt!

Analysis of Synthetic Soil-Based Occlusion on Tableware Detectors with a Procedural Tableware Data Generator

Jacky Philipp Mach

machja@uni-bremen.de

Matriculation No. 4329647

1. Examiner:	Prof. Dr.-Ing. Udo Frese
2. Examiner:	Dr. Rene Weller
Advisor:	Prof. Dr.-Ing. Udo Frese

A Thesis Submitted for the Degree of
Master of Sciene in *Computer Science*

October 21, 2024

Jacky Philipp Mach

I Cannot Believe All this Dirt!

Analysis of Synthetic Soil-Based Occlusion on Tableware Detectors with a Procedural Tableware Data Generator

Master Thesis, Faculty 3: Mathematics and Computer Science

University of Bremen, October 2024

Declaration of Authorship

Hereby, I declare that I have composed the presented work independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

Bremen, the October 21, 2024

P. Mach

Jacky Philipp Mach

Acknowledgements

Throughout the writing of this thesis, I have received a great deal of support and assistance. I would first like to thank my supervisor, Prof. Dr. -Ing. Udo Frese, whose expertise was invaluable in each stage of the process, from the formulating of the research topic to the writing of this thesis. You supported me greatly and were always willing to help me.

I would also like to thank Dr. Rene Weller for taking the time to review my thesis.

In addition, I would like to thank my family and friends, who supported me by providing happy distractions to rest my mind outside my research.

Last but not least, I would like to thank Desiree Bergner for all the support she has given me during this difficult time.

Abstract

With the growth of the global market for robotics technology, the investment in the catering industry and the potential automation of physical tasks such as collecting and cleaning soiled tableware, object detector models are required that can handle dynamic environments, object conditions and sensory variations. For an object detector to handle such challenges, data with high variability is required, which is not available in context of detecting tableware. Thus, this thesis develops a synthetic image data generator, which creates images of a dining room with clean and soiled tableware and the corresponding ground truth images. Furthermore, the importance of soil-based occlusion will be analyzed by training an artificial neural network (ANN) on a dataset with synthetic images of only clean tableware and on a dataset with synthetic images of clean and soiled tableware. The results show that the performance of tableware detectors that are only trained on images with clean tableware, deteriorates when soiled tableware must be detected. This result shows that disregarding soiled-based occlusion on tableware can deteriorate the real-world applicability of tableware detectors. For the future, the model should be compared to other existing tableware detector models. In addition, real images should be included in the dataset to analyze the domain gap between synthetic and real images in the domain of tableware detection.

Contents

1 Introduction **1**

2 Related Works **5**

2.1 Soil Detection Solutions 5

2.2 Synthetic Data Generators 8

2.3 Tableware Detection and Robotic Grasping Systems 10

3 Synthetic Dataset Generation **13**

3.1 Blender 14

3.1.1 Modifiers 15

3.1.2 Geometry Nodes 16

3.1.3 Materials and Shader Nodes 19

3.1.4 Compositor 23

3.1.5 Blender Python API 23

3.2 Dining Room 24

3.2.1 Plate 25

3.2.2 Fork 27

3.2.3 Knife 31

3.2.4 Spoon 33

3.2.5 Glass 36

3.2.6 Table 37

3.2.7 Chair 40

3.2.8 Room 44

3.2.9 Table Distribution 48

3.2.10 Lighting 49

3.2.11 Camera 51
3.3 Dining Room Materials 53
3.3.1 Tableware Materials 53
3.3.2 Furniture and Floor Materials 54
3.3.3 Wall Materials 61
3.4 Food Soil 63
3.4.1 Soil Definition 64
3.4.2 Implementation 65
3.5 Dataset Generation Pipeline 66
3.5.1 Blender Settings 66
3.5.2 Data Generation Process 67
3.5.3 Computational Cost of the Dataset Creation 68

4 Tableware Datasets 69

5 Convolutional Neural Network 73
5.1 Architecture 73
5.2 Training Setup 77
5.3 Training Results 78
5.3.1 Loss 79
5.3.2 Metrics 79

6 Evaluation 83
6.1 Quantitative Analysis 83
6.2 Qualitative Analysis 84

7 Discussion 91

8 Conclusion 95

Bibliography 97

List of Figures

1.1 Structure of the Thesis 4

3.1 Example of a Modifier Stack 16
3.2 Example of a Geometry Node 17
3.3 Example of a Node Graph 18
3.4 Applications of the Principled BSDF-Shader 21
3.5 Examples of Texture Maps 22
3.6 Plate Attributes 25
3.7 Generating Process of the Plate Object 27
3.8 Fork Attributes 29
3.9 Generating Process of the Fork Object 31
3.10 Knife Attributes 32
3.11 Generating Process of the Knife Object 33
3.12 Spoon Attributes 34
3.13 Generating Process of the Spoon Object 35
3.14 Glass Attributes 36
3.15 Generating Process of the Glass Object 37
3.16 Table Attributes 38
3.17 Generating Process of the Table Object 40
3.18 Chair Attributes 42
3.19 Generating Process of the Chair Object 44
3.20 Room Attributes 46
3.21 Generating Process of the Room Object 47

3.22 Generating Process of the Window Object 48
3.23 Tableware and Chair Distribution 49
3.24 Procedural HDRI Node Graph 50
3.25 Indoor Lighting Distribution 51
3.26 Example of Camera Placement 52
3.27 The Ceramic, Glass and Stainless Steel Material 54
3.28 The Wood and Wood Plank Material 55
3.29 The Marble Material 56
3.30 The Veined Marble and Marble Tile Material 57
3.31 The Plastic Material 58
3.32 The Metal Material 58
3.33 The Stone Tile Material 59
3.34 The Checkered Tile Material 60
3.35 The Carpet Material 61
3.36 The Concrete and Plaster Material 62
3.37 The Brick Material 63
3.38 Examples of Food Soil 64
3.39 The Soiled Tableware Material 66

4.1 Examples of the Generated Tableware Images 70
4.2 PCA of Cleaned and Soiled Dataset 71
4.3 Heatmaps of the Ground Truth Images 71

5.1 The CNN Architecture 75
5.2 Maximum Model Parameters Calculation 77
5.3 Loss Curves of the Model 79
5.4 Binary IoU Curves of the Model 80
5.5 Precision Curves of the Model 81
5.6 Recall Curves of the Model 82

6.1 Clean Model: Detection of Clean Plates 85
6.2 Clean Model: Misclassification of Reflections 86
6.3 Clean Model: Misclassification of Large Plates 86

6.4 Clean Model: Undetected Plates 87
6.5 Soiled Model: Detection of Clean Plates 88
6.6 Soiled Model: Identification of Glass Occlusion 88
6.7 Clean Model: Soil classified as Plate 89

List of Tables

6.1 Loss and Metric of Model Evaluation 84

1. Introduction

With the ongoing development of robotics [52] and the growth of the global market for robotics technology [40], more and more robots can be found in a wide range of industries [2, 15, 51, 55, 66]. Although the catering industry is a significant sector [27], where companies invest money, the use of robotics is not common due to multiple problems such as human-robot collaboration, object conditions, architectural obstacles, sensory variations, dynamic variations, and real-time constraints [31]. Still, Chui et al. [17] state “73 percent of the activities workers perform in food service and accommodations have the potential for automation, based on technical considerations”. In their analysis, they included physical activities such as preparing, cooking or serving food; cleaning food preparation areas; preparing hot and cold beverages; and collecting soiled tableware. Each physical activity can be further divided into different challenges, e.g. collecting soiled tableware can be categorized into detecting, grasping, and sorting tableware.

For object detection, machine learning or deep learning algorithms achieved meaningful results, but require data with high variability [35, 81, 86, 87]. Detecting tableware contains challenges such as object conditions, sensory and dynamic variations [23, 30, 31, 46, 47], which means that high variability in data is also required for this task [12, 30, 31, 36, 46, 56].

Collecting and labeling such a domain-specific dataset is costly, labor-intensive, time-consuming, has copyright implications in the case of external collections, and has ethical and legal concerns in terms of privacy. It is also error-prone due to inaccuracies during manual labeling and inconsistencies in case multiple people are labeling the data. Synthetic Data Generation is a solution for the mentioned problems, because it is automated, synthetic, generates error-free ground truths, and can generate the required variability in data such as object variations, sensory variations, and dynamic environments. With this approach, an unlimited number of realistic synthetic data can be created and used to train deep learning algorithms. Multiple works have been published using synthetic data for model training and state improved model performances [16, 37, 38, 63, 65, 83].

In the context of the catering industry and the use of robotics technology, multiple works (see chapter 2) have been published on soil detection solutions, tableware detection, and robotic grasping systems, but are limited to a set of tableware or environments, and did not consider soiled dishes. These limitations affect the applicability of these systems in real environments. In addition, different works [11, 14, 23, 46] state that soil can significantly affect model performance, for example, due to soiled-based occlusion.

To address this research gap, the thesis provides an analysis on the effects of soil-based occlusion on the performance of a convolutional neural network (CNN) by asking the following question: (1) "Does the recognition of dishes in a CNN improve when images of soiled tableware are included in the model training process?".

The structure of the thesis can be seen in figure 1.1. In chapter 2, related works on soil detection solutions, tableware detection and robotic grasping systems, will be outlined. Afterward, the implementation and functionality of the synthetic tableware dataset generator, which is used to address the research gap, will be presented in chapter 3. The dataset generator is capable of creating synthetic images of soiled tableware in dining rooms and allows users to customize characteristics such as the set of tableware, types on soil on the tableware, dining room layout, camera perspective, lighting, and the level of soil. With the dataset generator, two synthetic image datasets will be created for a pixel-wise binary classification task. One dataset contains images of dining rooms with soiled tableware, and the other dataset contains images of dining rooms with clean tableware. The ground truth images will be binary masks of clean and soiled plates. Attributes, such as geometry or material, of the dining room and tableware will be randomized for each image. Chapter 4 will present an analysis of both datasets, such as the distributions of the randomized attributes of the tableware or the dining room, to provide insight into the generated datasets. In addition, heatmaps of the percentage of occurrence of labeled pixels in the ground truth images will be presented. To answer the research questions, both datasets will be used to train a CNN, which should detect plates in images and output the pixels that belong to a plate. The chosen model architecture and training process will be explained in chapter 5. The performance of the CNN after being trained on both datasets will be analyzed, compared and presented in chapter 6. Based on this research, the results and current limitations of the thesis will be discussed and possible future research areas will be highlighted in chapter 7 and chapter 8.

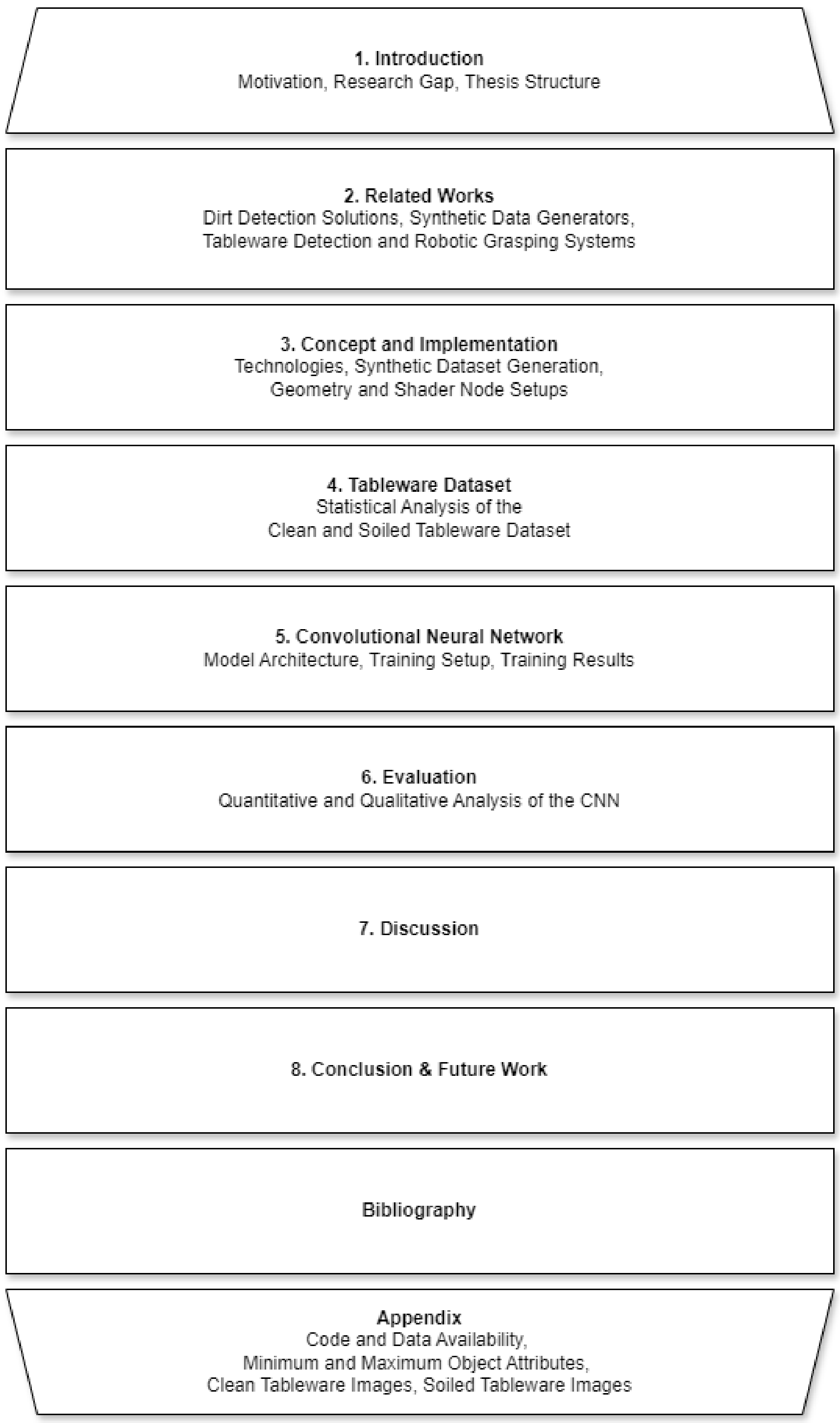

1. Introduction
Motivation, Research Gap, Thesis Structure
2. Related Works
Dirt Detection Solutions, Synthetic Data Generators,
Tableware Detection and Robotic Grasping Systems
3. Concept and Implementation
Technologies, Synthetic Dataset Generation,
Geometry and Shader Node Setups
4. Tableware Dataset
Statistical Analysis of the
Clean and Soiled Tableware Dataset
5. Convolutional Neural Network
Model Architecture, Training Setup, Training Results
6. Evaluation
Quantitative and Qualitative Analysis of the CNN
7. Discussion
8. Conclusion & Future Work
Bibliography
Appendix
Code and Data Availability,
Minimum and Maximum Object Attributes,
Clean Tableware Images, Soiled Tableware Images

Figure 1.1 Structure of the Thesis

2. Related Works

In the following chapter, related works will be presented to provide an overview of the different research areas that this thesis belongs to, such as different soil detection solutions, synthetic data generators, tableware detection and robotic grasping systems. Although research has been conducted in these fields, the related works have stated different research gaps, which this chapter will highlight and how the thesis will address them.

2.1 Soil Detection Solutions

There are different approaches to soil detection that are applied to different domains such as tableware, floors or cars. This task can be interpreted as pixel-wise detection [5] to detect soil as occlusion, object detection [9], object classifier [14] to categorize soil (for example, solid and liquid soil) or segmentation task [33] to determine the position and shape of soil. The following related works present different soil detection solutions that utilize artificial neural networks (ANN). Although these works focus on soil detection for floors, while the thesis focuses on soiled object detection, the related works offer insight into different approaches to gather, create and utilize image data, which contain soil, for training ANNs. In addition, there are research gaps and possible difficulties in using image data that contain soil.

Kim et al. [46] analyzed different control methodologies for robots used for cleaning tasks, such as sweeping floors, washing dishes and wiping windows. The authors differentiated between classic control approaches consisting of position control, force and impedance control, and learning-based controlled methods, such as learning from demonstration, supervised learning, and reinforcement learning. The authors concluded that current cleaning robots dedicated to specific applications cannot handle the variability of domestic environments. Additionally, they state that the performance of supervised learning algorithms is dependent on given data, thus the performance deteriorates in new environments compared to trained environments. This means that training data for supervised learning algorithms are required to have variability. This problem will be addressed by the thesis by developing a synthetic image generator of a dining room scene with soiled tableware. Scene characteristics such as lighting, set of tableware, soil or dining room layout will be adjustable to increase variability.

Bormann et al. [9] developed a visual soil and office item detection system, which is based on the YOLOv3 framework, for cleaning robots for tasks such as optimizing wet cleaning results and facilitating demand-oriented daily vacuum cleaning. For training data, a synthetic image dataset generator from a small set of real scene, dirt and object examples was developed. The dataset generator blends randomly modified patches of dirt and office objects with clean ground floor images, and optionally adds simulated light sources and shadows to the synthetic scene. Random numbers of segmented dirt and object samples are selected and randomly rotated, flipped and resized before being blended into the clean ground images. For post-processing, artificial shadows and brightness of random shape and intensity at random positions were added to mimic real scene variance. The authors concluded that collecting an larger dataset, compared to the dataset used in their work, might benefit detection performance and robustness as the set of real scene, dirt and object examples limits the variability of the synthetic images as the set is used for blending. The presented problem is addressed by this thesis by replicating a 3D dining room scene, where scene characteristics like room layout, lighting or set of tableware can be changed and randomized, to create synthetic image data with variability.

Canedo et al. [14] propose a vision system, based on the YOLOv5 framework, to detect soiled spots on the floor. For the training of their ANN, they extended a synthetic data generator found in literature to solve the lack of real data in this domain. They collected images of solid and liquid soil and clean floors online through Google Search. Floor images are selected by segmenting the floor from other objects in the image such that soiled spots are only generated on the floor and do not overlap with other objects. This approach helps models distinguish between soiled spots and objects in the image, which reduces the number of false positives. The tool blends dirt samples into clean floors in random locations and adds simulated light sources and shadows. It can also apply geometric transformations to floors and dirt samples, such as flipping the clean floor images horizontally and vertically. For testing, the authors conducted four experiments. In one experiment, their ANN was trained on floor images, where objects were removed from the training dataset to observe the impact of using floor images with objects on the models' performance. The results showed that the models falsely classified objects as soiled spots. For evaluation, their models were tested on a real dataset (ACIN dataset) and achieved positive results. Canedo et al. concluded that increasing the variety of floors, solid and liquid dirt in the synthetic image dataset might improve their reported results. In addition, their work was limited in liquid dirt samples, which affected the performance of their models. They also stated to include floor image samples with partially visible objects, because their models misclassified objects that appear partially in the image, such as chairs, wires or shoes, as soiled spots. This work shows that the use of synthetic image data for training ANNs can achieve positive results on real data and that the models are able to generalize on real data. Furthermore, the experiment showed that additional objects in the scene can reduce the misclassification of objects as target objects, in this case soil. To address the mentioned problems, additional objects such as chairs, glasses, spoons and forks will be added to the dining scene in the synthetic image data generator of this thesis.

2.2 Synthetic Data Generators

To train an ANN, task-specific labeled data is required, which in some cases are not available due to the task, the domain or difficulties in creating such dataset. Manually collecting and labeling training data can be costly, labor-intensive, time-consuming, has copyright implications in the case of external collections, and has ethical and legal concerns in terms of privacy. Additionally, manual labeling is error-prone due to inaccuracies and inconsistencies in the case of multiple people labeling the data. Synthetic data generators are an automated alternative that generates ground truths without errors and can generate labeled synthetic data with the required variability that is needed to train ANNs to achieve meaningful results [35, 81, 86, 87]. For creating synthetic image data, the generators render a replicated 3D scene of the domain environment and the corresponding ground truth label, or use a set of real images to combine and create new synthetic images [9, 14]. The presented works achieved positive results in different domains with the use of synthetic image data generators to generate datasets that are used to train ANNs. They give insight into the usage, problems, results and adaptability of synthetic data generators.

With the aim of automating containerized freight transport processes, Delgado et. al [19] developed ANNs to detect and segment categorized damages, such as concave damages on the container face, damages located on the edges of the container or dents. In addition, the detection of markers that identify the type of transported goods and a door/no-door classifier. Due to the lack of real image data in such a domain, a methodology was developed to generate a synthetic dataset, which is realistic, varied, balanced and labeled, for visual inspection tasks of containers in a deck environment, which was modeled in the 3D computer graphics software Blender. The scene contains the container, an ship-to-shore crane with the clamp that hooks the container as a model. The purpose of the crane is to provide fidelity to the rendered scene. Camera locations and scene specifications are based on the existing setup of a port in the real world. The scene mimics position, focal length and image resolution with similar distortion and properties to the real cameras on the dock. For the lighting, high dynamic range images (HDRIs) are used as sole light source of the scene and background, but are randomized for each image. The set of HDRIs covers the variety of daytime and night-time images and urban or industrial environments with different weather conditions. The container object was modeled after the standard international organization for standardization (ISO) measures of containers. For realistic materials, defects such as dirt, aging, rust or scratches were added to the base material. The data generation pipeline is executed by a Python script, which interacts with the Blender scene objects natively. The color of the container material changes randomly given a list of RGB values. In addition, the container material attributes are randomly changed to age the container. For the randomization, the range of values were set empirically. A statistical analysis of the generated dataset were conducted to prove that the dataset is balanced and has variability due to the randomness of the object attributes. To evaluate their ANN, the authors conducted several experiments, which show that ANNs, trained on synthetic datasets, are applicable for real-world scenarios. They concluded that synthetic labeled datasets are a feasible alternative to train state-of-the-art ANNs that are applied for real-world problems. In addition, they stated that their methodology of generating synthetic image data allows researchers to generate any setups regarding of their configurations and also allows to generate any situation of interest. This thesis will adopt this methodology by implementing a dining scene, model the objects and scene after real-world measurements, randomize aspects of the objects and materials, implement a Python script to execute the data generation pipeline and conduct a statistical analysis of the two generated datasets. As stated in the work, objects of the target scene can be modeled to provide fidelity in the remodeled scene. This thesis will also adopt this approach and include objects such as fork, knife, spoon, glasses, table and chairs to the dining room scene.

To automate the risk assessment of occurring nanoparticles in various environments as a consequence of man-made processes, which raises concerns about their impact on the environment and human health, Mill et al. [56] developed an ANN that segments nanoparticles in imaging techniques such as microscopy and spectroscopy. For training the ANN, a synthetic image data generator was developed due to the lack of annotated training data in such a domain. The synthetic scene was created and rendered with the software Blender. To model nanoparticles, statistical information on nanoparticle morphology was extracted with respect to size, shape, and distribution from high-resolution microscopy images. In addition, real images of nanoparticles were used as references to reproduce features for the nanoparticle models. Potential image artifacts, such as dirt, that may arise from sample preparation procedures, were included for realism. The image data generator computes a photorealistic synthetic microscopic image of nanoparticles and potential artifacts, and a subsequent render produces the respective error-free ground truth label image. The authors concluded that the segmentation accuracy of their trained ANN is comparable to the segmentations performed by microscopy experts. In addition, they state that the synthetic data generation pipeline is able to produce an unlimited number of realistic synthetic images with error-free ground truth labels that can be used to effectively train a CNN, and is an approach to solve the lack of image training data and the problems of the manual annotation process of ground truth labels, such as time, cost and errors. In this thesis, the rendering process of the presented work is adopted by first rendering a photorealistic synthetic dining room scene and then rendering the corresponding error-free ground truth label image.

2.3 Tableware Detection and Robotic Grasping Systems

In the following section, two related works in tableware detection and robotic grasping systems will be outlined to provide insight into different tableware detection systems that utilize ANNs, their applicability in the domain of the food service industry and their research gaps.

With the aim of automating processes in the food service industry, Fukuzawa et al. [30] developed a robotic system consisting of a six-degree-of-freedom (DOF) robotic manipulator, a robotic hand capable of grasping and suction operations, and a three-dimensional (3D) camera for tableware recognition. The system is able to grasp different tableware from a commercial dishwasher. For tableware classification and orientation detection, an ANN named Faster R-CNN was used. The dataset for model training was manually collected and annotated. Six types of tableware were used for data collection. The trained ANN returned a bounding box, which is used to calculate the grasping points and postures. The authors concluded that strong reflection and overlap of the tableware significantly affected the accuracy of the tableware recognition. In addition, they reported that recognition accuracy could be improved by increasing the number of training data. In future work, they aim to extend the applications of their ANN by including more types of tableware. With the work of the thesis, the mentioned problems could be solved. The synthetic data generator is capable of generating an unlimited number of images and types of tableware to include in a new training dataset. It also allows to customize the rendered scene such as lighting or overlapping tableware. Furthermore, the work does not mention any tests with soiled tableware for their recognition system.

Yue et al. [89] developed a lightweight object detection algorithm based on deep learning for robots to detect empty tableware and calculate the grasping point for the various types of dishes in images, so that the robot's arm can pick them up. The dataset, used for training their model, were manually collected and annotated. Twenty different objects such as coffee, chopsticks, cup, bowl and spoons were included in their dataset. Like the works presented before, the authors developed an object detection algorithm to detect empty tableware, but did not test their algorithm with soiled dishes, which is more based on a real-world application.

3. Synthetic Dataset Generation

To answer the research question (1) “Does the recognition of dishes in a CNN improve when images of soiled tableware are included in the model training process?”, a labeled image dataset is required to train the CNN. A dataset to compare the effects of soiled-based occlusion on tableware detection does not exist or is not publicly available.

A dataset named “Cleaned vs Dirty V2 Dataset” [18], which is used for training classification models that classify images as dirty or clean, was found online on kaggle.com. The dataset contains 764 real images of clean plates and 292 real images of soiled plates. The dataset cannot be used for this thesis, because of the low sample size, the required ground truths, the dataset does not contain images with multiple plates and the plates were photographed from the same top view perspective, which limits the variability in the data.

As manual collecting and labeling image data can be labor-intensive, time-consuming or error-prone, a synthetic image data generator was developed that creates images of randomized dining rooms with clean or soiled plates, and the corresponding error-free ground truth label.

The generator will be used to create one image dataset that contains clean tableware and another image dataset that contains clean and soiled tableware. The dining room scene was modeled and rendered in the software Blender. The models and materials of the scene are procedurally generated, which means that the objects are created algorithmically in subsequent steps, which allows users to adjust the geometry of the object, such as length or width, or the color of the material without manual modeling. In addition, procedural generation allows for the randomization of geometry and materials by changing attribute values that are utilized in the object creation algorithm. Randomization of object characteristics allows for variability in the dataset.

To execute the data generation pipeline, a Python script was implemented that interacts with the Blender scene objects through the Blender Python application programming interface (API).

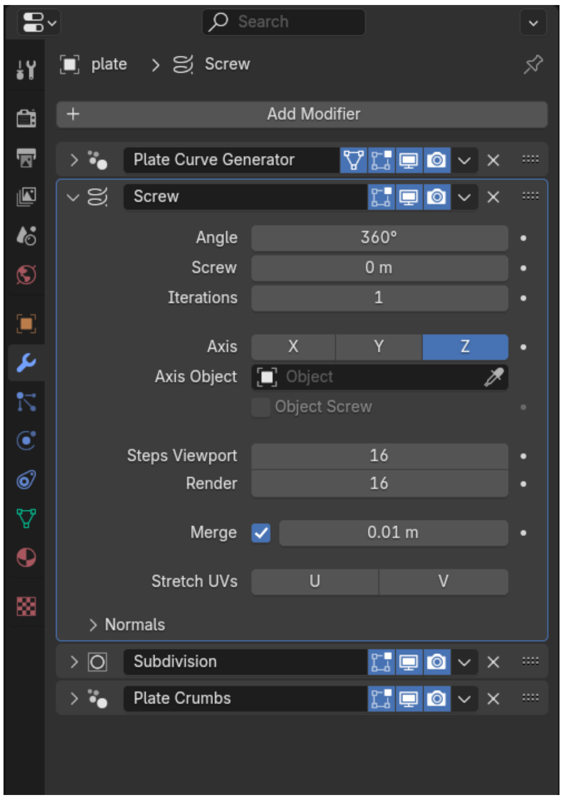
In the following chapter, the concepts and features for developing procedurally generated objects and materials, the implementation of the dining room objects, the used reference and implementation of soil, and the dataset generation pipeline will be presented in detail.

3.1 Blender

Blender is a free and open-source 3D computer graphics software, which runs on Windows, MacOS and Linux. It was developed in C and C++, but offers an API to interact with Blender scenes through Python scripts. The software offers features like object modeling, animation, sculpting, texturing, rendering, video editing, python scripting, compositing and physics simulation such as smoke, fluid or particles. Due to the variety of features and accessibility, the dining room scene was modeled and rendered in Blender 4.2.1. Features named *Modifiers* and *Geometry Nodes* were utilized to model and manipulate the geometry in a procedurally and non-destructive approach, which adjust the appearance of the object without changing the geometry data of the object. In addition, it allows users to adjust or randomize object attributes such as height, width or length without manual modeling. For texturing, *Shader Nodes* were used to generate procedural object materials that can be adjusted by attribute values.

3.1.1 Modifiers

Modifiers are built-in Blender operations that affect the appearance of the object without changing and finalizing the object's geometry data. Multiple modifiers, which form the "modifier stack", can be applied to an object. An example of the modifier stack can be seen in figure 3.1, of the plate object in the dining room scene. The input of a modifier is the object's appearance after the effects of the previous modifier have been applied, which means the order of modifier stack affect the final appearance of the object. Blender organizes the available modifiers after their functionality in four categories: *Edit*, *Generate*, *Deform* and *Simulate*. *Edit*-modifiers adjust object properties that do not directly affect the geometry of the object, such as the UV map of the object. *Generate*-modifiers add or remove geometry of the object. *Deform*-modifiers change the geometry of the object without removing or adding geometry. *Simulate*-modifiers are the implemented physics simulations such as smoke, fluid or particle simulation. To generate the dining room objects, only *Generate*-modifiers were utilized to add new geometry to the objects.



Angle 360°
Screw 0 m
Iterations 1
Axis X Y [✓ Z]
Axis Object Object
Object Screw
Steps Viewport 16
Render 16
Merge ✓ 0.01 m
Stretch UVs U V

Figure 3.1 Example of a Modifier Stack

3.1.2 Geometry Nodes

Geometry Nodes is an uncategorized modifier due to the different application possibilities such as object distribution, object modeling or geometry editing. The modifier executes functions based on a node graph, which is an approach of visual programming languages that abstracts source code as a graph, where nodes, which consist of function options, input and output sockets, are source code functions.

The node input and output sockets can have different data types, such as float, vector, color or geometry, that are differentiated by the color of the socket. The shape of the sockets defines whether the input or output is a single value, a field or a temporary single value that can also be a field. A circular socket shape visualizes a single value and cannot accept a field input. A diamond shape visualizes an input or output field. A diamond shape with a dot in the center visualizes a single current value, which can also be used as a field. An example of a geometry node can be seen in figure 3.2.

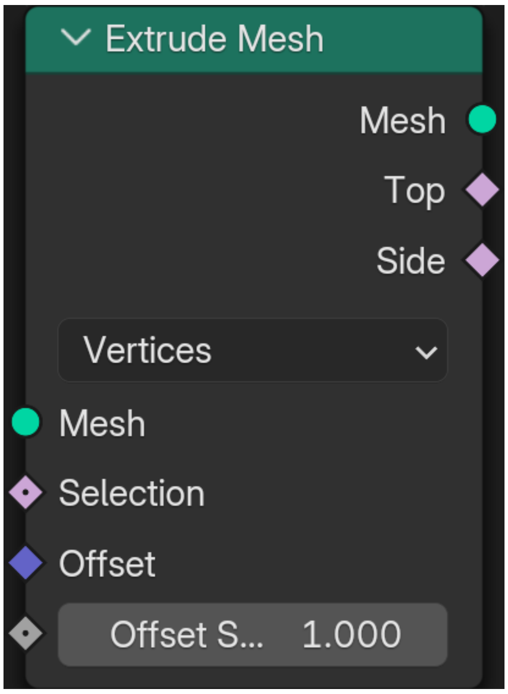


Figure 3.2 Example of a Geometry Node

A single value can be, for example, an integer or float value. A field, in the terms of programming, can be compared to a list and, for example, can contain numeric data types, such as vectors or floats, but also geometric data types such as vertices, edges or faces. Nodes with input field sockets perform the operation for each element in the input field, unless only a single value is connected. In this case, the output is identical for each element, because the single value will be used for the calculation for each element.

Furthermore, nodes are connected through edges to pass data from one node to another. The edges also visualize the graph's reading orientation. Node graphs can have *Group Input*-nodes, which allows the customization of the modifier result with a list of inputs, if the inputs are utilized as inputs for node operations, without manual adjustments to the structure of the node graph. For example, in terms of programming, the list of inputs in the *Group Input*-node can be interpreted as the input parameters of a function, which is the node graph. The input values can be adjusted for each object that contains the *Geometry Nodes*-modifier with the same node graph in the modifier stack. A *Group Output*-node must be present in the node graph to output the result, which is often the changed geometry. In the example, which can be seen in figure 3.3, the node graph moves the geometry of the input object by one unit on the x-axis and returns the moved geometry.

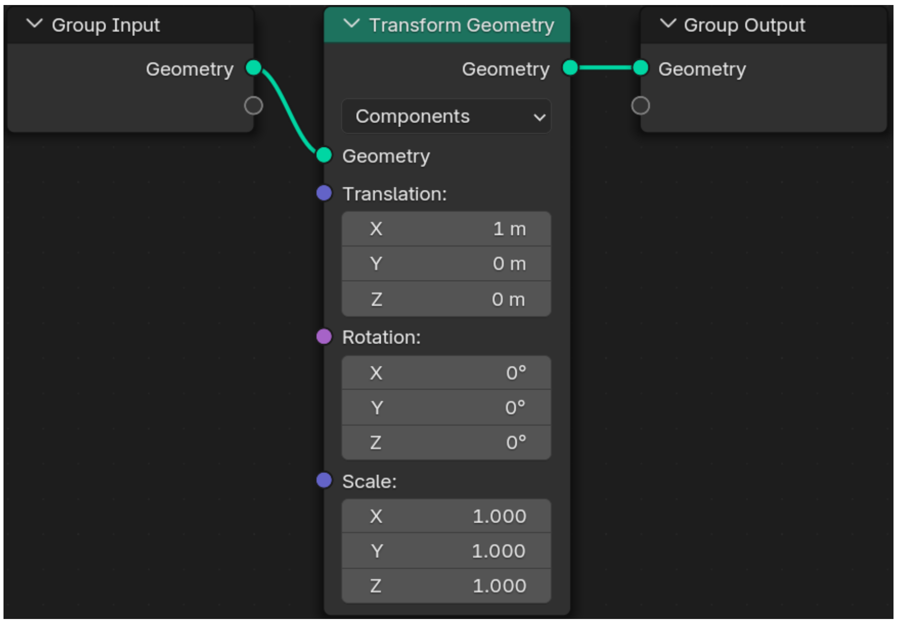


Figure 3.3 Example of a Node Graph

For this thesis, Geometry Nodes were used to create the objects of the dining room scene and for the distribution of objects such as tableware on the table, the table placement in the dining room or the placement of indoor lights. For creating the objects, new mesh primitives, such as a cube, cylinder, curve or circle, are created with nodes and are used as the base geometry. Then, the base geometries are altered by selecting sets of vertices, edges or faces and apply different node operations that use the values of a *Group Input*-node as input. This allows for the adjustment or randomization of object characteristics without manual modeling. To distribute objects, the existing geometry of an object is used to determine a distribution area by selecting a set of faces or using the volume of the object. *Points* will be generated on the selected faces or in the object's volume with constraints such as minimum distance to avoid object clipping. The generated points are then used to generate the objects on them. The node graphs for creating and distributing the dining room objects will be explained in detail in section 3.2.

3.1.3 Materials and Shader Nodes

Materials consist of three shaders which define the appearance of the surface, the volume of the object and the displacement of the surface. The surface shader controls material properties like color, texture, light interaction or metallic. The volume shader can be used for smoke, fire, fluid or glass objects. The displacement shader affects the appearance of the geometry, which can add additional surface details, without changing the geometry data.

For creating materials, Blender offers the approach named *Shader Nodes*, which can also be used for lights and backgrounds. It was developed with the concept of the node graph, which is used for Geometry Nodes and presented in detail in section 3.1.2, but offers a different set of nodes for material creation.

For the implementation of realistic materials, physically-based shading was implemented by introducing the following nodes: *Principled BSDF*, *Principled Hair* and *Principled Volume* shader. To achieve realistic materials, the shaders contain properties that follow the physics of the real world, such as being energy-conserving, which means the materials cannot reflect more light than the amount of incoming light [54]. The materials created for this thesis utilize the *Principled BSDF*-shader, which is based on the OpenPBR Surface shading model [3]. For the light interaction, the shader uses a bidirectional scattering distribution function (BSDF) [6] that calculates how the incoming light is scattered by the surface. Light can be reflected, refracted into the mesh, or absorbed.

To achieve a variety of realistic materials, the *Principled BSDF*-node has multiple inputs to customize the material properties such as the base color, roughness, metallic, index of refraction (IOR), alpha, normals, subsurface, transmission, coat, sheen and emission. *Base Color* is the general color of the material. *Roughness* determines how sharp or diffuse light is reflected. *Metallic* determines whether the material is metallic or not, which also affects the way light is reflected. *IOR* defines how the material bends the direction of incoming light rays passing through the material. The higher the index, the closer the light ray direction will be to the surface normal. *Alpha* sets the transparency of the surface material. *Normal* controls the normals of the material. It is often used to add additional details, such as scratches, to the material without affecting the geometry data of the object. *Subsurface* allows for subsurface scattering, which determines how light scatters below the surface. It could be used for materials such as skin, milk or wax. *Transmission* allows the material to reflect the incoming light on the surface and also transmits it to the interior of the object. This input can be used for materials like glass or liquids. *Coat* can be used to add a coating on top of the material to simulate materials such as clearcoat, lacquer or car paint. *Sheen* can be used to simulate small fibers or dust on the material, adding reflection near the edges like in fabrics. *Emission* determines how much light is emitted from the material and ignores the energy-conserving property. The effects of the different inputs can be seen in figure 3.4. The first row shows the application of the base color for different types of materials. In the second row, the transition from a non-metallic to a metallic material can be seen. The third row shows increasing levels of roughness and thus the difference between sharp and diffuse reflections on the surface. The fourth row shows the increasing IOR levels, where the reflective light ray direction approaches the normal direction of the surface. The fifth row shows the transition from a transparent material to an opaque material.

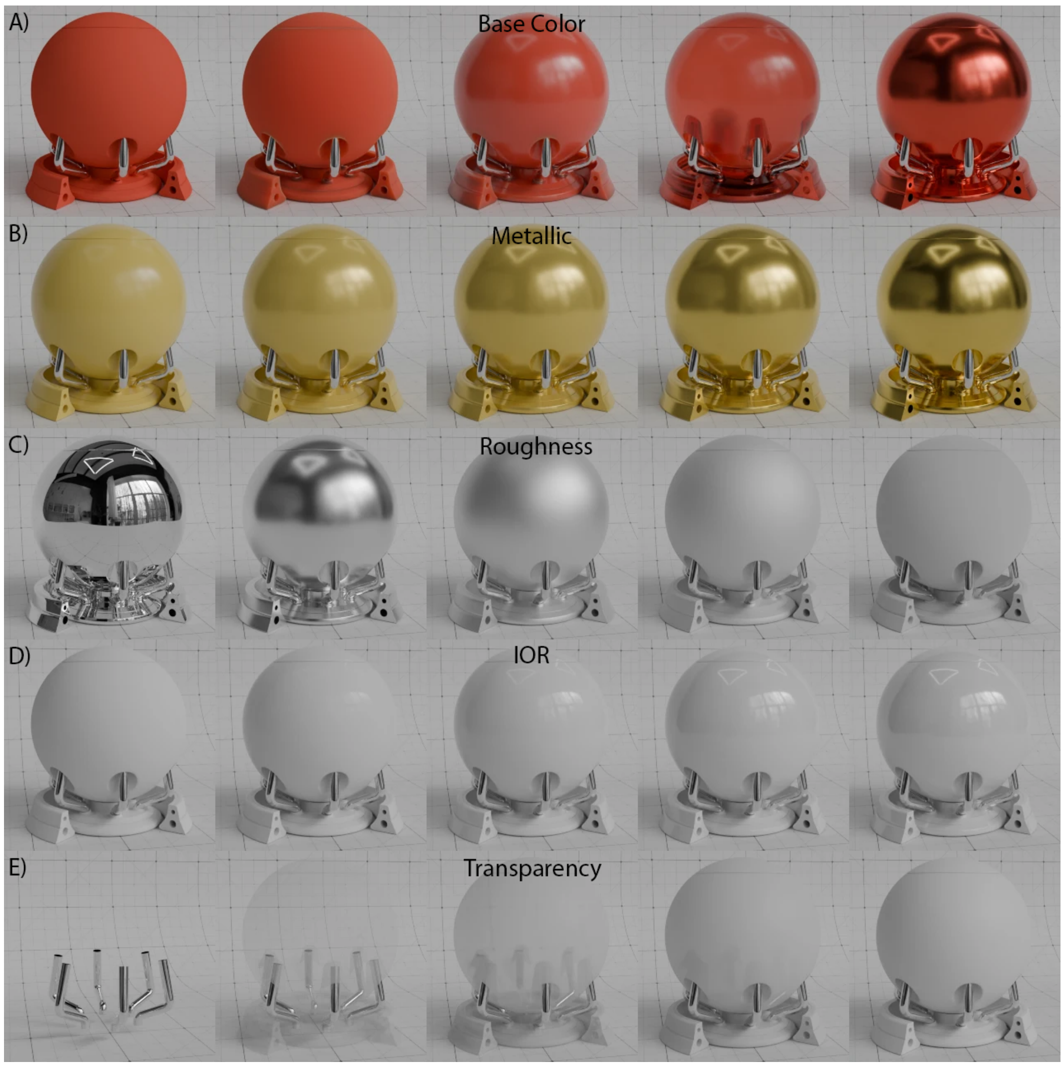


Figure 3.4 Applications of the Principled BSDF-Shader

The input of the *Principled BSDF*-node can be adjusted by single values or so-called texture maps, which are images that contain material information in the form of RGB color data. The advantage of texture maps is that they offer more material information compared to single values, because RGB color data is saved in every pixel. They are often used for the base color, metallic, roughness, normals and displacement input. The RGB color data of each texture map match the different required inputs of the *Principled BSDF-node.* For example, color maps can consist of any RGB value, but the roughness map must be a grayscale image in which black-colored pixels determine where the material reflects incoming light sharply, and white-colored pixels determine where the material reflects incoming light diffusely. An example of different texture maps such as a base color, metallic, roughness and normal map can be seen in figure 3.5.

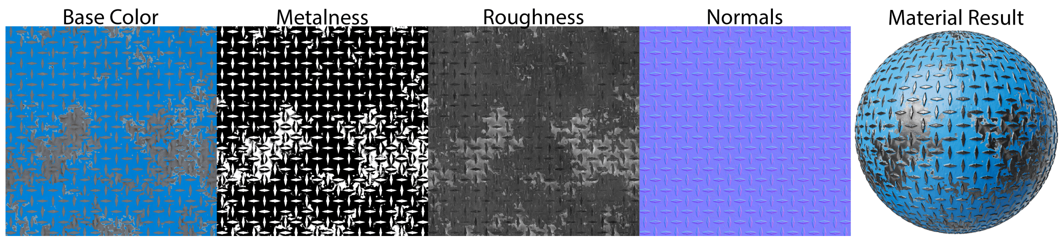


Figure 3.5 Examples of Texture Maps

Texture maps can be created manually with a different software such as Adobe Substance 3D [1]. To utilize the created texture maps, they must be exported and then imported as images in Blender. A different approach is to procedurally generate texture maps in Blender with *Texture Nodes*, which generate different procedural textures such as brick texture. By combining and modifying these textures with additional node operations, the different texture maps can be generated. *Texture Nodes* can also be used to generate masks to modify only parts of the material.

To apply and position the texture maps to the surface of the object, texture coordinates must be specified for the texture maps, which can be automatically generated coordinates from: the vertex positions of the mesh, the object normals, the UV map of the object, an object, the object itself, the position coordinate in camera space, the location of the object on the screen or the direction of the reflection vector. For manually created texture maps, an existing UV map of the object is required for the correct projection. As the aim of the synthetic data generator is to automate the data generation process and randomize the dining room objects, the manual creation of UV maps is not suitable. Automating the creation of UV maps is time-consuming and complex because of the changing geometry. In addition, the aim of the data generator is to also randomize material properties such as color, which cannot be done with manually created texture maps.

In this thesis, every material in the dining room scene was generated procedurally to enable customization and randomization of the material properties. Furthermore, procedurally generated materials do not require UV maps and can use the object itself as the source for texture coordinates.

3.1.4 Compositor

For this thesis, the compositor is used to generate the corresponding ground truths of the created images. The compositor allows users to edit rendered images with *Composition*-Nodes. For example, the colors of the image, depth of field or exposure can be adjusted after rendering. Like the *Geometry*- and *Shader Nodes*, the compositor was developed with the concept of the node graph.

Blender offers multiple outputs that can be utilized for editing images after rendering, for example, a depth map that visualizes the distance to visible surfaces. One of the outputs are binary masks of objects or materials in the rendered image. To create the masks, the objects or materials must be assigned with an index in the object or material properties. In addition, the option for the output must be activated in the view layer properties. Activating the option will create an output socket for the mask in the input node of the compositor named *Render Layers*. To read and use the binary mask, the *ID Mask* node must be used with the corresponding index for the object or material. The ceramic material, which is used for the plate object, will also be used to identify the plate objects in the images and create the binary masks, which are used as ground truth. In addition, the compositor is used to set the image file formats, name them and to save both images under a specific path.

3.1.5 Blender Python API

The Blender Python API allows users to interact with Blender scenes through Python scripts. Tasks such as automating object modeling, analyzing Blender-specific data, modifying Blender settings, or creating custom user interfaces or plugins, are possible with the API. Every built-in Blender function can be used in the Python scripts. In addition, by utilizing Python, the API extends Blender's functionality by the available Python packages, which can be imported into the scripts. To automate and execute the data generation pipeline, a Python script is implemented that randomizes and logs scene settings and object characteristics through the Blender Python API. In addition, a custom user interface was implemented for Blender to allow users to use the functions that are executed in the data generation pipeline, which will be explained in detail in section 3.5.

3.2 Dining Room

In addition to the plate object, which is the focus of this thesis, tables, chairs, dining rooms and tableware objects such as spoons, forks, knives and glasses were modeled to add fidelity to the dining room scene, as in the work of Delgado et al. [19].

All objects are generated procedurally with *Geometry Nodes* and *Shader Nodes*, and allow for randomization of their object and material characteristics for data variability. The associated node graphs for the creation of the objects and the materials are created with the approach described in section 3.1.2 and section 3.1.3.

The value range for the dimensions of the dining room objects, for example, length or thickness, are true to scale. Measurements for different parts of an object are inspired by available online sources, such as dimensions.com [21], or guidelines and standards. The range of some characteristics were chosen empirically, because there were no available data or no standards as the design of dining room objects, such as tables, chairs, glasses or spoons, can be a creative choice. Nevertheless, the ranges were chosen under the constraint that it allows for data variability, but it does not alter the object's fundamental perception. The materials for the furniture and dining room are implemented based on image references that can be found online with the search term "dining room" on Google Images. The materials of the tableware are based on different online sources [84, 49, 10] that discuss the usage of different materials for tableware.

Potentially, the additional objects could also be included for object detection for future work. In the following, the creation of each tableware object will be presented by visualizing the geometry in each step of the generating process and explaining those steps. The implemented materials will be explained in section 3.3.

3.2.1 Plate

The plate object is used as the detection object for the convolutional neural network (CNN). In this thesis, the plate is limited to a circular shape and white porcelain as material. This limitation can be addressed in future work.

As can be seen in figure 3.6, which shows a plate's half of a cross section, a plate can be divided into components that are named well, lip, rim and base [45]. The following attributes can be adjusted: plate radius, plate height, plate thickness, well radius, lip radius, lip height, rim radius, rim height, base, base radius, base height and base width.

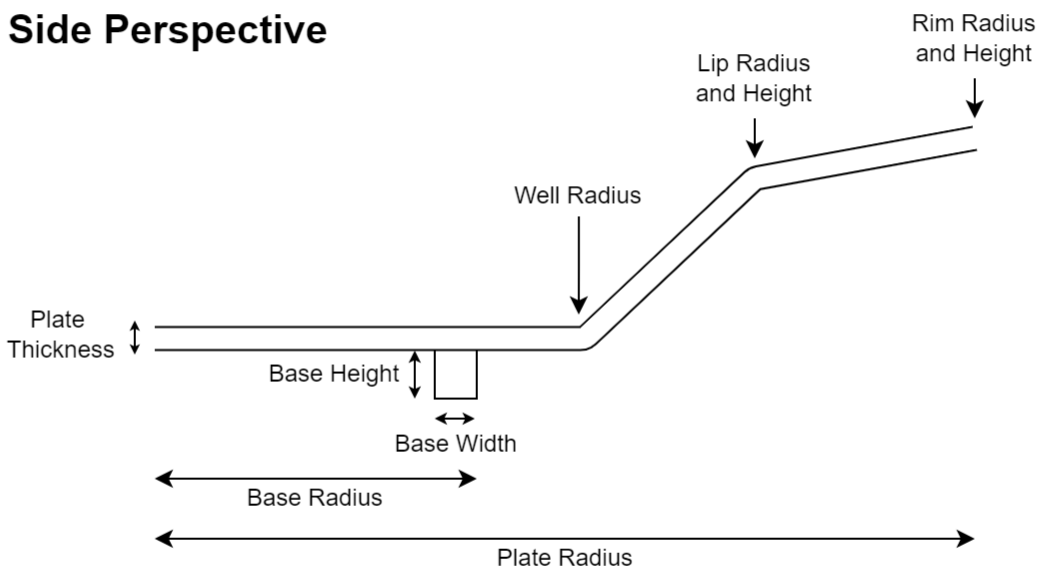


Figure 3.6 Plate Attributes

Plate Generator

The procedural generating process of a plate object can be seen in figure 3.7. For the base geometry, the plate generator creates a flat curve object with the length of the plate radius input. Next, the curvature of the object will be adjusted by the well radius, lip radius, lip height, rim radius and height input. In the third step, thickness is added to the curve object. Adding thickness can result in intersecting geometry in the corners of the plate curvature, which can be seen in figure 3.7d. As the geometry is connected, the intersecting geometry will be removed by selecting the indices of the vertex before and after the intersection point and deleting the vertices with the indices in range of the indices of the selected vertices. Then, the selected vertices are merged on the intersection point. To generate the plate base (see figure 3.7e), geometry of the bottom side of the curve will be removed between the points set by the base radius and base width. The points are then extruded and merged to create the plate base with the base height input. Afterward, the plate object will be centered to the origin point and placed on the ground. The current generated geometry is a half of the cross section of a plate. For the final step, the plate object will be generated by using the *Screw*-Modifier, which can generate circular objects with a cross section of the object (see figure 3.7h).

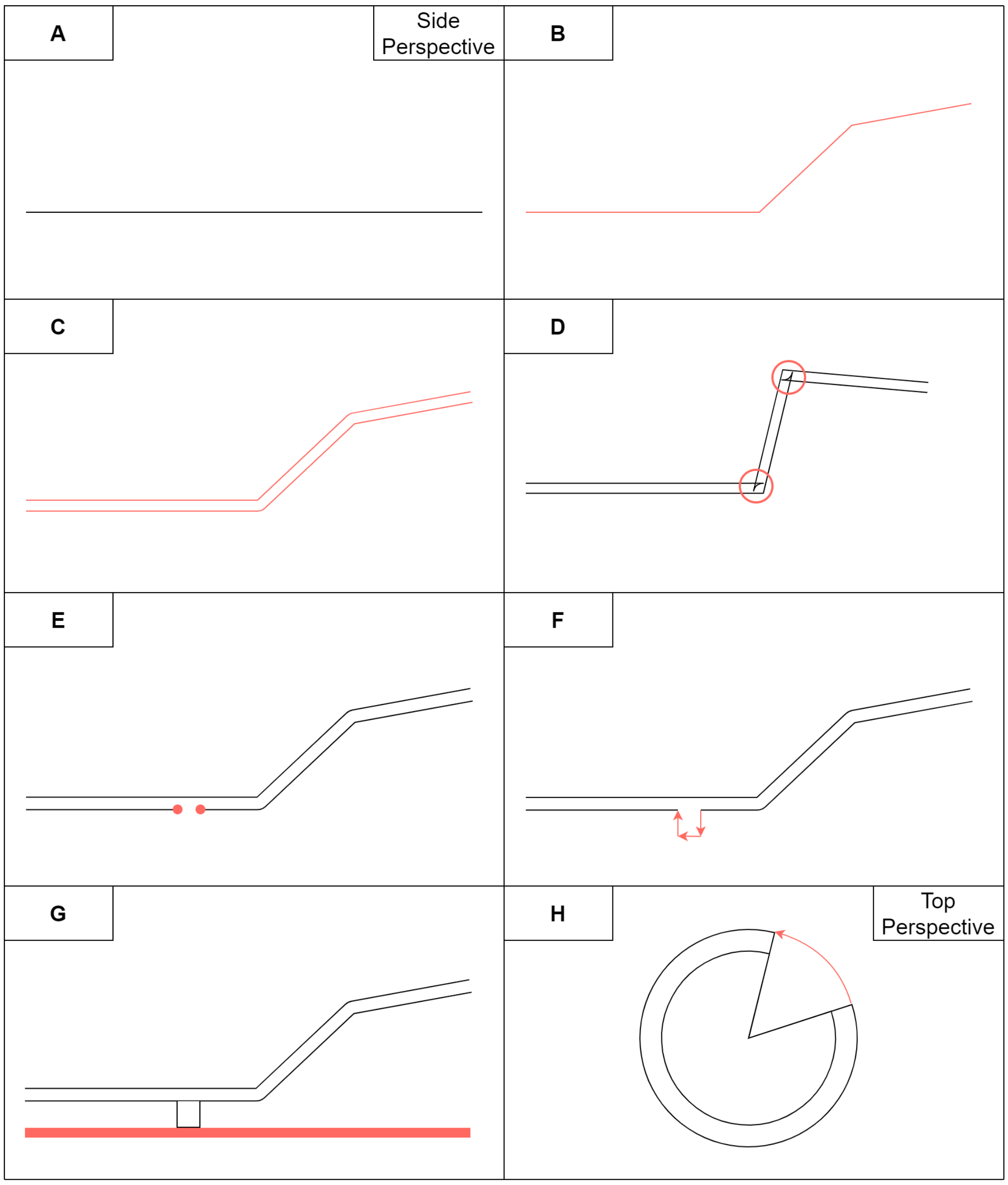


Figure 3.7 Generating Process of the Plate Object

3.2.2 Fork

A fork can be divided into tines, roots, the back, the neck and the handle [29]. The adjustable attributes can be seen in figure 3.8 and are length, thickness, amount of tines, tine length, tine curvature, root curvature, back length, back width, neck length, neck height, handle length, handle width, handle end height, handle end width and handle end curvature. The material of the fork object is stainless steel.

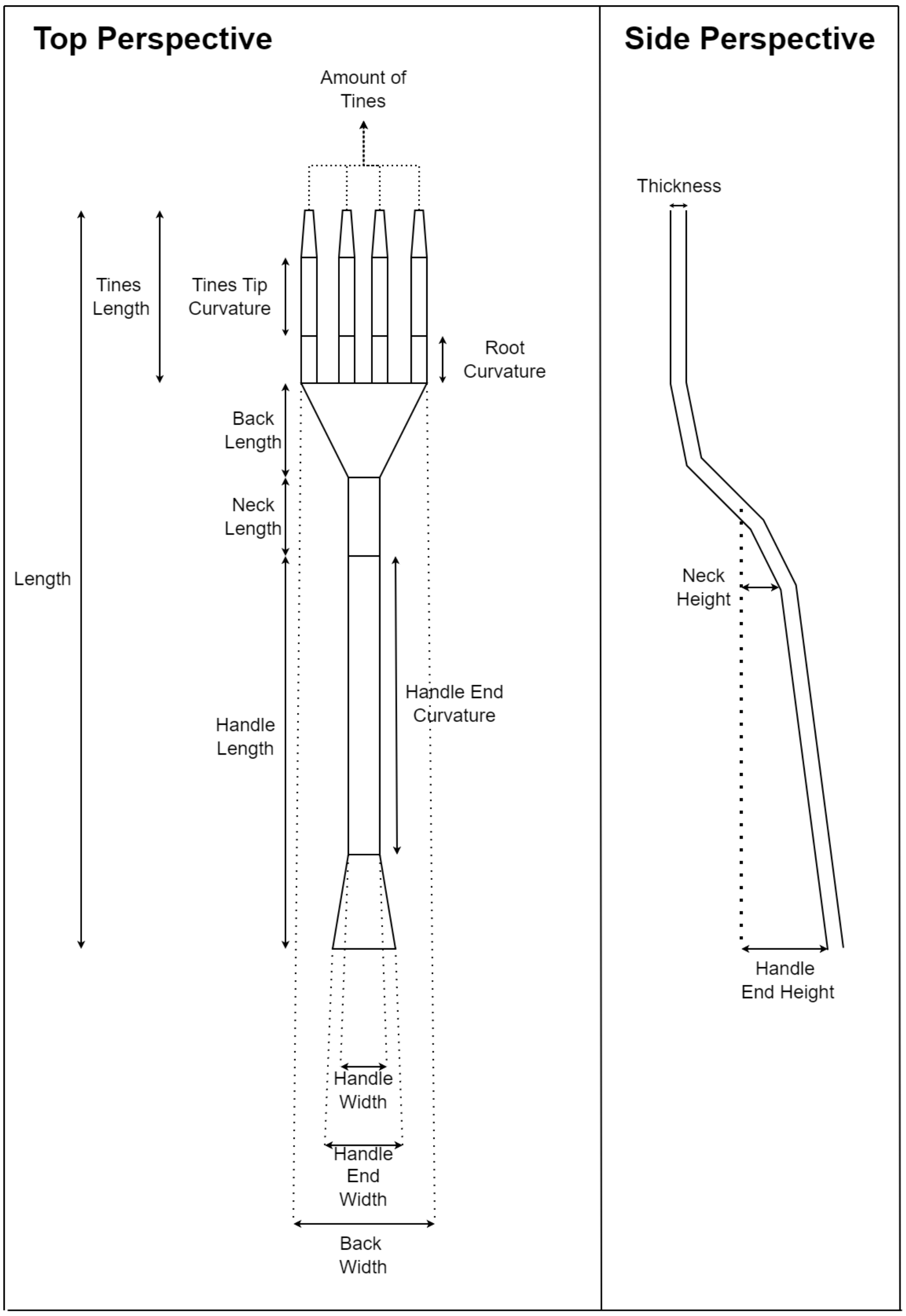


Figure 3.8 Fork Attributes

Fork Generator

The process of creating the fork object can be seen in figure 3.9. The base geometry is a plane with inner edges. The edges of one side of the plane are alternatively selected, extruded to the length of the tine length. For the curvature of the tines, the edges are moved toward the horizontal center of the object. The back of the fork is created by selecting the edges on the opposite side of the tines and extruding them to the back length. The edges are then horizontally scaled so that the length corresponds to the width of the handle. For the fork shape, the back is vertically curved. Next, the edges are extruded three times to create the neck, the handle and the end of the handle and adjusted using the corresponding attribute inputs. Like in the spoon generation process, the *Solidify*-Modifier will be used to generate geometry to add thickness to the geometry, and the *Bevel*-Modifier will be used to smoothen the edges of the object by generating additional edges.

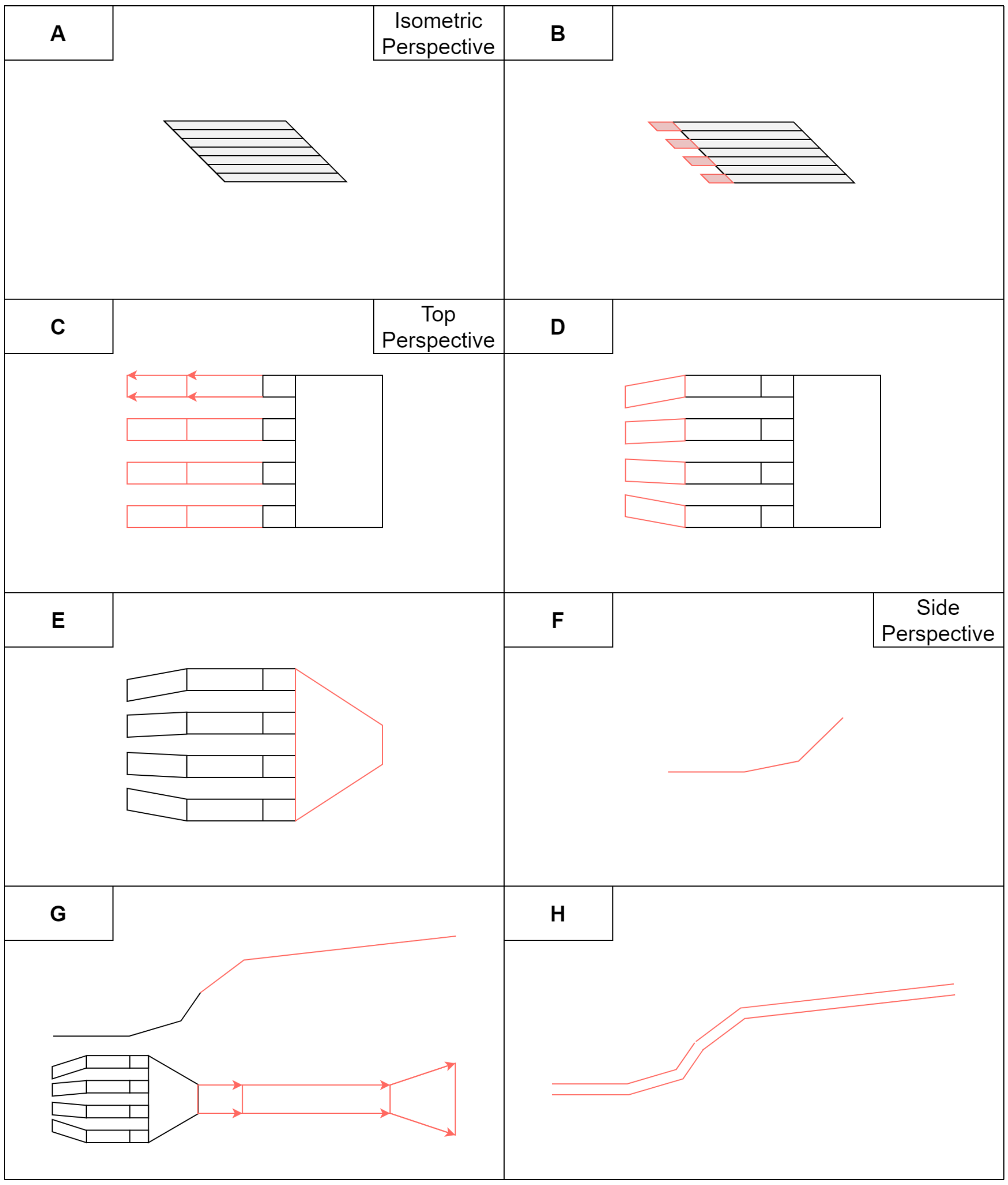


Figure 3.9 Generating Process of the Fork Object

3.2.3 Knife

A knife consist of a blade and a handle. To customize the knife geometry, the following attributes are implemented: width, length, thickness, blade thickness, blade length, blade tip intensity, blade tip curvature, blase base intensity, blade base curvature, handle length, handle width, handle end curvature and handle end width. The material of the knife object is stainless steel. An overview of the different knife attributes can be seen in figure 3.10.

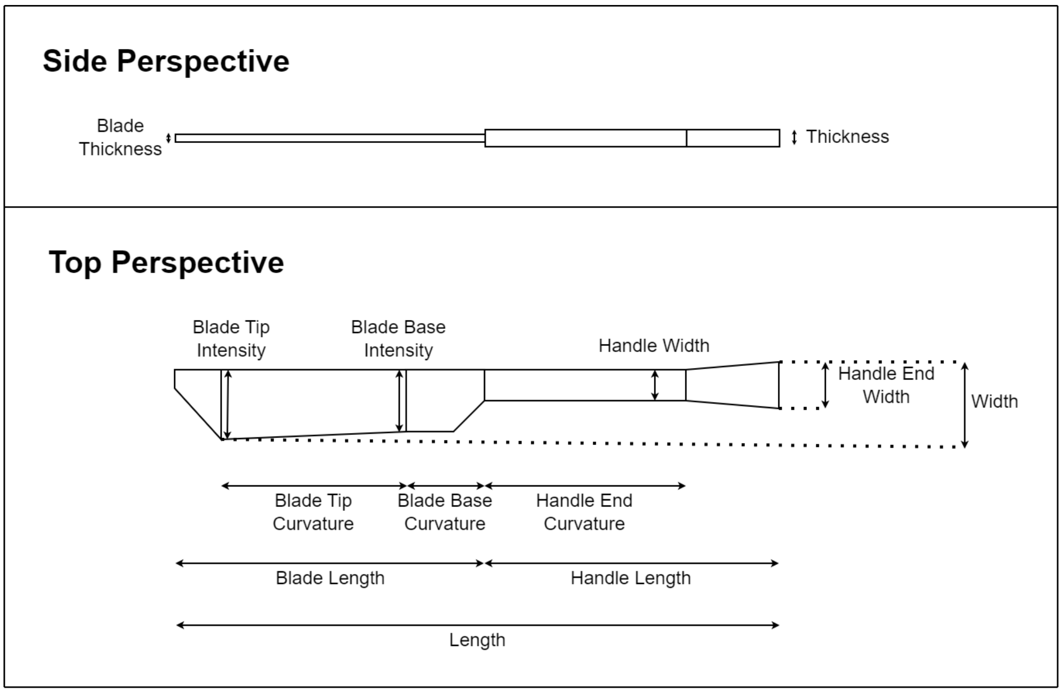


Figure 3.10 Knife Attributes

Knife Generator

The procedural generating process of a knife object can be seen in figure 3.11. The generator creates a cube, that will be the knife handle, with an inner edge, which is used for the handle end curvature. Next, the cube is adjusted according to the handle end curvature, handle length and handle width input. For the width of the handle end, the face of the end of the handle is scaled horizontally. Extruding and scaling the face on the opposite side of the handle end, creates the blade base. It can be adjusted by the blade base intensity and curvature input. The intensity attributes adjust the curvature of the knife blade. Furthermore, the face of the blade tip is extruded two times to create the rest of the blade and the blade tip, which can also be adjusted by the corresponding inputs.

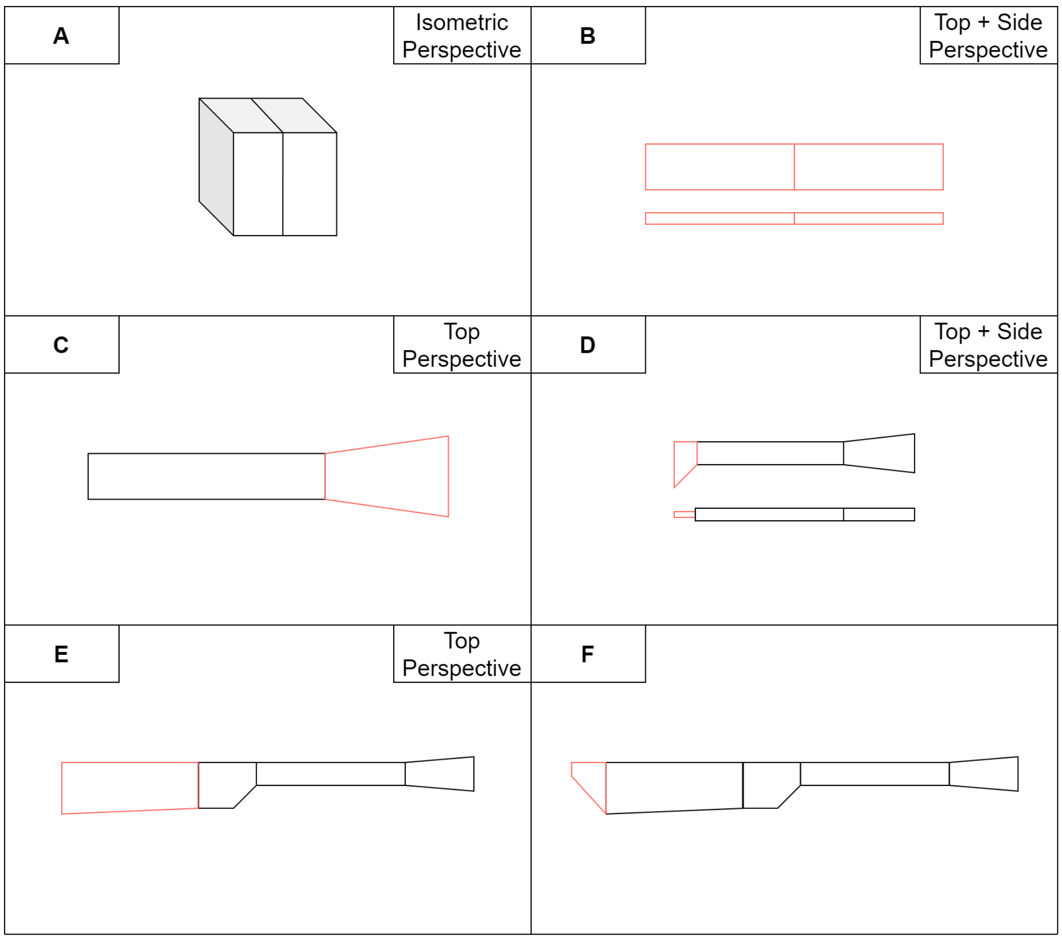


Figure 3.11 Generating Process of the Knife Object

3.2.4 Spoon

As can be seen in figure 3.12, a spoon can be divided into bowl, neck and handle [80], and the components can be adjusted by the following attributes: length, thickness, bowl width, bowl length, bowl depth, neck length, neck height, handle length, handle width, handle end height, handle end width and handle end curvature. The curvature attributes add inner geometry, that does not affect the shape of the base geometry, to adjust the smoothness at corners of the geometry such as the connection between the handle and the spoon bowl. The material of the spoon is stainless steel.

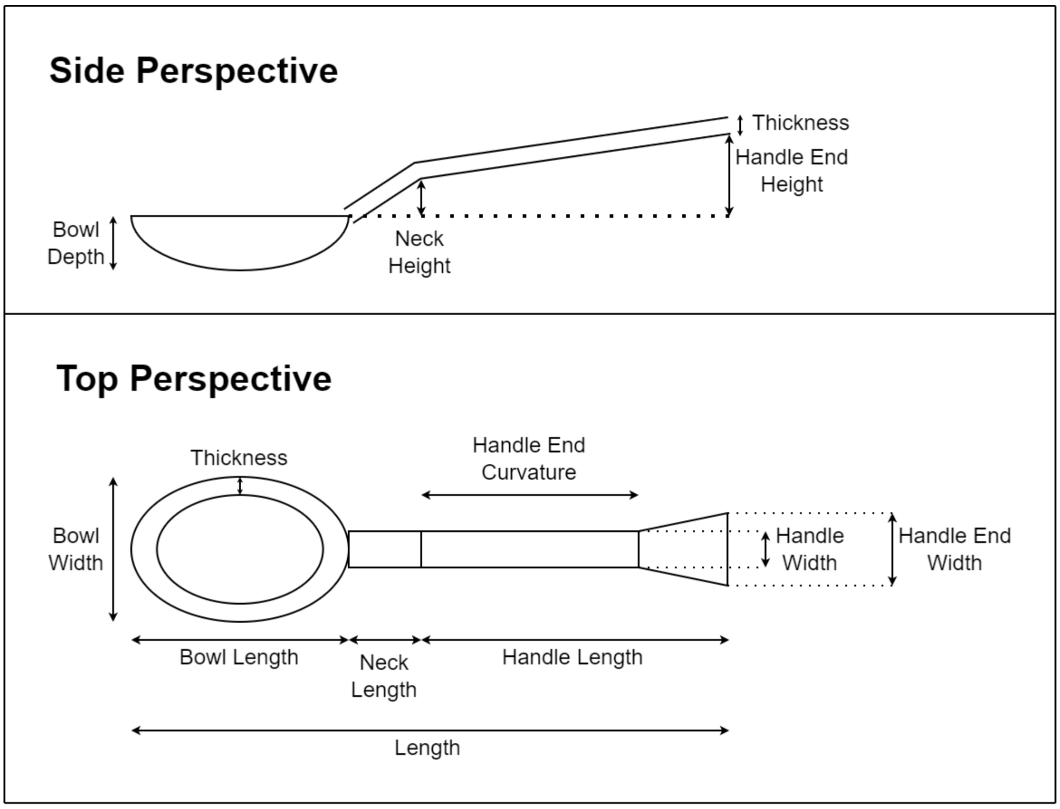


Figure 3.12 Spoon Attributes

Spoon Generator

For generating a spoon object, which can be seen in figure 3.13, a sphere object is created as base geometry. It is then cut in half and shaped to the spoon bowl with the bowl length, bowl width and bowl depth input. To create the neck and handle, two vertices are created at the upper edge of the spoon bowl, with the distance between the two vertices being the neck width. The two vertices are then connected with an edge and extruded three times to create the neck, the handle and the end of the handle. The position of the three components are adjusted by the corresponding input attributes. Last, the *Solidify*-Modifier will be used to generate geometry to add thickness to the geometry, and the *Bevel*-Modifier will be used to smoothen the edges of the object by generating additional edges.

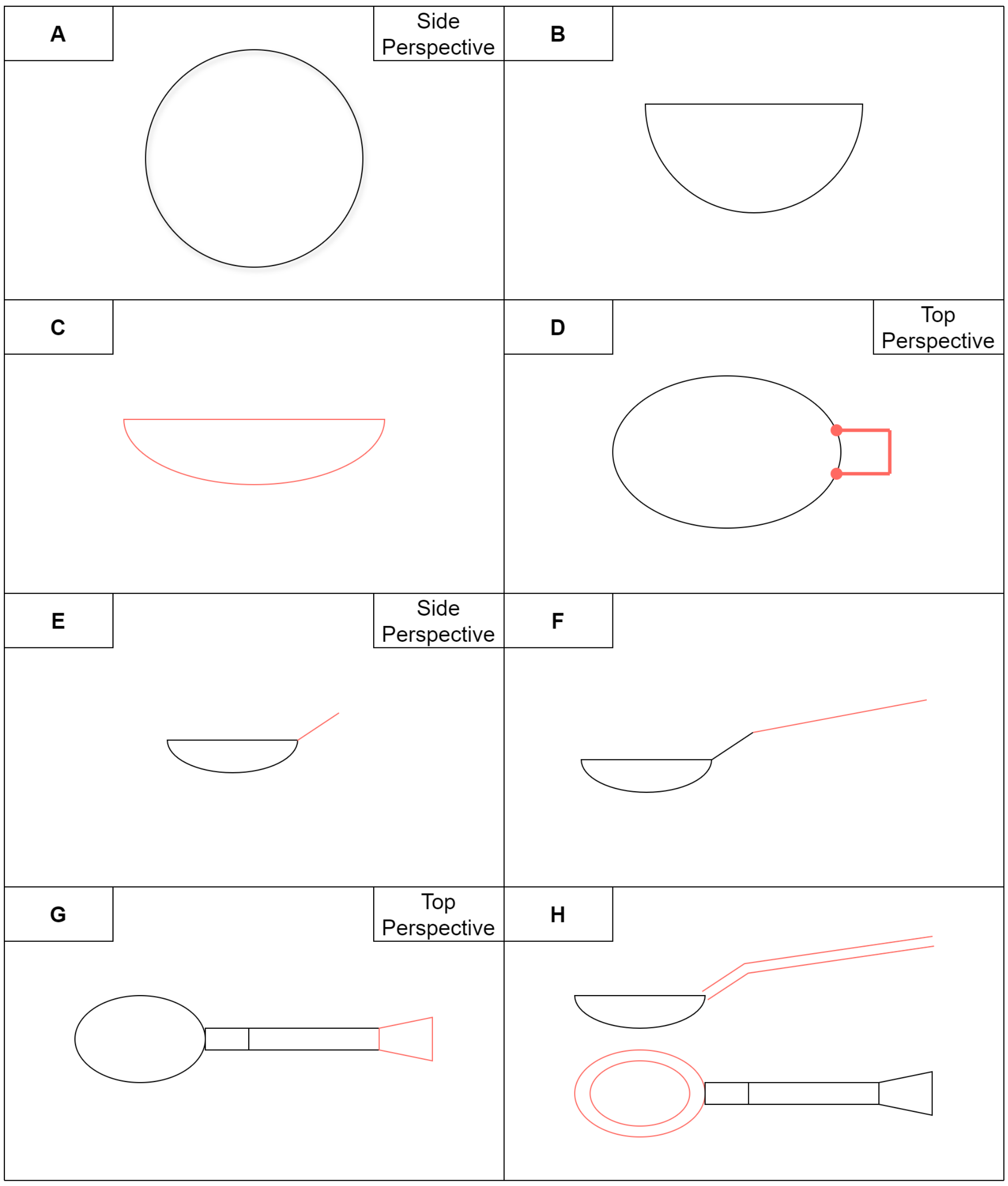


Figure 3.13 Generating Process of the Spoon Object

3.2.5 Glass

The different components of a glass are called base, rim and bowl [26]. To generate a variety of drinking glasses, the following attributes, which are visualized in figure 3.14, are implemented to customize glasses: height, thickness, base diameter, base curvature, mid curvature height, mid curvature diameter, rim curvature, rim diameter and bowl curvature. For the material, glass is implemented.

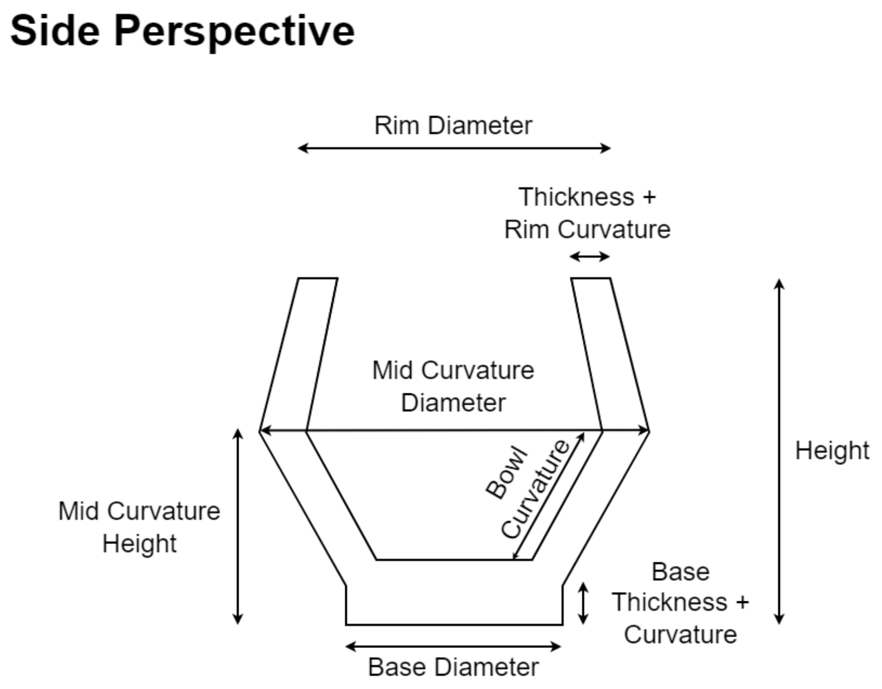


Figure 3.14 Glass Attributes

Glass Generator

The glass generating process, which can be seen in figure 3.15, starts by creating a circle with the base diameter. In the next step, the edges are extruded toward the center and merged to close the circle and form the base. Afterward, the outer edges are selected and extruded upward three times for the base thickness, mid curvature height and general height of the glass. For the glass thickness, the upper edges are extruded with the desired thickness toward the center. To create the glass bowl, the upper edges are extruded downward four times, considering the different diameters of the base, mid curvature and rim, to ensure the glass thickness and to add the bowl curvature. Last, the hole in the glass bowl is closed by extruding the edges toward the center and merging them at the center, as in the generating step, which can be seen figure 3.15b.

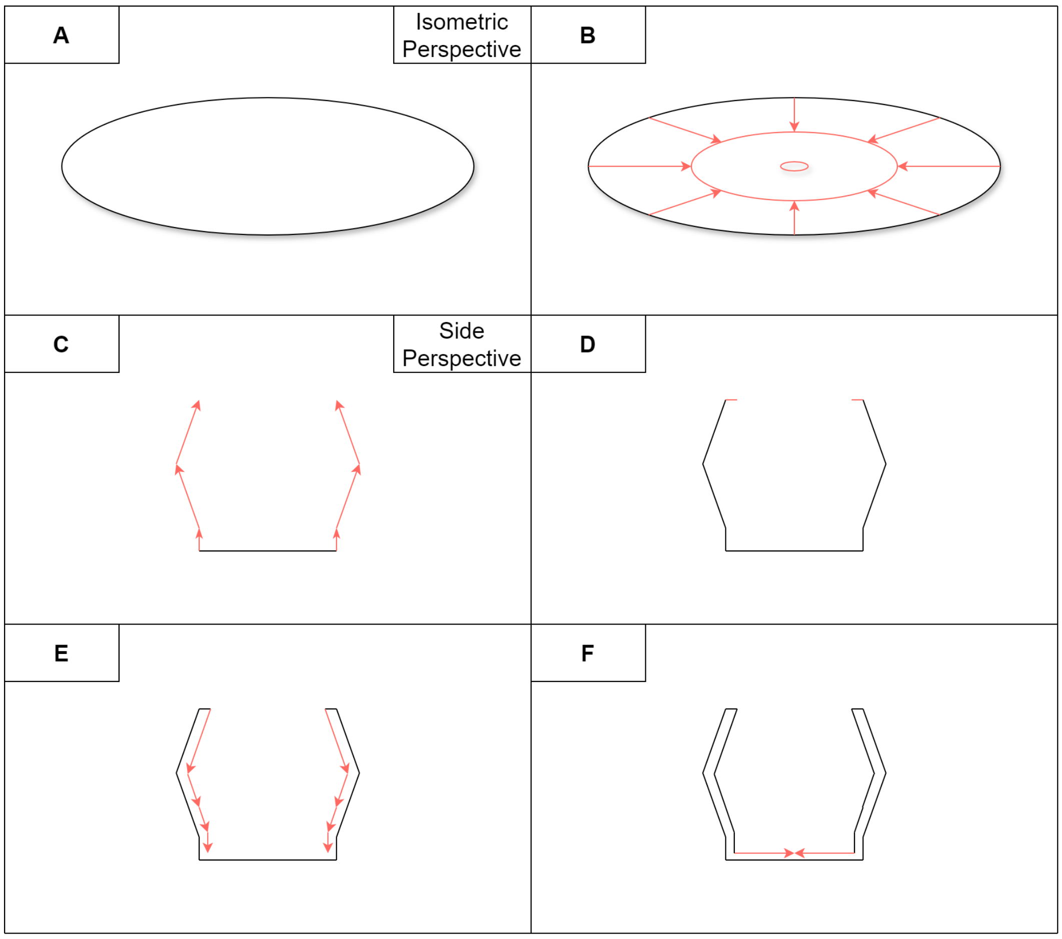


Figure 3.15 Generating Process of the Glass Object

3.2.6 Table

A table consists of the table top, table legs and the apron [24], which is the supporting structure under the table top. To adjust the different parts of a table, the following attributes, which can be seen in figure 3.16, are implemented for the table generating process: height, width, depth, round/rectangle table top, round/rectangle apron, table top thickness, table top curvature, apron thickness, apron size reduction, leg width, leg thickness and leg angle. In this thesis, the table size was limited to a table for four people, the shape of the table was limited to a rectangle or a circle, and the shape of the table legs are also limited to a rectangle. The material of the table top can be wood, wood planks, marble, veined marble, plastic or ceramic. The material of the table legs and the apron can be metal or wood. In addition, the colors of the material are randomized by random RGB values or different color palettes. The materials and color palettes will be explained in detail in section 3.3.

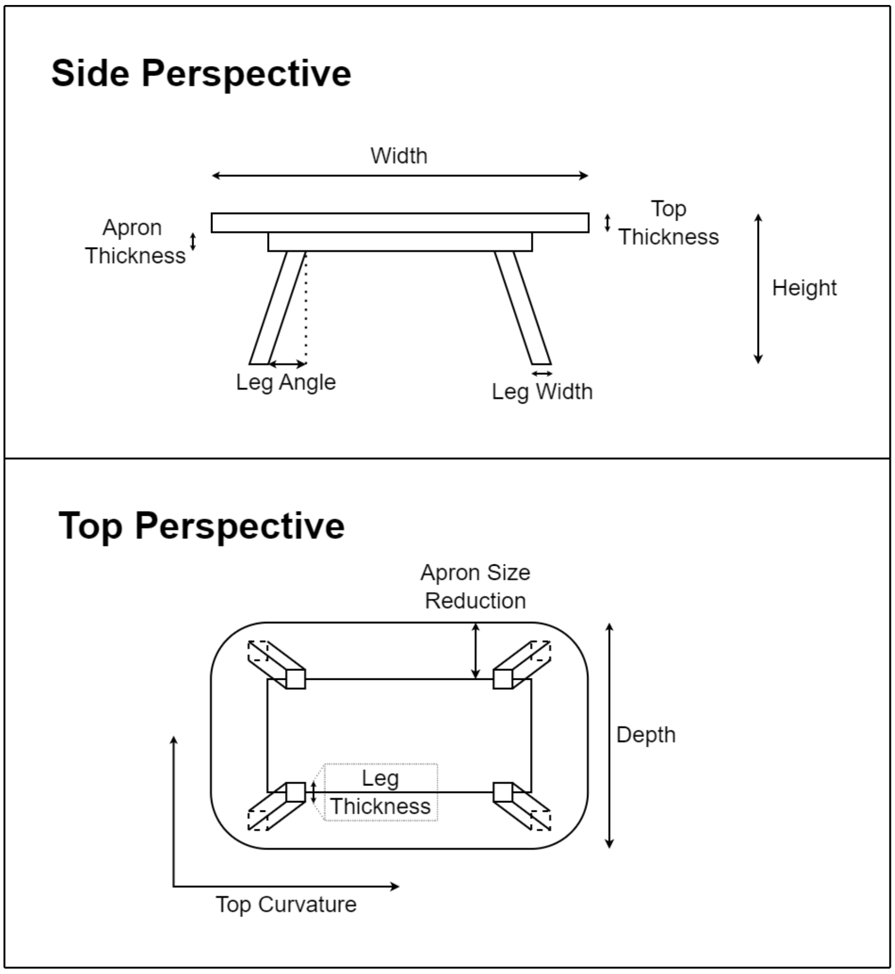


Figure 3.16 Table Attributes

Table Generator

Figure 3.17 shows the generating process of a table object. First, the table top will be created by generating a rectangle or circle, which will then be extruded vertically to create the thickness. Next, the surface area that does not protrude over the edges of the table top will be calculated to determine the maximum size of the apron (see figure 3.17c). A second rectangle or circle with the maximum apron size will be created and vertically extruded to create the table apron. Like the table top, the shape of the apron can be a rectangle or a circle (see figure 3.17e). The maximum size of the apron is reduced so that the table legs can be generated at the corners of the apron without protruding over the edges of the table top. Last, the bottom faces of the table legs are selected and moved away from the center to angle the table legs. The maximum angle of the table legs is reached when the bottom faces reach the edge of the table top. After the generating process, a *Bevel*-Modifier is used for smooth edges.

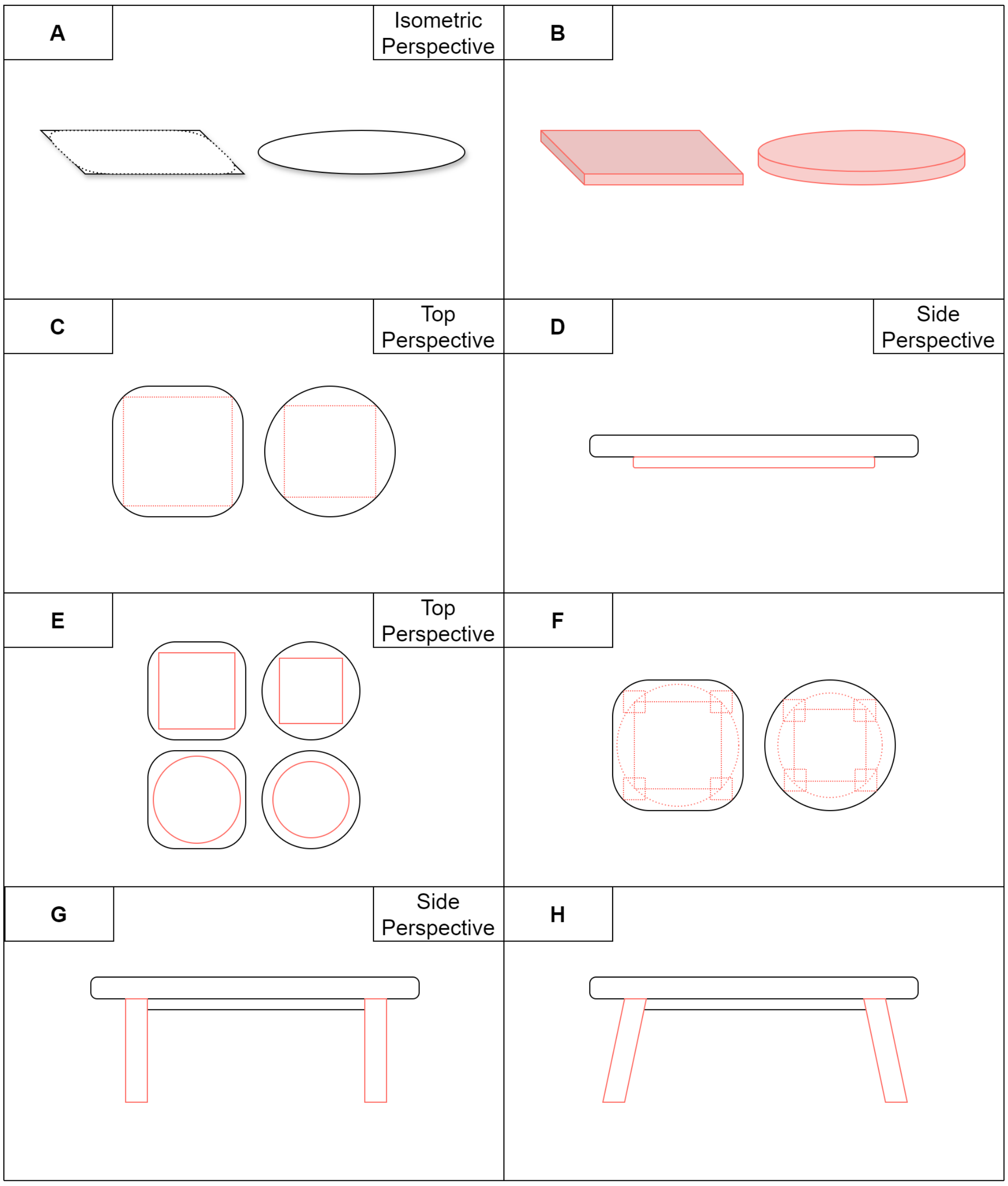


Figure 3.17 Generating Process of the Table Object

3.2.7 Chair

In this thesis, the chair object is limited to slat and ladder back chairs. These types of chairs consist of a top rail, back posts, cross rails, slats, a seat, a seat rail and chair legs [57]. In addition, the shape of the seat and seat rail is limited to a rectangle and a circle. The shape of back posts, cross rails, slats and chair legs is also limited to a rectangle. To customize the different components of the chair object, the following attributes, which are visualized in figure 3.18, are implemented: height, width, depth, curved/straight back post, back post angle, back post width, back post thickness, top rail height, top rail thickness, amount of cross rails, cross rail height, cross rail thickness, amount of slats, slat width, slat thickness, rectangle/round seat, seat height, seat thickness, seat curvature, rectangle/round seat rail, seat rail reduction, seat rail thickness, leg width, leg thickness and leg angle. A visualization of the different attributes can be seen in figure 3.18. The material of the seat can be wood, wood planks and plastic. The material of the top rail, cross rails, back posts, slats, seat rail and legs is wood, which has different color palettes to randomize the material color.

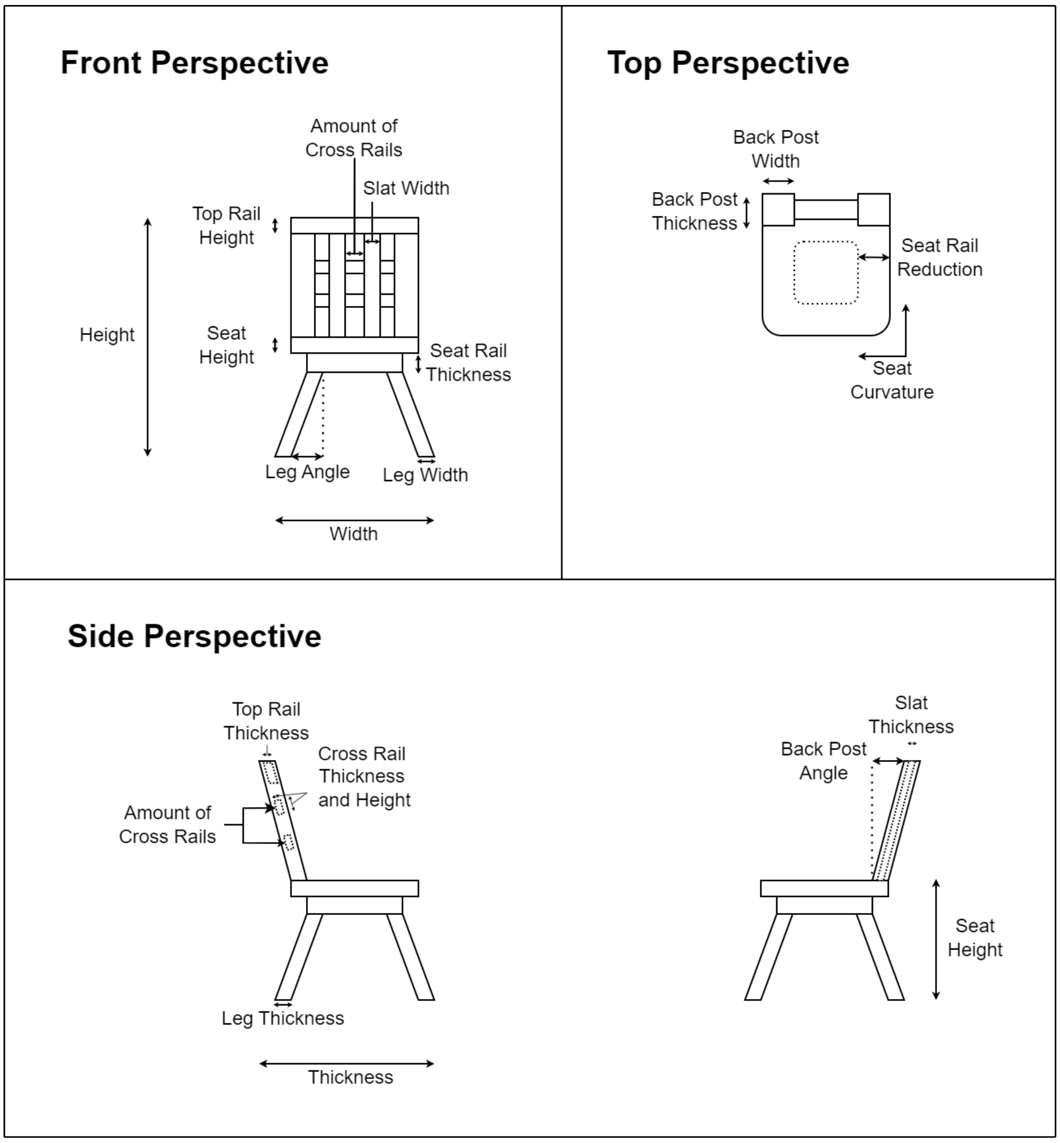


Figure 3.18 Chair Attributes

Chair Generator

The seat, the seat rail and the chair legs are created using the same approach as for the table object (see section 3.2.6). The corresponding attributes are adjusted so that the chair components are true to scale. To create the chair object, the top rail, back posts, cross rails and slats were added, which is visualized in figure 3.19.

First, the back posts with the desired length, width and thickness are generated on the corners of the upper face of the seat, such that they do not protrude over the edge of the seat. The upper faces of the back posts are selected and moved horizontally to angle the back posts. Next, vertices on the upper part of the back posts are copied to create the top rail that has the same shape as the back posts, because the back posts can also be curved and different shapes can result in intersecting geometry. By extruding the copied vertices, the top rail is created with the desired height and thickness. The same approach is used to create the cross rails. For the positions of the cross rails, vertical edges of the back post from the seat to the lower edge of the top rail are copied and realigned for even space between the cross rails. Back post vertices at the cross rail position and vertical adjacent vertices are copied and extruded to create cross rails with the desired height and thickness. To create the slats, a line with evenly spaced vertices is generated that has the same length as the distance between the inner edges of the back posts. To create the slats, the vertical edge, where the vertices are used to position the cross rails, is copied and extruded to the desired width and thickness. Last, a *Bevel*-Modifier is used to smooth the edges of the chair object.

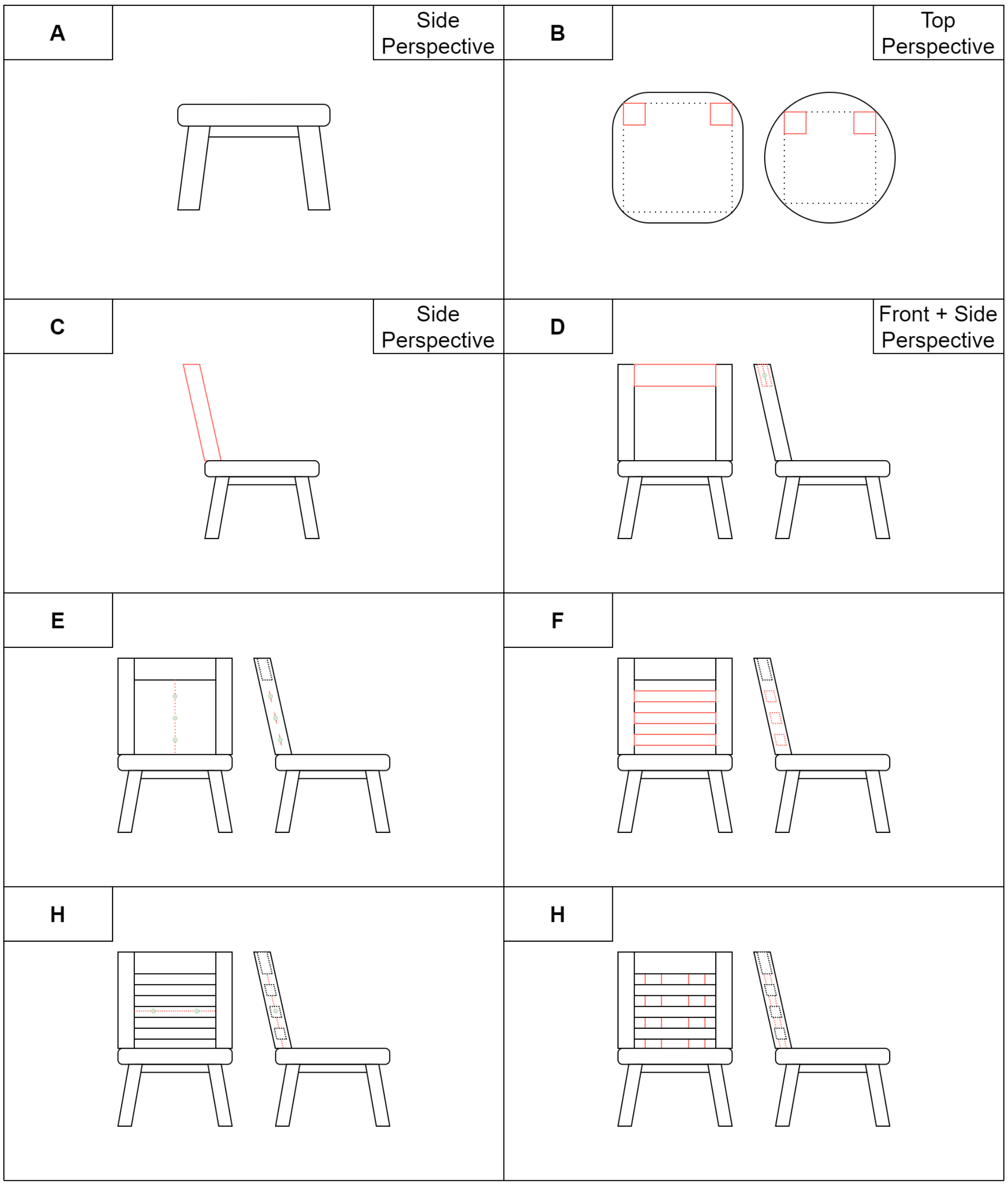


Figure 3.19 Generating Process of the Chair Object

3.2.8 Room

For the background of the generated images, a dining room was implemented. It consist of a floor, baseboard, windows and a ceiling. In addition, the dining room object is used to randomize the position of the table, the lighting of the scene and the camera position. The lighting distribution and camera placement will be explained in detail in section 3.2.10 and section 3.2.11. The size of the room area was limited to forty square meters, which is a recommended size from multiple sources [32, 88, 39]. The room can be customized with the following attributes: room area, dining room circulation space, wall height, wall thickness, baseboard height, baseboard width, amount of windows, vertical window position, window height, window width, window frame thickness, window frame depth, window depth, window thickness and glass thickness. The different attributes are visualized in figure 3.20. The material of the floor can be carpet, wood planks, checkered marble, rectangular stone tiles, square stone tiles or marble tiles. For the wall material, brick, plaster and concrete is implemented. The material of the baseboard and window frames is plastic. Glass material was used for the window glass.

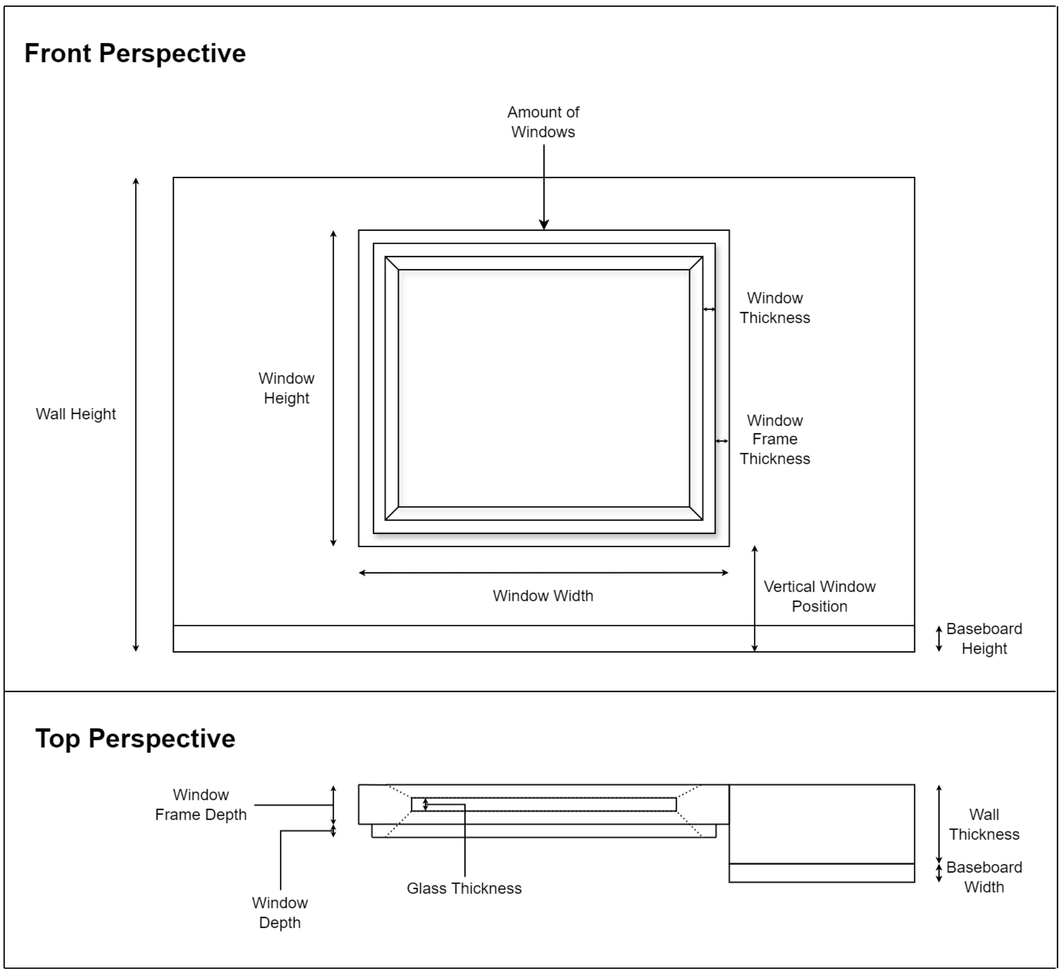


Figure 3.20 Room Attributes

Room Generator

For randomized room layouts, the room construction function [64] of the Python package Blenderproc2 [20], which offers a procedural Blender pipeline for photo-realistic rendering, is implemented. The function creates a room floor layout with the desired area size, the number of extrusions and floor sections. First, a squared floor is created with sixty to eighty percent of the desired area size. Next, the generated area is randomly cut by the number of floor sections. Last, the outer edges are randomly selected and extruded. This step is repeated according to the number of extrusions. The extrusions make up the remaining forty to twenty percent of the desired area size. The described steps and the remaining generating process of the dining room object are visualized in figure 3.21. Next, the walls are created by extruding the outer edges of the floor upward. To create the rectangular baseboard, the outer edges of the floor are copied and extruded. For the windows, holes in the wall are created by removing geometry of the wall that intersects with the bounding box of the windows.

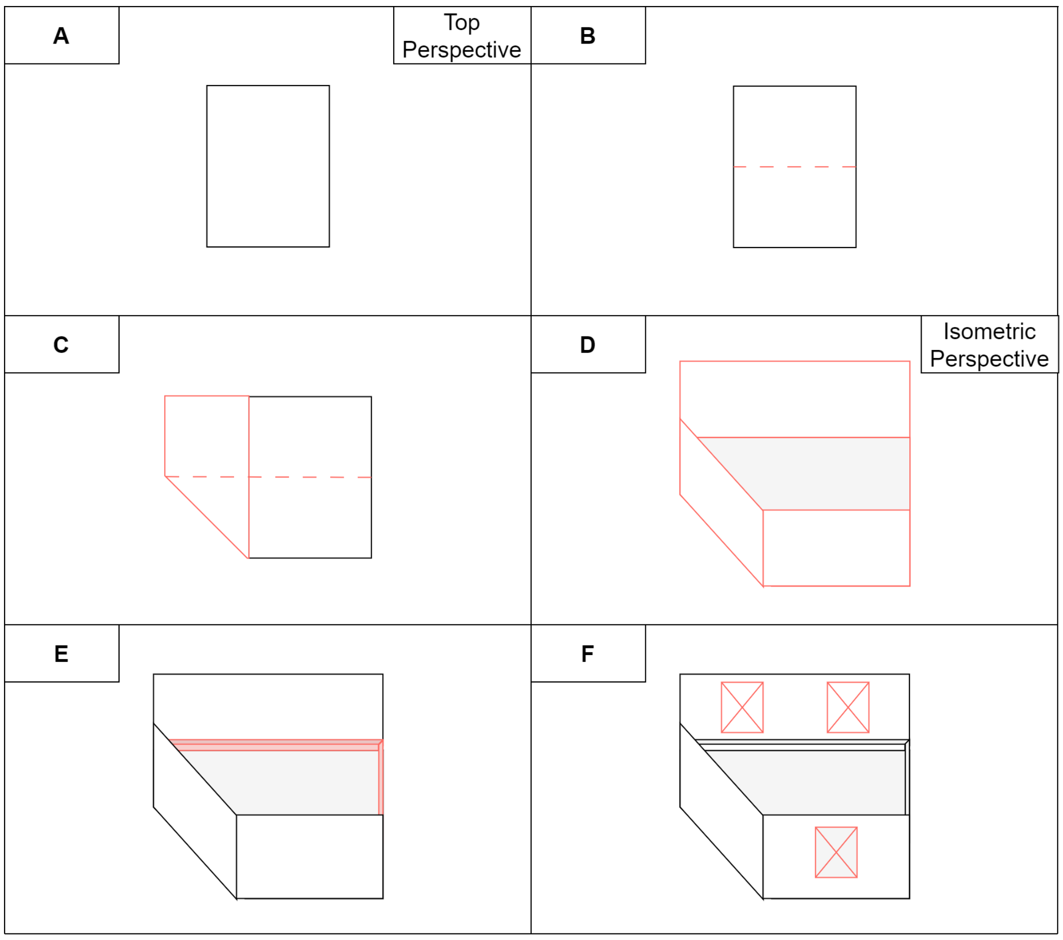


Figure 3.21 Generating Process of the Room Object

The windows are created by generating a cube and removing the back and the front face. The edges are then extruded toward the center of the selected edges to create the window frame. For the window, the edges of the window frame are extruded forward to create thickness. The window edges are then extruded toward the center of the edges and then moved backward to create the window inset. In the last step, the window glass is created by using the edges of the window to create a face.

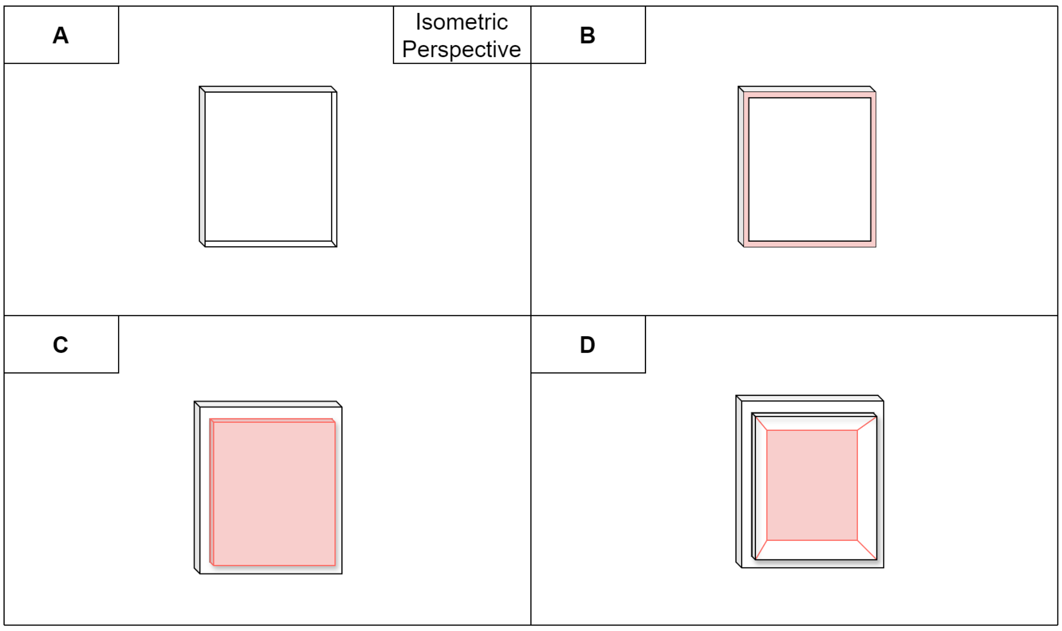


Figure 3.22 Generating Process of the Window Object

3.2.9 Table Distribution

To distribute tableware on the table top, randomly distributed points, which are used to position the tableware, are generated on the table top surface with a minimum distance to each other that corresponds to the maximum radius of all tableware objects. The tableware objects are then generated with random rotation.

To distribute and position the chairs, four points are generated on each side of the table. The distance between the points and the table is random. Like the tableware, the chairs are generated with random rotation, but is limited so that the chair backrests will not intersect with the table. A possible distribution of tableware and chairs can be seen in figure 3.23.

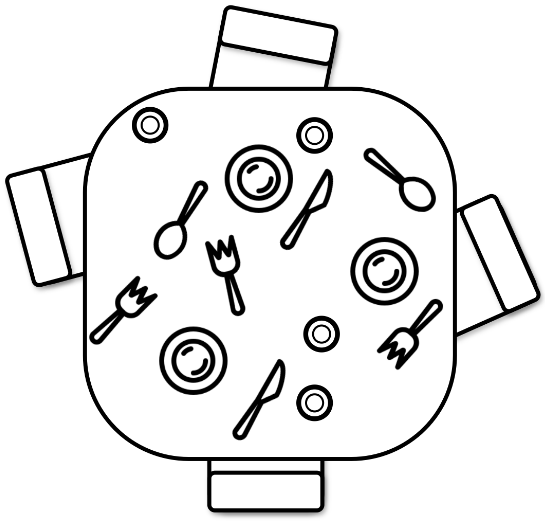
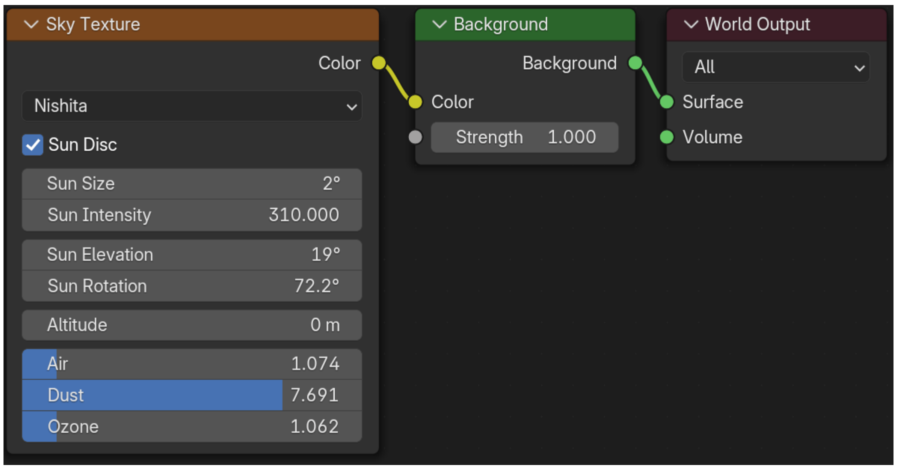


Figure 3.23 Tableware and Chair Distribution

3.2.10 Lighting

The lighting of the scene is separated into outdoor and indoor lighting. For outdoor lighting, a node graph (see figure 3.24) is implemented that utilize the *Sky Texture*-Node, which creates a procedural HDRI that can simulate the sky. The *Sky Texture*-Node can simulate the sun and change attributes such as size, intensity, elevation and rotation to simulate the day and night cycle. In addition, the density of air, dust, water droplets and ozone can be adjusted to simulate different atmosphere conditions such as clear atmosphere, high pollution, haze or city-like atmosphere.

For randomized outdoor lighting, every input of the *Sky Texture*-Node is randomized for each image except the sun size and altitude. To affect the dining scene, windows were generated with randomized measurements and placed randomly on the walls.



Nishita
✓ Sun Disc
Sun Size 2°
Sun Intensity 310.000
Sun Elevation 19°
Sun Rotation 72.2°
Altitude 0 m
Air 1.074
Dust 7.691
Ozone 1.062
Strength 1.000
All

Figure 3.24 Procedural HDRI Node Graph

For indoor lighting, the volume of the room is used to randomly distribute light objects to imitate lamps and ceiling lights. An example of the light object distribution can be seen in figure 3.25. The bounding box of the table and an area around the table called dining room circulation space [22], which is the recommended walking space around the table, are removed from the volume of the room to avoid lights intersecting the dining room objects. In addition, the recommended height of lights above the dining table was taken into consideration for the distribution of lights.

The amount of light objects is calculated considering the recommended lumen per square meter [7], the square meter of the floor area and the light bulb strength [50]. The following formula was used to determine the amount of recommended light bulbs:

$$\text{Amount of Light Bulbs} = \frac{\text{Floor Area } [\ m^2] * \text{Recommended Lumen per } m^2}{\text{Light Bulb Strength [Lumen per } m^2]}$$

Last, the light bulb temperature, which ranges from cold to warm lighting, was chosen at random and corresponds to the recommended range of light bulb temperature for dining rooms [28]. The light bulb temperature can be adjusted in Kelvin and was implemented as node graph.

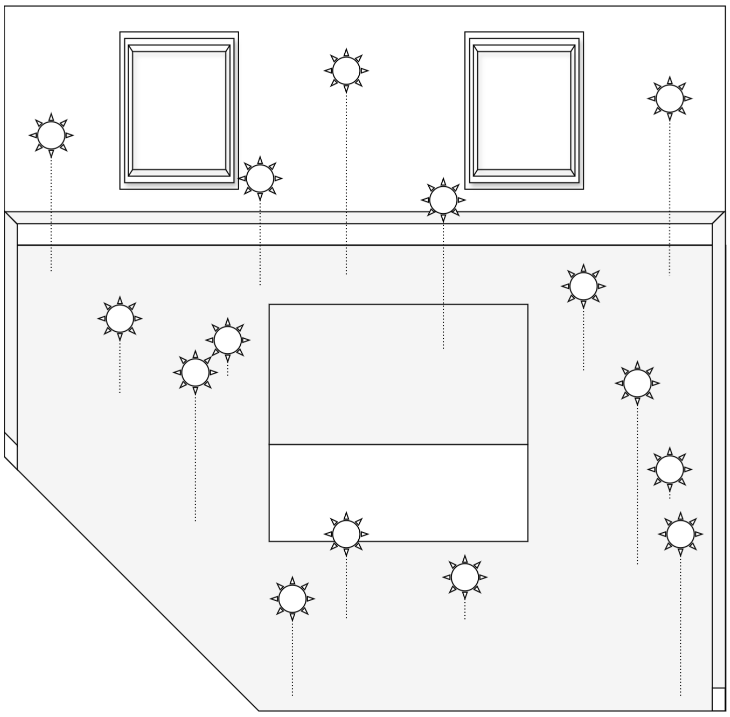


Figure 3.25 Indoor Lighting Distribution

3.2.11 Camera

For camera placement, an area around the table called dining room circulation space [22], which is the recommended walking space around the table, was implemented to generate points that are used for possible positions for the camera. One of the points is randomly selected and used to place the camera in the dining room. The height of the camera is chosen randomly and the height range corresponds to the heights of cameras in robots. To align the camera, the surface of the table top is used to create points, one of which is randomly selected for the camera to focus on. An example of the camera placement is illustrated in figure 3.26.

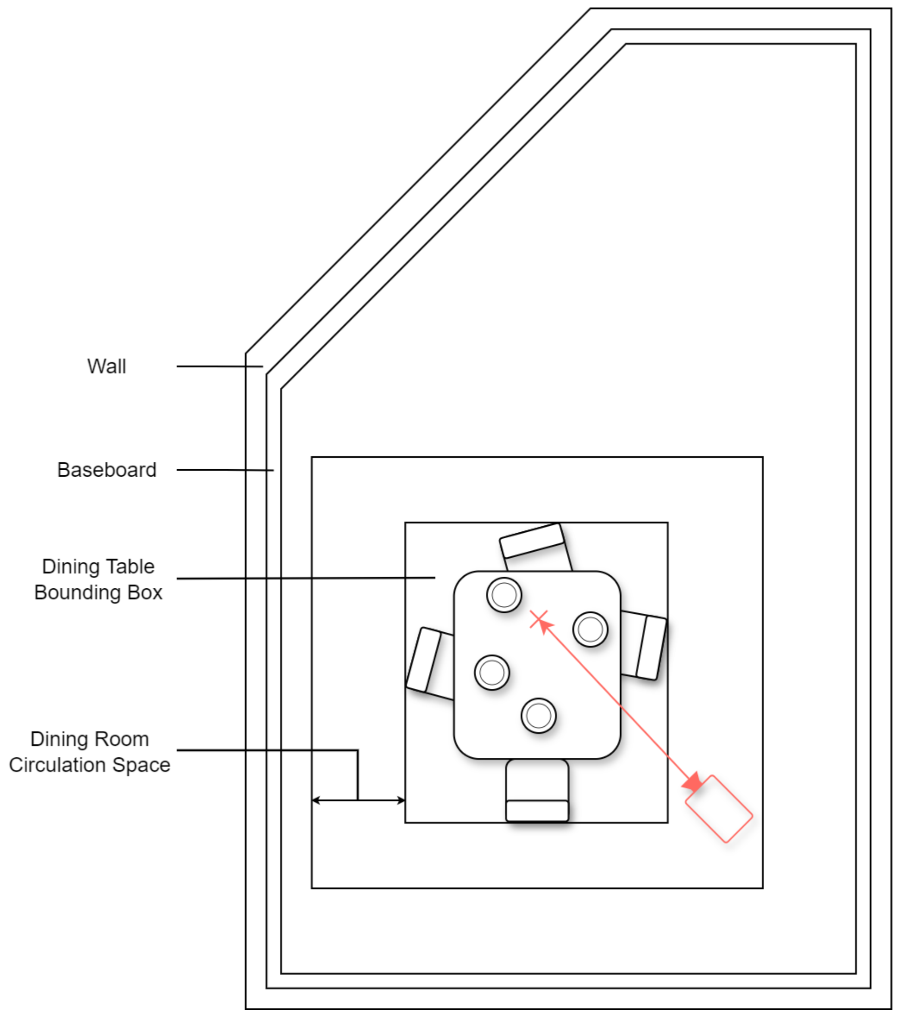


Figure 3.26 Example of Camera Placement

Camera settings such as focal length, camera sensor size, depth of field, exposure and aspect ratio can also be adjusted. The focal length is chosen randomly and ranges from 35mm to 85mm, which matches the focal length of standard camera lenses [25]. The camera sensor size was kept at 36mm, which is the size of full frame sensors and is the most common sensor implemented in consumer model cameras [13]. An aperture of f/4 was selected for the depth of field, which is recommended for indoor photography [82]. For the images, an aspect ratio of 4:3 was chosen, which is common for cameras implemented in robots. For exposure, the Blender add-on named *Photographer 5* [60] was used to automatically adjust the camera exposure to the current lighting of the scene. For the automatic adjustment [61], the addon calculates the exposure based on the average brightness in the scene by analyzing the viewport render, which is a low-resolution render for previewing the scene. Last for realism, the *Photographer 5* add-on was used to add film grain to the images.

3.3 Dining Room Materials

All the materials for the dining room are generated procedurally with *Shader Nodes* and the associated node graphs are created using the approach described in section 3.1.3. It allows the adjustment of the materials such as color, roughness, rotation, scale, surface imperfections, or tile size for materials such as brick or stone tile. Procedurally generated materials also do not require UV maps for the correct projection of material on objects. However, the different characteristics of the materials were empirically created except the IOR values, which correspond to reference values [53, 41, 42] that are found online. For variability, the rotation of the material is randomized. The colors of the materials are also randomly selected. The color of materials, that require one color, are chosen by randomized RGB values. For materials that require multiple colors, color palettes are created that are based on image references that are found online by searching the material name with Google Images.

3.3.1 Tableware Materials

Ceramic

The ceramic material [8] is used for the plates and table tops. The ceramic material can be seen on the left in figure 3.27. The applied IOR value corresponds to a reference value [53]. For the table tops, the color of the ceramic was changed by randomly selecting RGB values. The color of the plates did not change and remained white during data creation.

Glass

The glass material [4, 59] was implemented for the drinking and window glass. To create the glass material, Blender implemented a shader node named *Glass BSDF*. In addition, the node *Volume Absorption*, which absorbs light that passes through the volume and is recommended for colored glass or water [85], was used to create the material. The color of the glass, which is white, did not change in the creation of the data. The glass material can be seen on the left of figure 3.27.

Stainless Steel

The stainless steel material [74] is used for forks, spoons, and knives. The stainless steel material can be seen in the middle of figure 3.27. For fidelity, texture as in brush metal was generated. In addition, small scratches are added as surface imperfections. The color of the tableware did not change and remained gray during data creation.

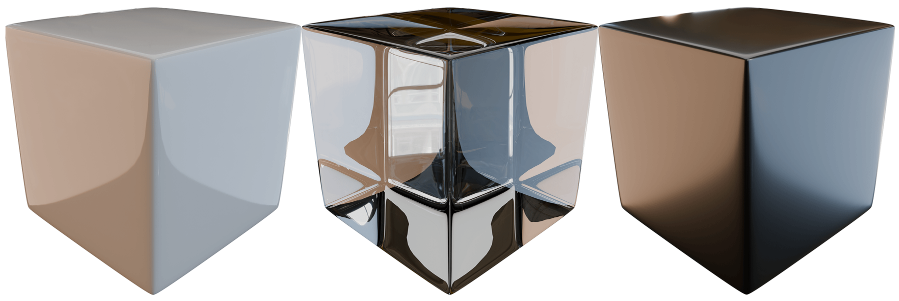


Figure 3.27 The Ceramic, Glass and Stainless Steel Material

3.3.2 Furniture and Floor Materials

Wood and Wood Planks

To create wood material [48], details such as wood grain and wood knots are implemented. In addition, four different types of scratches were added for surface imperfections. Three colors are required for the material, for the base color, the wood grain and the scratches. For the data creation, eight different color palettes, that are randomly chosen for each image and object, were added. In addition, the rotation was randomized for each object. The wood material is used for all components of the table and chair object.

The wood plank material [71], which is used for the table top, chair seat and floor, was created using the same approach as the wood material, with the addition of plank texture that can be adjusted in width and length. Like the wood material, the rotation and color of the wood plank material is randomized in the data creation. The same color palettes are used for this material as for the wood material.

Both wood materials with all color variations can be seen in figure 3.28.

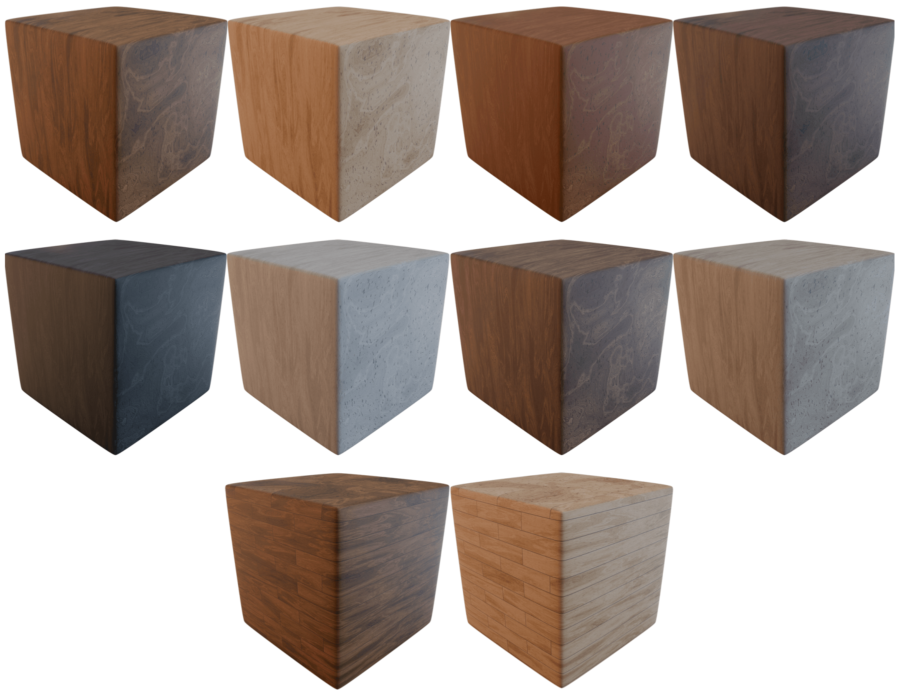


Figure 3.28 The Wood and Wood Plank Material

Marble, Veined Marble and Marble Tile

Three different types of marble material, which differ in their marbling pattern, was implemented for this thesis. The marble and veined marble material is used for the table top. The marble tile material is used for the floor.

The marble material [72] requires two colors, for the base color and the marbling. For surface imperfections, the marbling pattern was used. For variety, four different color palettes were added for this material. The marble material can be seen in figure 3.29.

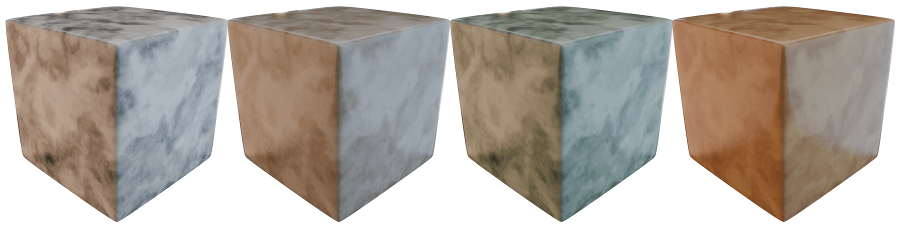


Figure 3.29 The Marble Material

The veined marble material [68] requires three colors for two base colors and the marbling. The veins can be adjusted by the scale, amount of distortion, visibility and detail. For this material, six color palettes were added. The veined marble material with the six color variations can be seen on the the top and middle row in figure 3.30.

The marble tile material uses the veined marble material as a base and implements a tile texture. In addition, the rotation of the material is randomized for the data creation, but the same color palettes are used as for the veined marble material. The marble tile material with one color variation can be seen on the bottom row in figure 3.30.

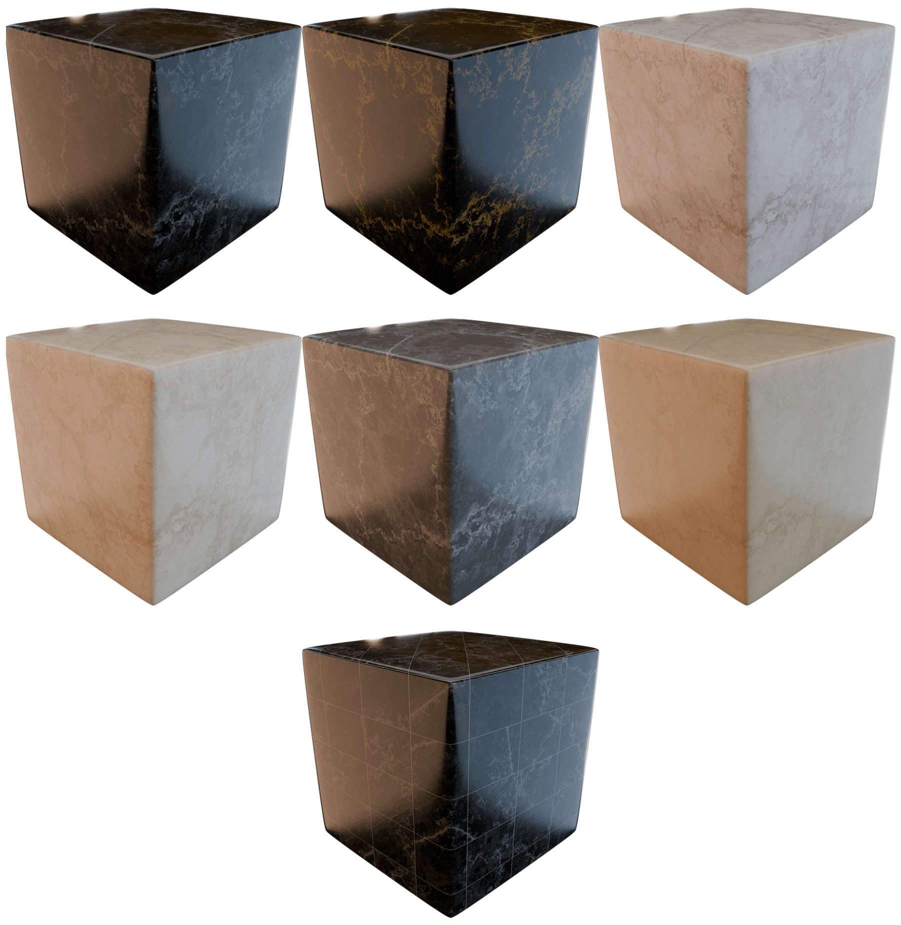


Figure 3.30 The Veined Marble and Marble Tile Material

Plastic

The plastic material [77], which can be seen in figure 3.31, is implemented for table top, chair seat, window, window frame and baseboards. For details, four different types of scratches are added as surface imperfections. The color of the material is randomly selected from RGB values in the data creation. In addition, the rotation of the material is randomized.

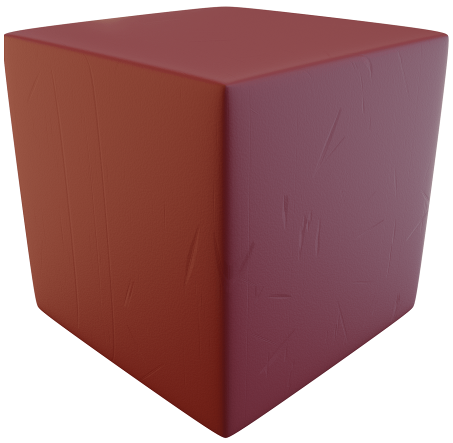


Figure 3.31 The Plastic Material

Metal

The metal material [76] is used for the table apron and table legs, and requires two colors for two base colors. In addition, randomly generated surface imperfections are implemented. For variety, three color palettes were added that can be seen in figure 3.32.

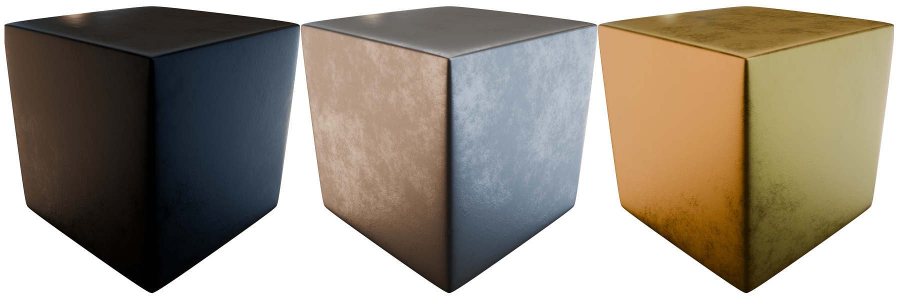


Figure 3.32 The Metal Material

Stone Tile

The stone tile material [79], used for the floor, requires three colors for two base colors and one for details. The tiles can be customized by scale, length and width.

The material was implemented twice. Once with square tiles and once with rectangular tiles. For details, the bricks have an uneven surface generated by random. For data creation, six color palettes were added and the rotation of the material is randomized. The material with the color variations can be seen in figure 3.33.

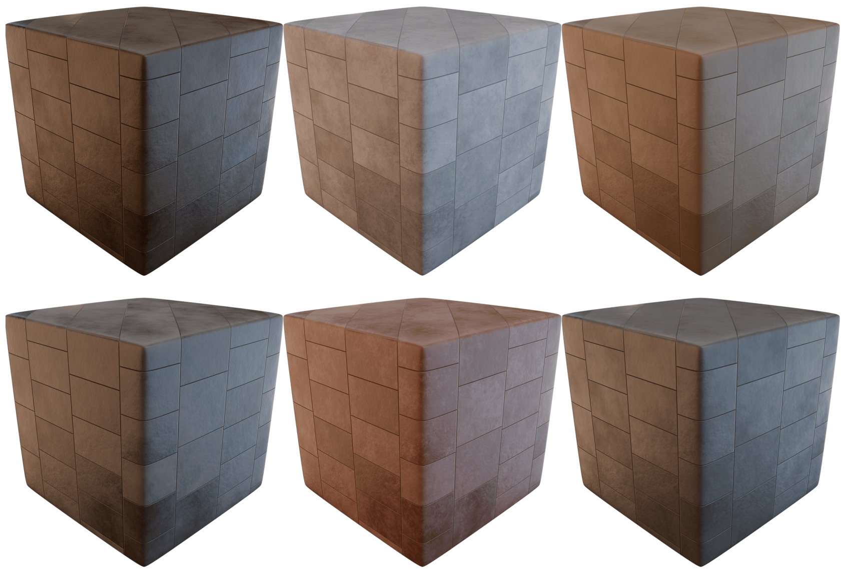


Figure 3.33 The Stone Tile Material

Checkered Marble

The checkered marble material [70] alternates between black and white marble tiles, so four colors are required for the two base colors and the two marbling patterns. For variety in the data creation, six color palettes were added and the rotation of the material is randomized. In addition, the tile scale, tile rotation, marble scale and marble distortion can be adjusted. The material was implemented for the floor and can be seen in figure 3.34.

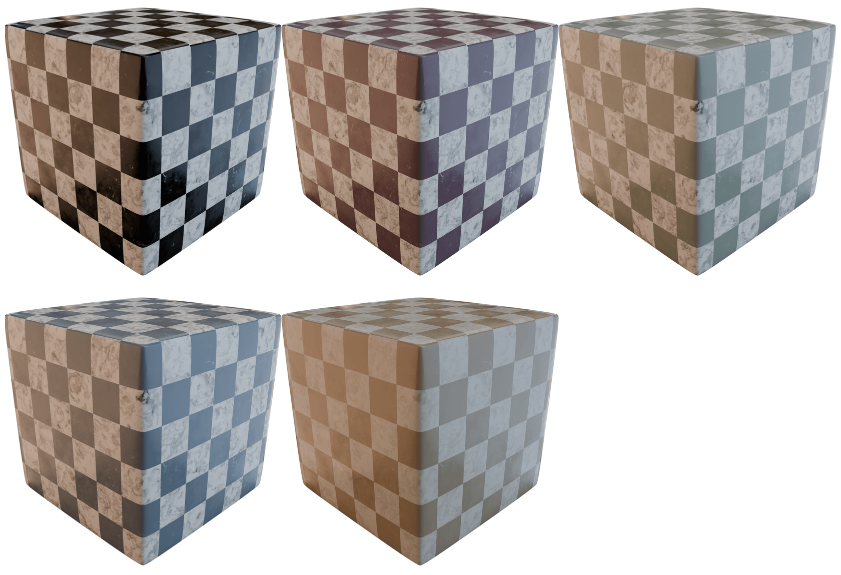


Figure 3.34 The Checkered Tile Material

Carpet

The carpet material [44], which was implemented for the floor, requires three colors for the base color, color variation in the carpet and dirt. In addition, the dirt can be adjusted by scale and amount. For variety, eight color palettes were added and the rotation of the material is randomized. The material and the color variations can be seen in figure 3.35.

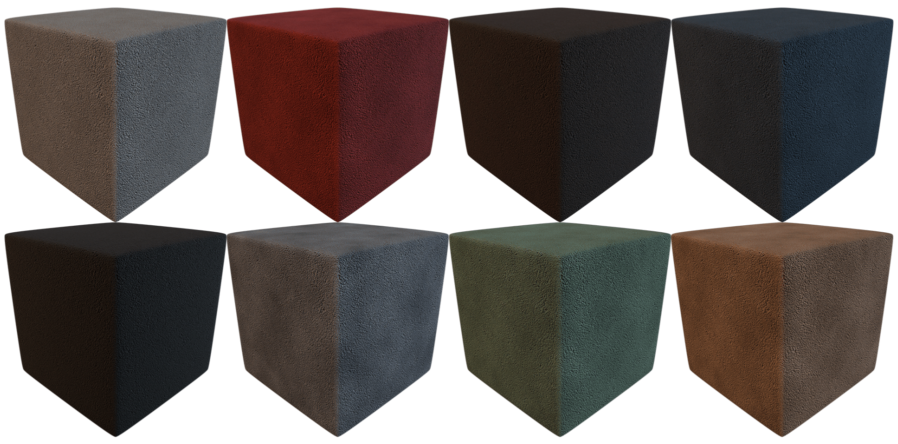


Figure 3.35 The Carpet Material

3.3.3 Wall Materials

Concrete

For the concrete material [78], multiple small imperfections were implemented, such as small bumps, holes, and pebble. The imperfections can be adjusted by intensity and scale. The material requires two colors for two base colors, but the colors were not changed for the data creation and remained two shades of gray. The concrete material can be seen on the left in figure 3.36.

Plaster

The plaster material [75] requires one color for the base color, which is randomly selected by RGB values in the data creation. For details, small bumps were added as surface imperfections. The plaster material can be seen on the right in figure 3.36.

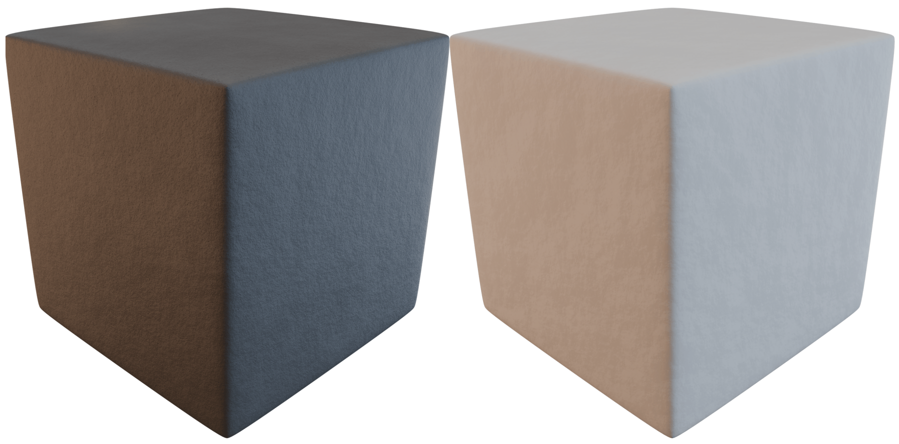


Figure 3.36 The Concrete and Plaster Material

Brick

The brick material [69] was created with the *Brick Texture*-node. The bricks can be customized by scale, width and length. For details, erosion and bumps are added as surface imperfections that can be adjusted by scale and intensity. The material requires four colors for two different bricks, dirt and erosion. Six color palettes were added for variety and can be seen in figure 3.37.

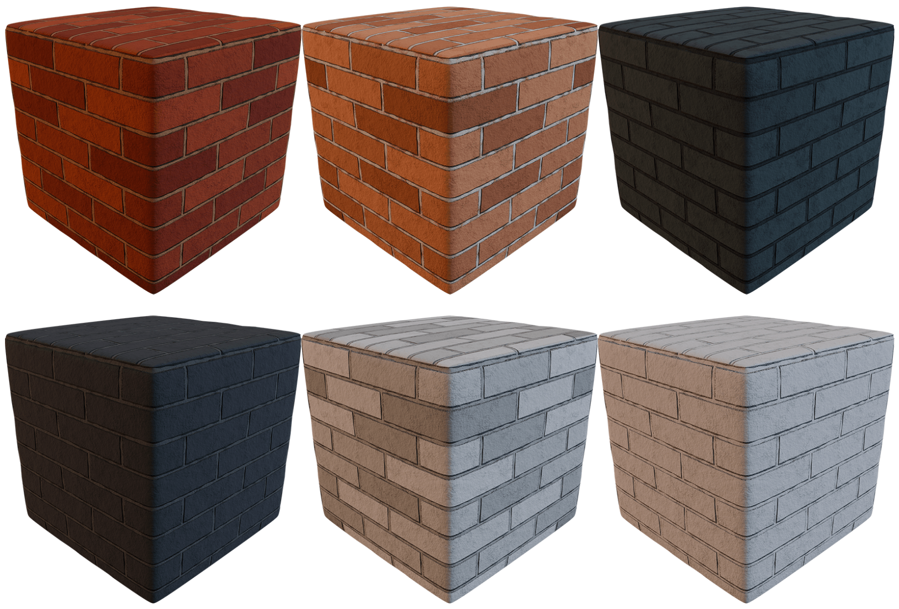


Figure 3.37 The Brick Material

3.4 Food Soil

To implement the soil, the defining characteristics of the soil must be identified. For reference, images of soiled plates were analyzed from the dataset "Cleaned vs Dirty V2 Dataset" [18], online image databases such as Shutterstock, Flickr, Unsplash, iStock, and image results from Google Images. For Google Images and the image databases, the following search terms were used to find soiled tableware images: dirty dish/es, dirty plate/s, dirty tableware, dirty cup/s, dirty cutlery, dirty flatware, food smear/s, food stain/s, food crumbs, food spot/s. Only real high-resolution images, in which soil or soiled tableware is visible and not blurred, were selected as a reference. Examples of the chosen references can be seen in figure 3.38.

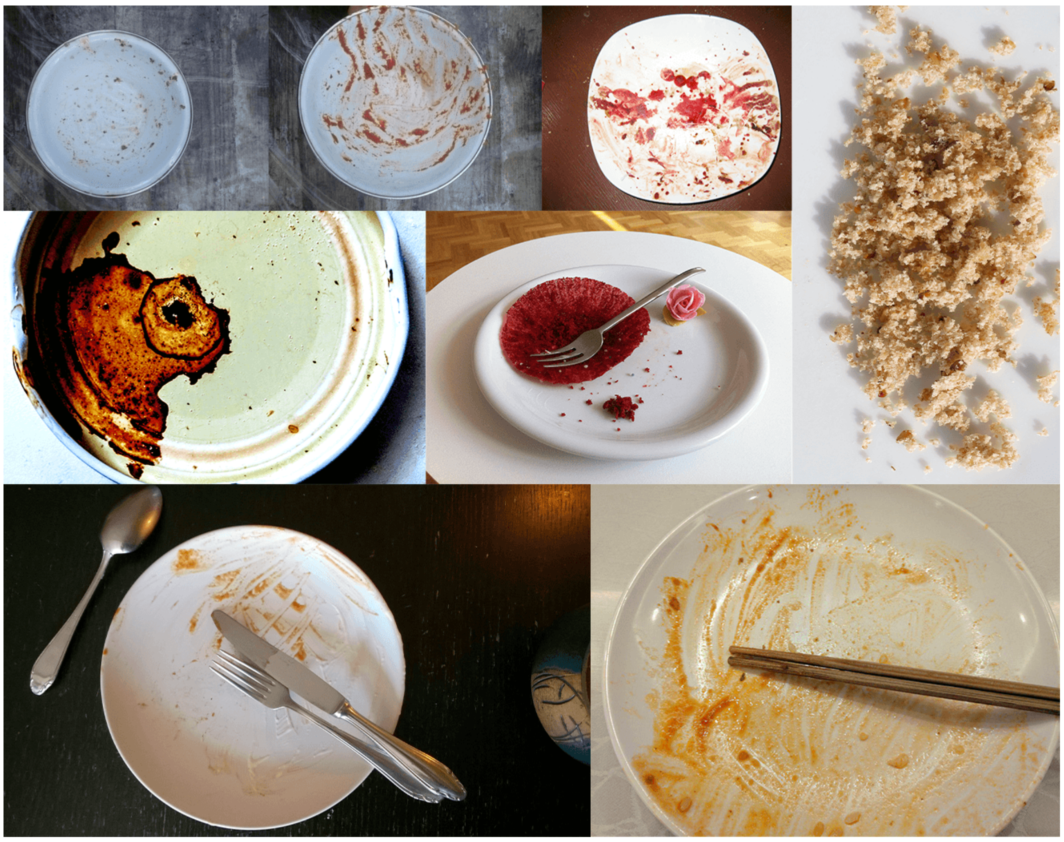


Figure 3.38 Examples of Food Soil

3.4.1 Soil Definition

Based on references (see figure 3.38), this thesis focuses on two aspects of soiled dishes, namely the type of food residue and soil patterns based on the characteristics of food and the way people consume food. The reference images include sauces, food residues, crumbs from baked foods and oil or fat from fried foods. Sauces are also divided into low-viscosity and high-viscosity sauces. Low-viscosity sauces behave more like water, for example, soy sauce, and high-viscosity sauces behave more like honey, for example ketchup. The different types of food produce different soil patterns on the plates. Low-viscosity sauces accumulate and form splatters or droplets. High-viscosity sauces do not accumulate and can form smudges, smears or streaks if the sauce is picked up with flatware during the meal.

Eating baked foods such as bread or cakes produces crumbs, which can also be considered a soil pattern. Food prepared by frying tends to be greasy and results in splatters of oil on the plate. The oil can increase the complexity of tableware detection in terms of reflection. In this thesis, food residues are not considered as they depend on the food and are therefore too broad to be implemented.

3.4.2 Implementation

Splatters, droplets, smears and smudges of low- and high-viscosity sauces and oil spots are implemented with *Shader Nodes*. Crumbs are generated and distributed with *Geometry Nodes*. A procedurally generated pastry material [67] is implemented for the crumb material. Two examples of the soil tableware material can be seen in figure 3.39.

For the different sauces and oil splatters, a combined material [43, 62, 73] was implemented that uses the ceramic material as the base material and adds randomly generated soil. The combined material implements two different node graphs for low- and high-viscosity sauces. Oil spots are added for both types of sauces and are generated on unoccupied surfaces. Both sauce types can be adjusted by the material scale, sauce scale, droplet scale, oil scale, sauce coverage, oil coverage and color. For data creation, a random variable is implemented that randomizes the inputs for both sauce types and also randomizes which sauce type will be generated. In addition, the colors of both types of sauce are chosen by randomized RGB values.

For the crumbs, cubes are generated as base geometry. For the variety in crumb shapes, the geometries of the cubes are changed by randomizing the position of the vertices. The amount of different crumb shapes can be adjusted by the input. The minimum and maximum size of the crumbs can also be adjusted. To distribute and position the crumbs, randomly placed points are created on the top of the plate. In addition, a mask is implemented that defines random areas on the plate, where the crumbs are to be generated, in order to make the crumb placement more varied and to avoid crumbs being generated on the entire surface. The density of the crumbs and the mask shape can be adjusted.

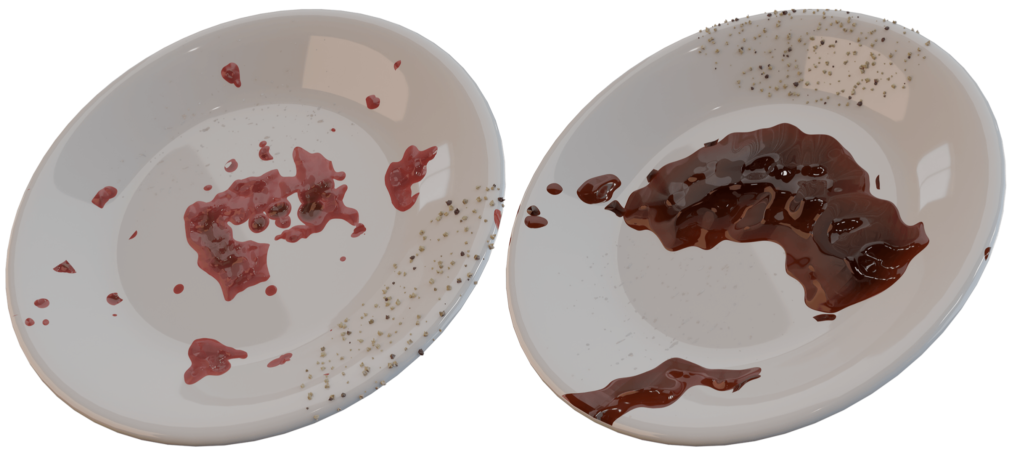


Figure 3.39 The Soiled Tableware Material

3.5 Dataset Generation Pipeline

In the following, the Blender settings, the hardware, the data generation process and the computational cost of the data generation are presented.

3.5.1 Blender Settings

The scene was rendered in the engine named *Cycles* and uses the *OptiX* option in the settings, which activates GPU and CPU usage. For color management, the display device setting was set to *sRGB* and the view transform setting was set to *AgX*.

Each image was rendered with a maximum number of 512 samples and a noise threshold of 0.0001. When rendering images, visual artifacts can occur in the image, which are referred to as noise. To remove noise, the lighting of the scene or the maximum number of samples for rendering can be increased, or the feature named denoiser, which utilizes AI to remove noise, can be used. For this thesis, the denoiser and the option that the denoiser utilizes the GPU for calculations were activated with the settings: *OpenImageDenoise* for the denoiser, *Albedo and Normal* for the passes, *Accurate* for the prefilter and *High* for the quality. In the settings for the light paths, all maximum bounces that influence the light behavior were set to 32. The clamping options were set to 0.0 for direct light and 10.0 for indirect light.

For a second denoising in the compositor, the *Denoising Data*-Option, which is found in the view layer properties under passes, was activated in the scene for the image. To save the ground truth, the *Material Index*-Option, which is also found in the view layer properties under passes, must be activated in the scene for the ground truth.

3.5.2 Data Generation Process

To execute and automate the data generation pipeline, a Python script is implemented. In addition, the script contains a custom user interface for Blender that allows users to use the functions that are executed in the data generation pipeline.

Before starting the pipeline, the shading mode of the 3D viewport, which is used to interact with the 3D scene, must be set to *Rendered*. The mode renders the 3D viewport to provide a low-resolution preview of the final result. Enabling the *Rendered*-shading mode is a requirement for the automatic adjustment of exposure of the Blender add-on *Photographer 5*.

The data generation process starts by randomizing the scene settings, the objects and their materials, presented in section 3.2 and section 3.3, with the corresponding value ranges of the individual features and inputs. The process is then paused for twenty seconds, so that the low-resolution preview of the final image can be rendered for the automatic exposure adjustment. The preview was rendered with a maximum number of fifty samples and a noise threshold of 0.1. In addition, the denoiser activates after twenty samples. After the twenty seconds time limit, the rendering of the image of the dining room is started. If the rendering of the image is finished, the corresponding ground truth is rendered with five samples.

Afterward, the image and the ground truth are named with a random seed number and saved under a specific path using the compositor (see section 3.1.4). Last, all values of the randomized attributes and random seed values of the dining room scene are saved as an entry in a CSV-file.

3.5.3 Computational Cost of the Dataset Creation

An NVIDIA GeForce RTX 2060 GPU and an AMD Ryzen 7 2700X eight-core processor CPU were used to render the images.

For this thesis, 2201 images with clean tableware were generated. The median rendering time was approximately four minutes and 38 seconds. It took about 198 hours and 18 minutes to render all the images with clean tableware.

2201 images with soiled tableware were generated and the median rendering time was approximately five minutes and 33 seconds. The total rendering time for all images with soiled tableware were 225 hours and 50 minutes.

The ground truth for the images with clean and soiled tableware took about 10 seconds to generate.

4. Tableware Datasets

For the training of the artificial neural network (ANN), two datasets that contains images of clean and soiled tableware were created. The two datasets are presented in the following chapter to provide an insight into them.

For the clean tableware dataset, 2201 images with clean tableware were created with the corresponding ground truth images, which are binary masks of the plate objects. For the soiled tableware dataset, 2201 images with soiled tableware were created with the corresponding ground truth images, which are also binary masks of the plate objects, but the soil is included in ground truth labels. For both datasets, images were rendered with the aspect ratio of 4:3 and the image size of 1920 x 1440. Examples of the images of the clean and soiled tableware datasets can be seen in figure 4.1.

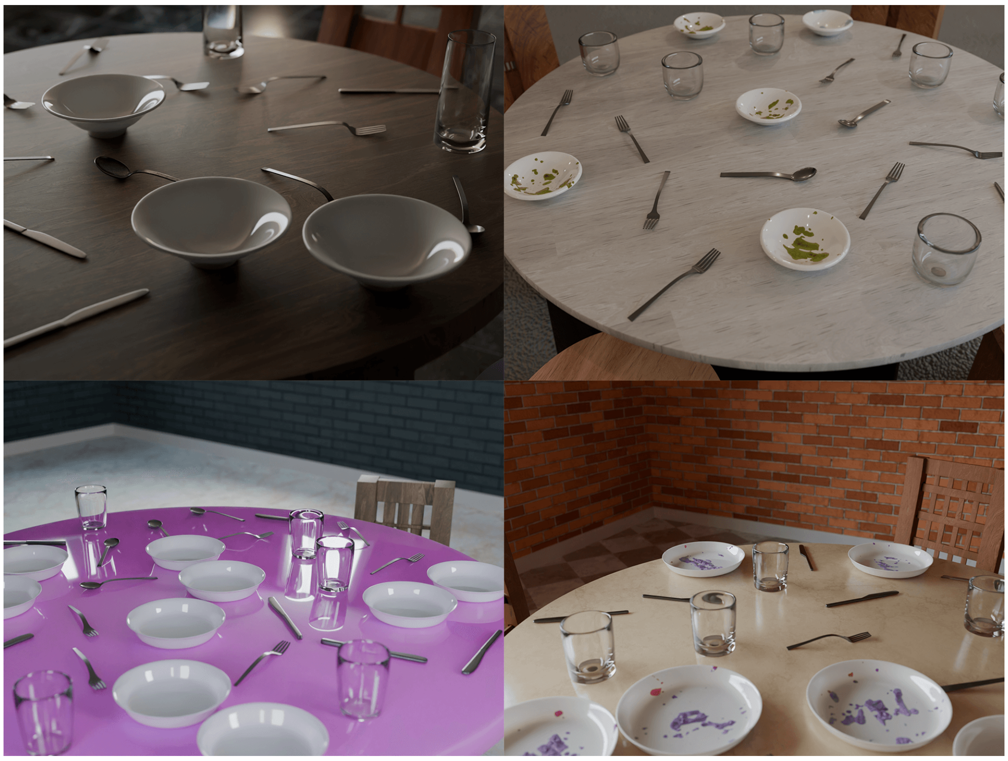


Figure 4.1 Examples of the Generated Tableware Images

To visualize the variety in the image data of both datasets, the principled components analysis (PCA) was performed on the continuous attributes of the objects of the dining room scene for each dataset. The analysis did not consider eighteen discrete object attributes and included 135 continuous object attributes to create two principal components. The PCAs for the clean and soiled tableware datasets are visualized in figure 4.2.

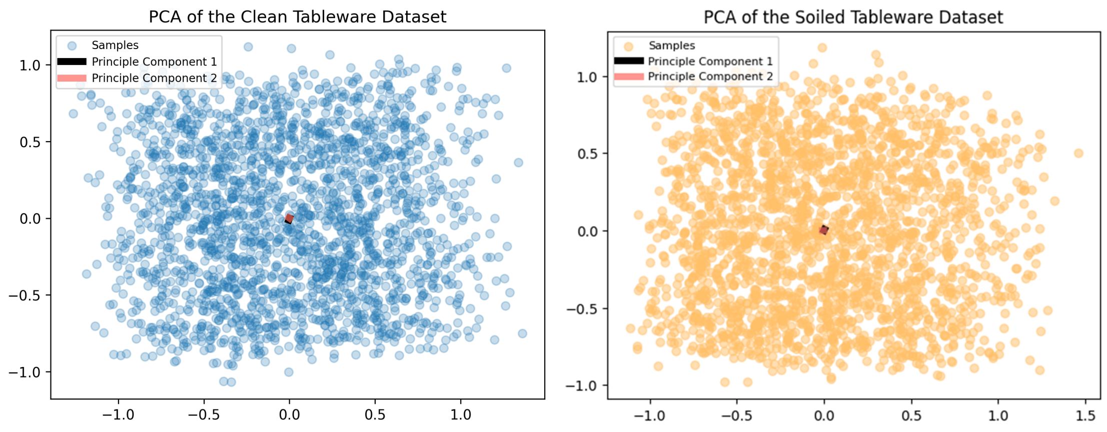


Figure 4.2 PCA of Cleaned and Soiled Dataset

Heatmaps were created to visualize the pixel distribution of ground truth labels in both datasets by combining all ground truth images into one image. The heatmaps for both datasets can be seen in figure 4.3.

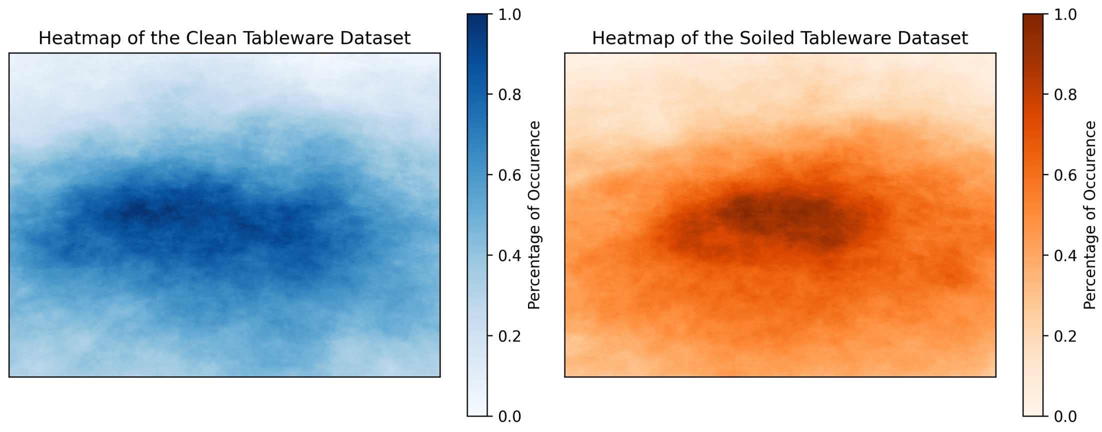


Figure 4.3 Heatmaps of the Ground Truth Images

5. Convolutional Neural Network

A convolutional neural network (CNN) model was implemented to be trained on the clean and soiled tableware dataset to analyze the effects of soiled-based occlusion on tableware detection models. In the following chapter, the architecture of the model, the training of the model and the results of the training are presented.

5.1 Architecture

The CNN, implemented for this thesis, is based on the encoder-decoder architecture. The encoder reduces the amount of input data, in this case the image resolution, and extracts semantic information that is used to solve the given problem, which is dependent on the input data. In the case of this thesis, the encoder could extract characteristics of a plate such as material, color or shape to identify and detect plates in the images. The output of the pixel-wise detection of plates are low-resolution heatmaps. The decoder utilizes the extracted semantic information of the encoder and reconstructs the heatmaps in the original size of the input data.

Due to the complexity of image data, transfer learning is used to provide the implemented model with a general understanding of images by using the MobileNetV2 model, that is pre-trained on ImageNet, as backbone for the encoder. The last three layers of the MobileNetV2 model have been removed and replaced by a head that uses the sigmoid activation function in the output layer, since in this thesis the detection of plates in images is a pixel-wise binary classification task. The encoder outputs whether a pixel is part of a plate or not. The decoder was designed to output the heatmap in the original image resolution of the input. In addition, the architecture implements shortcut connections between the encoder and decoder. The shortcut connections, which can be seen in figure 5.1 as red arrows, provide the image data in the different image resolutions from the encoder to the decoder to improve the reconstruction of the heatmap in higher resolutions without adding parameters or computational complexity. The complete architecture is visualized in figure 5.1.

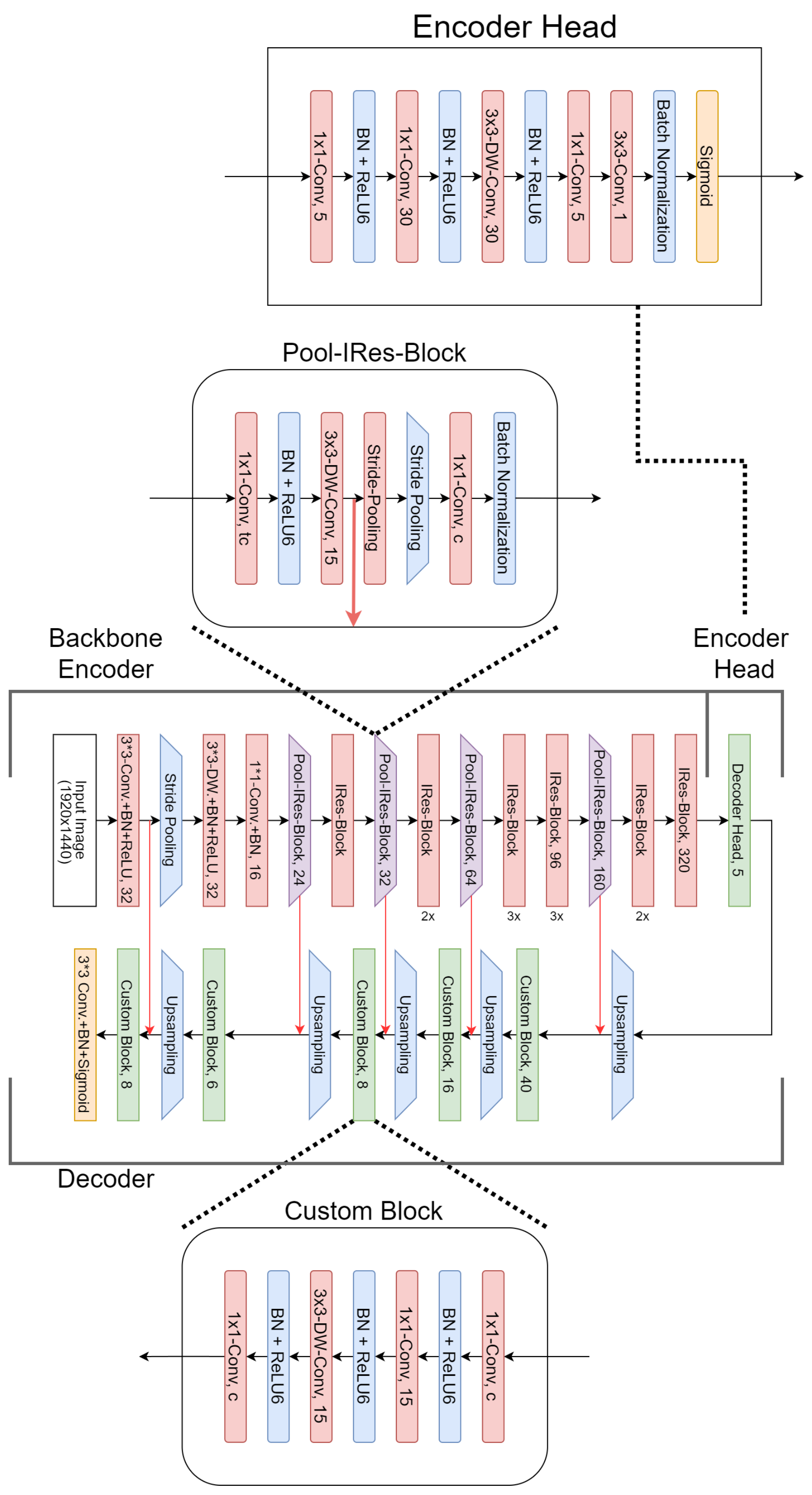


Figure 5.1 The CNN Architecture

In the design process of the encoder head and decoder, the rule of thumb named “One in Ten Rule” [34, 58] was taken into consideration. The rule can be used to estimate the number of parameters a model should have based on the training data to reduce the likelihood of overfitting. It states that there should be ten examples of each output for each model parameter.

In case of the tableware detection task, the number of pixels that are labeled as tableware in the ground truth images must be taken into consideration. Pixels that are not labeled as tableware can be omitted, as more pixels are used to display the background of the scene than the tableware. The image resolution must also be considered in the calculation, as a lower resolution of the image reduces the number of pixels and therefore reduces the number of pixels that are labeled as tableware in the ground truth images. The encoder reduces the image resolution five times because of the five stride pooling layers. The number of pixels is reduced by a factor of four for each pooling layer. The maximum number of parameters are calculated with the clean and soiled tableware datasets, which can be seen in figure 5.2. The calculation shows that the number of parameters of the encoder head and the decoder are below the recommended maximum number of parameters for each image resolution.

$$\begin{aligned}
Parameters_{Encoder\_Head} &= 2418 \\
Parameters_{Decoder} &= 32780
\end{aligned}$$

$$\begin{aligned}
Maximum\_Parameters_{clean} &= \frac{Amount\_of\_Ground\_Truth\_Pixels_{clean}}{10*(4*Resolution\_Level)} \\
&= \frac{3.974.448.591}{10*(4*0)} \\
&\cong 397.444.859
\end{aligned}$$

$$\begin{aligned}
2nd\_Resolution\_Level_{clean} &= \frac{Maximum\_Parameters_{clean}}{4} &\cong 99.361.215 \\
3rd\_Resolution\_Level_{clean} &= \frac{Maximum\_Parameters_{clean}}{16} &\cong 24.840.304 \\
4th\_Resolution\_Level_{clean} &= \frac{Maximum\_Parameters_{clean}}{64} &\cong 6.210.076 \\
5th\_Resolution\_Level_{clean} &= \frac{Maximum\_Parameters_{clean}}{256} &\cong 1.552.519
\end{aligned}$$

$$\begin{aligned}
Maximum\_Parameters_{soiled} &= \frac{Amount\_of\_Ground\_Truth\_Pixels_{soiled}}{10} \\
&= \frac{1.063.785.005}{10} \\
&\cong 106.378.501
\end{aligned}$$

$$\begin{aligned}
2nd\_Resolution\_Level_{soiled} &= \frac{Maximum\_Parameters_{soiled}}{4} &\cong 26.594.625 \\
3rd\_Resolution\_Level_{soiled} &= \frac{Maximum\_Parameters_{soiled}}{16} &\cong 6.648.656 \\
4th\_Resolution\_Level_{soiled} &= \frac{Maximum\_Parameters_{soiled}}{64} &\cong 1.662.164 \\
5th\_Resolution\_Level_{soiled} &= \frac{Maximum\_Parameters_{soiled}}{256} &\cong 415.541
\end{aligned}$$

Figure 5.2 Maximum Model Parameters Calculation

5.2 Training Setup

The CNN was trained twice, once with images of clean tableware and once with the images of clean and soiled tableware. The model was trained on Google Collab and an NVIDIA Tesla T4 GPU was used.

For the first training of the model, all 2201 generated images of the clean tableware dataset were used. In the second training of the model, the clean and soiled tableware dataset were combined and 2201 images were randomly selected. The mixed dataset contained 1107 images with clean tableware and 1094 images with soiled tableware. The datasets for both trainings were divided into 80% for the training set, 10% for validation set and 10% for the test set. Before the training of the model, the images were scaled, so that the pixel values of the images range from minus one to one, because this is a prerequisite for using the MobileNetV2 architecture.

The model was trained in four steps with each tableware dataset. First, the encoder head was trained. In the second step, the encoder head and the backbone will be trained together. The encoder was trained with ground truth images that were downscaled to fit the output size. It outputs heatmaps with 1/32 of the image resolution of the original ground truth image resolution. Next, the decoder will be trained on the images with the original image resolution. After the training of the decoder is finished, the backbone, the encoder head and the decoder will be trained together. In every step, the model was trained with fifty epochs. The metrics precision, recall and Intersection over Union (IoU) was used for the evaluation. The Adam algorithm was chosen as optimizer and the Binary Cross Entropy was used as the loss, because the task is a binary classification problem.

5.3 Training Results

In the following the loss and metrics of the model is presented after training the model with all layers and with the clean and soiled tableware dataset. The model that were trained on the clean tableware dataset will be called “clean model” and the model that were trained on the soiled tableware dataset will be called “soiled model”. All loss and metric values are rounded to the third decimal place.

5.3.1 Loss

The Binary Cross Entropy Loss was used in the training of the CNN. In figure 5.3, the loss curve of the model can be seen. The left graph shows the loss curve after training the model on the clean tableware dataset and the right graph shows the loss curve after training the model on the soiled tableware dataset.

After training the model with fifty epochs on both datasets, the “clean” model achieved a training loss of 0.017 and a validation loss of 0.067. The lowest training loss was achieved in the 43rd epoch with a value of 0.004 and a validation loss of 0.174. In the 22nd epoch, the model achieved the lowest validation loss of 0.031 and a training loss of 0.005.

The “soiled” model achieved a training loss of 0.006 and a validation loss of 0.307 after fifty epochs. In the 49th epoch, the lowest training loss was achieved with a value of 0.003 and a validation loss of 0.168. The lowest validation loss was achieved in the 31st epoch with the value of around 0.307 and a loss of around 0.005.

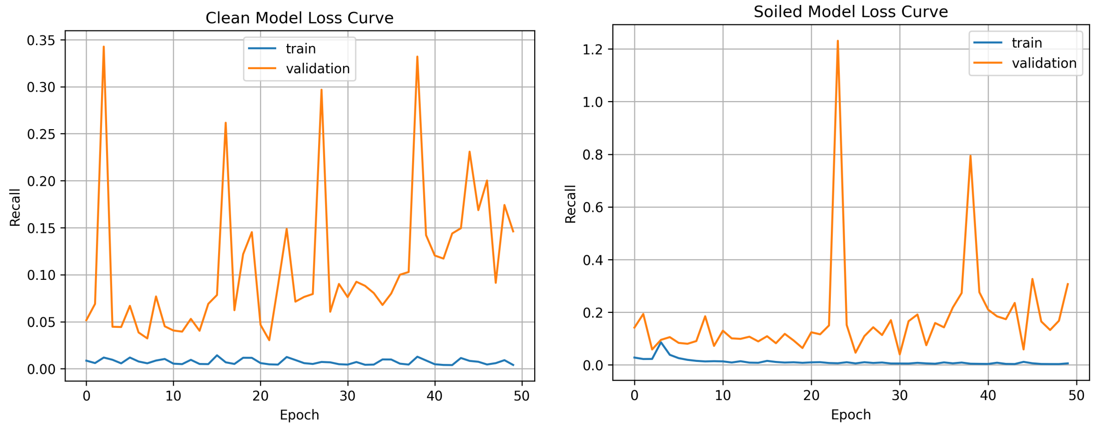


Figure 5.3 Loss Curves of the Model

5.3.2 Metrics

The figure 5.4 shows the Binary Insection-over-Union (IoU) metric curve of the model after being trained on the clean and soiled tableware dataset. On the left of the figure is the IoU curve of the “clean” model and on the right of the figure is the IoU curve of the “soiled” model.

The “clean” model achieved an IoU value of 0.691, which is the highest achieved value, in the training and a validation IoU of 0.5 after being trained for fifty epochs. The highest validation IoU value was achieved in the 22nd epoch with 0.7717 and an IoU value of 0.6738 in the training.

After fifty epochs, the “soiled” model achieved an IoU value of 0.663 in training and a validation IoU value of 0.271. The highest IoU value was achieved in the 49th epoch with 0.756 and a validation IoU value of 0.5. The highest validation IoU value was achieved in the 31st epoch with 0.693 and a IoU value of 0.657 in training.

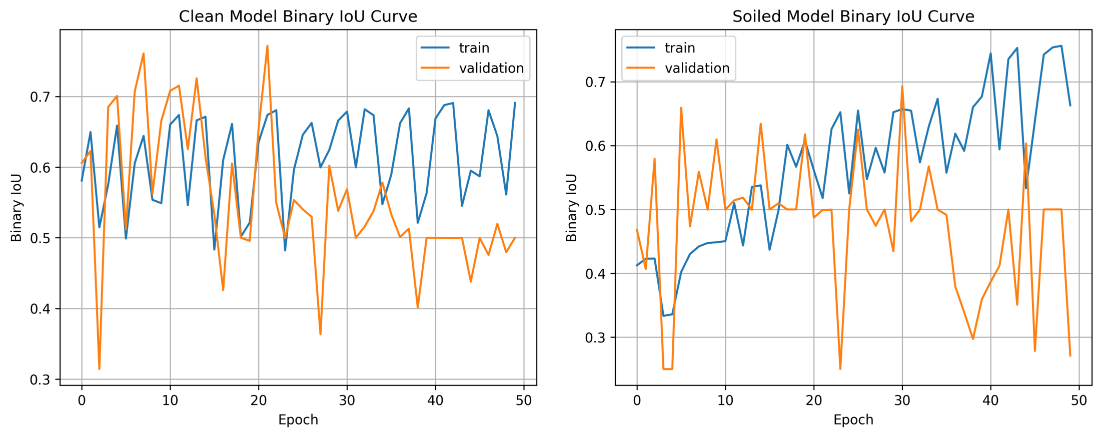


Figure 5.4 Binary IoU Curves of the Model

In figure 5.5, the precision curve of the “clean” model (left) and the “soiled” model (right) was visualized.

After fifty epochs, the “clean” model achieved a precision value of 0.9949 in training and a validation precision value of one. For the “clean” model, the highest precision value in training was achieved in the 43rd epoch with 0.9952 and a validation precision value of 0.999. The highest validation precision value of the “clean” model was achieved in the third epoch with one and a training precision value of 0.984.

In the final epoch, the “soiled” model achieved a precision value of 0.992 in training and a validation precision value of one. In the 49th epoch, the highest precision value was achieved with 0.995 in training and a validation precision value of one. The highest validation value was achieved in the 38th epoch with one and a precision value of 0.987.

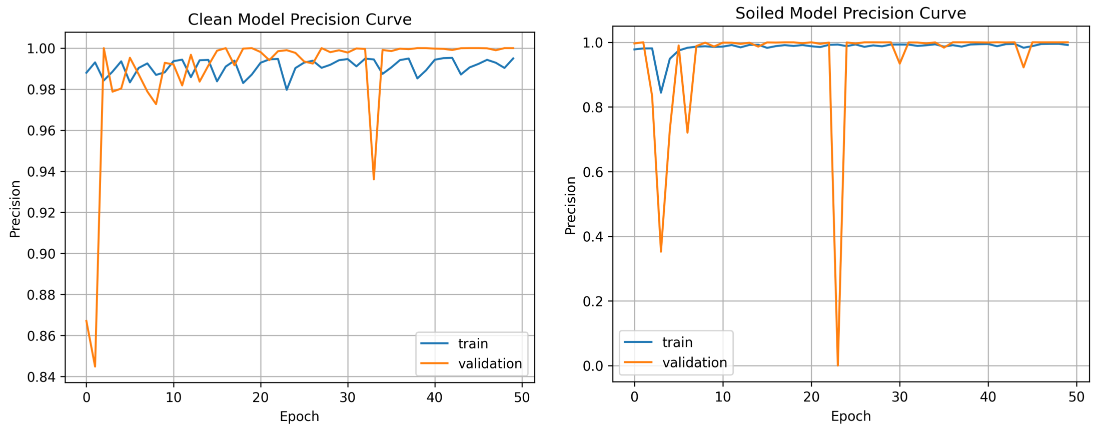


Figure 5.5 Precision Curves of the Model

The figure 5.6 shows the recall curve of the “clean” model (left) and the “soiled” model (right).

In the last epoch, the “clean” model achieved a recall value of 0.982, which is the highest value, in training and a validation recall value of 0.406. The highest validation recall value was achieved in the second epoch with 0.948 and a recall value of 0.9795.

The “soiled” model achieved a recall value of 0.983 in training and a validation recall value of 0.018 in the last epoch. The highest recall value was achieved in the 49th epoch with 0.9886 and a validation recall value of 0.5223. For validation, the highest recall value was achieved in the sixth epoch with 0.8934 and a recall value of 0.9359 in training.

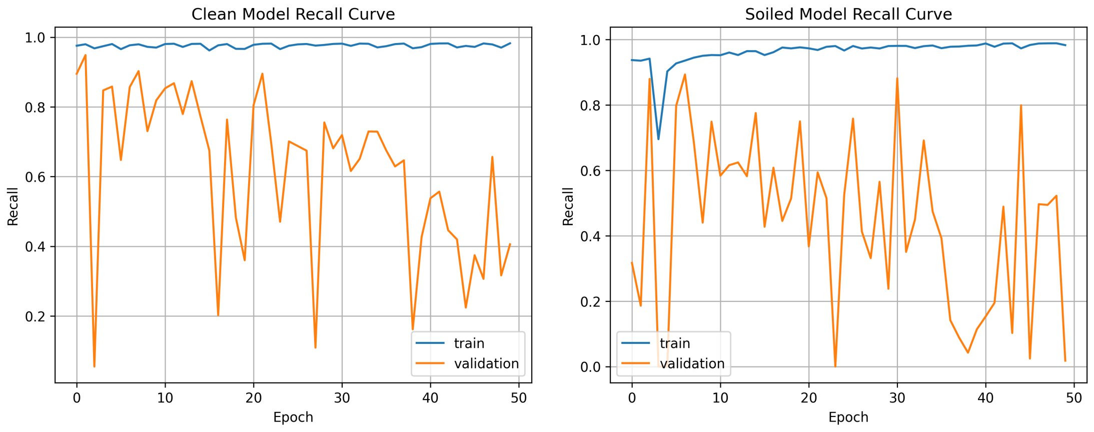


Figure 5.6 Recall Curves of the Model

6. Evaluation

Considering the training results in section 5.3, the evaluation of the model will be performed with the model weights after the 22nd epoch for the “clean” model and the 31st epoch for the “soiled” model, because the model is overfitting after the chosen epochs. In addition, the model with the weights of the chosen epoch achieved the lowest validation values for the loss and the IoU metric. The decision was made, because the evaluation will be performed on the test set of each dataset, which means the model should be able to generalize as good as possible on new data to analyze the effects of soiled-based occlusion in tableware detectors. In the following chapter, the model will be evaluated on the test data and the results will be presented. In addition, a qualitative analysis will be performed by analyzing the segmented images of the test set.

6.1 Quantitative Analysis

For the quantitative analysis, the model performance was evaluated with the test set of 221 images for both approaches. The results can be seen in table 6.1. It shows the loss and the Binary IoU, Precision and Recall of the model for each evaluation approach. First, the “clean” model is evaluated on images of clean tableware. The results can be seen in the first row of the table. Next, the “clean” model is evaluated on images of soiled tableware. The results can be seen in the second row of the table.

The “soiled” model were evaluated twice on two different test sets to examine if the performance of the “soiled” model is dependent on the test set. Each test set contained a random mixture of 221 images of clean and soiled tableware. The results can be seen in the third and fourth row of the table.

Evaluating the “clean” model on images with soiled tableware worsened the loss and the values of the metrics. Evaluating the “soiled” model on two separate test sets with images of clean and soiled tableware, did not change the loss or the values of the metrics.

Evaluation Approach	Loss	Binary_IoU	Precision	Recall
Clean Model + Clean Test Images	0.066624	0.598130	0.999084	0.741356
Clean Model + Soiled Test Images	0.229499	0.456875	0.997014	0.298518
Soiled Model + Soiled Test Images	0.171003	0.422707	0.999970	0.230109
Soiled Model + 2nd Soiled Test Images	0.176132	0.449973	0.999878	0.261309

Table 6.1 Loss and Metric of Model Evaluation

6.2 Qualitative Analysis

For the qualitative analysis, the model predicted heatmaps for the images in the test sets for each evaluation approach. In the following, observations of the generated heat maps of the model will be presented.

The “clean” model is able to detect clean tableware in the images of the test set. It is able to detect plates and bowls, which can be seen in the upper row of figure 6.1.

In addition, the model can detect plates that are partially visible near the border of the image, which can be seen in the bottom row of figure 6.1. The "clean" model tends to misclassify reflections, which can be seen in figure 6.2. Reflections on the plate can cause the part of the plate to be misclassified as not being part of a plate and reflections of the plate on background objects can cause the part of the background object to be misclassified as plate. Furthermore, the model tends to misclassify large plates or plates that are near the camera (see figure 6.3).

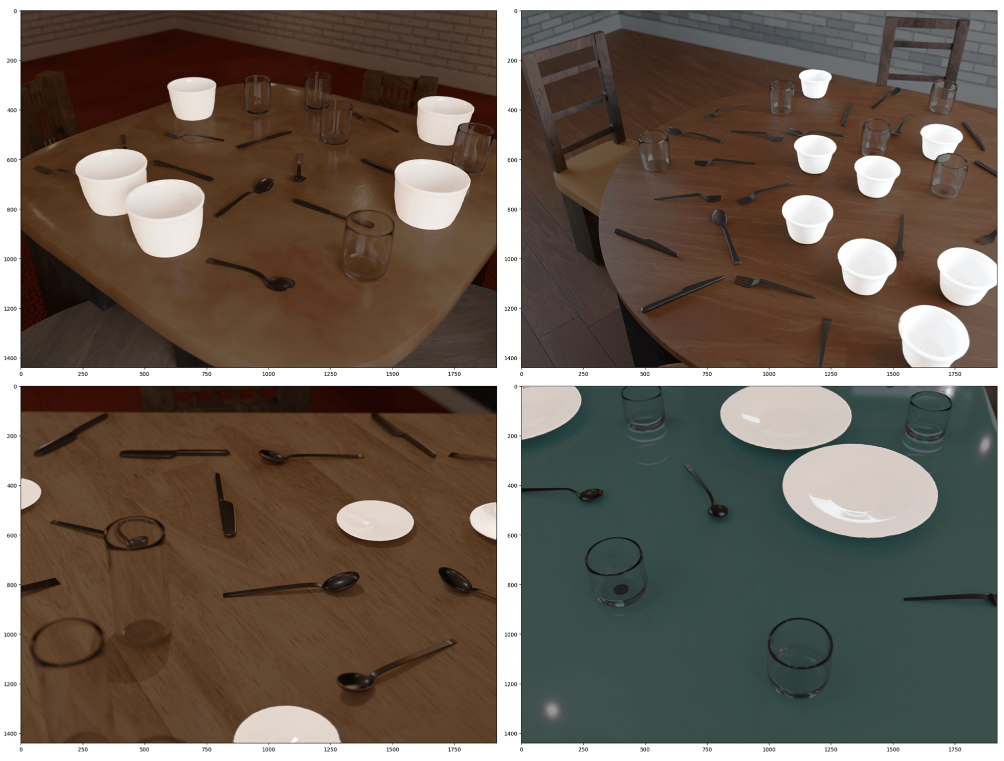


Figure 6.1 Clean Model: Detection of Clean Plates

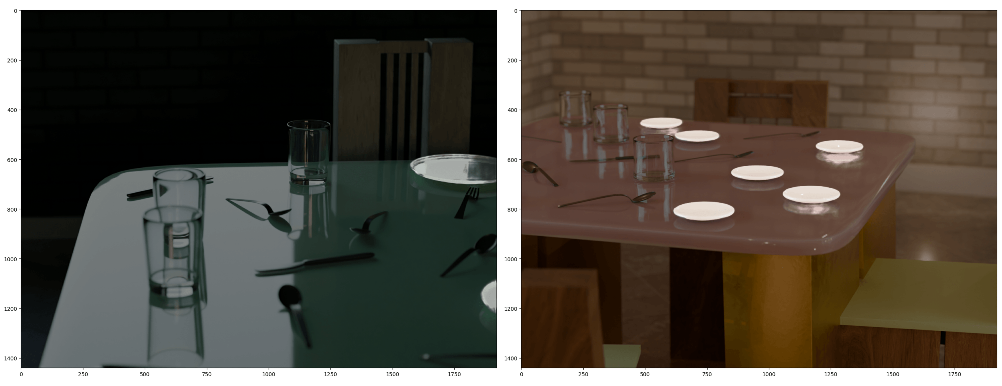


Figure 6.2 Clean Model: Misclassification of Reflections

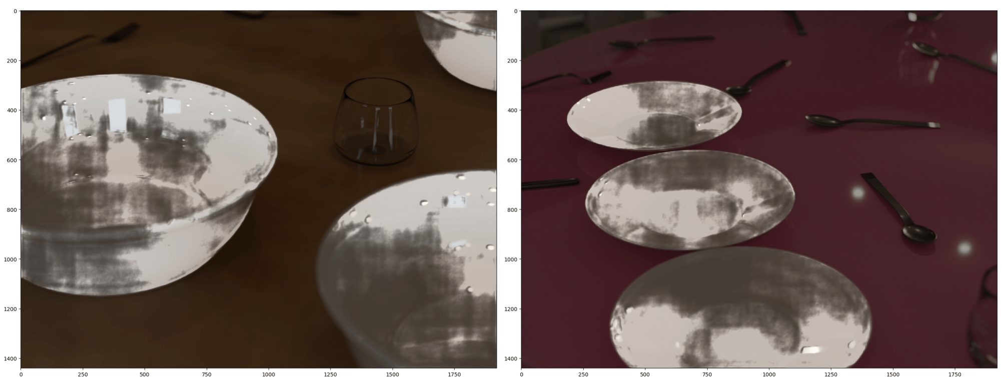


Figure 6.3 Clean Model: Misclassification of Large Plates

The evaluation of the "clean" model with images of soiled tableware resulted in the model not recognizing some plates in the images. Either all plates were not detected in the image or some plates were not detected, while other plates were detected in the same image. Some examples of the described case can be seen in figure 6.4.

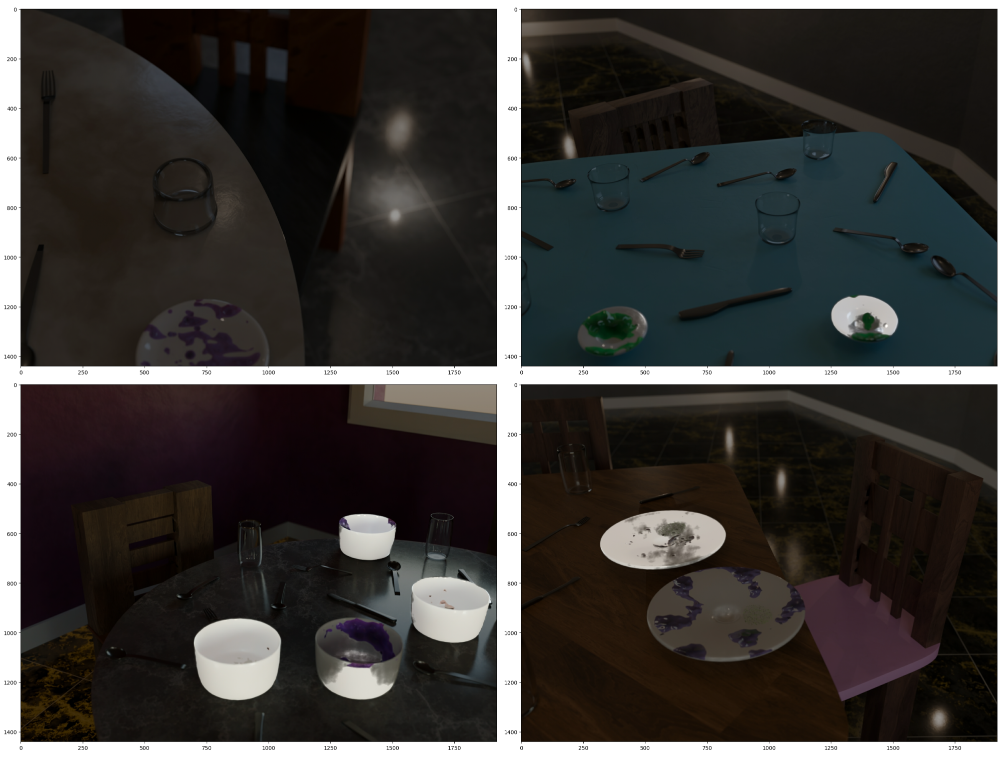


Figure 6.4 Clean Model: Undetected Plates

In both evaluation approaches of the “soiled” model, the model were able to detect clean plates, which can be seen in figure 6.5. It was able to identify occlusions from glasses as not being part of plates (see figure 6.6). Furthermore, soil were labeled as part of plates. Some examples of this case can be seen in figure 6.7. Like the “clean” model, the “soiled” model were not able to detect plates, while other plates were detected in the same image. Additionally, the “soiled” model also tend to not detect large plates or plates that are near the camera. In general, the heatmaps of the “soiled” model tend to be more nosy compared to the generated heatmaps of the “clean” model, when evaluated on clean tableware.

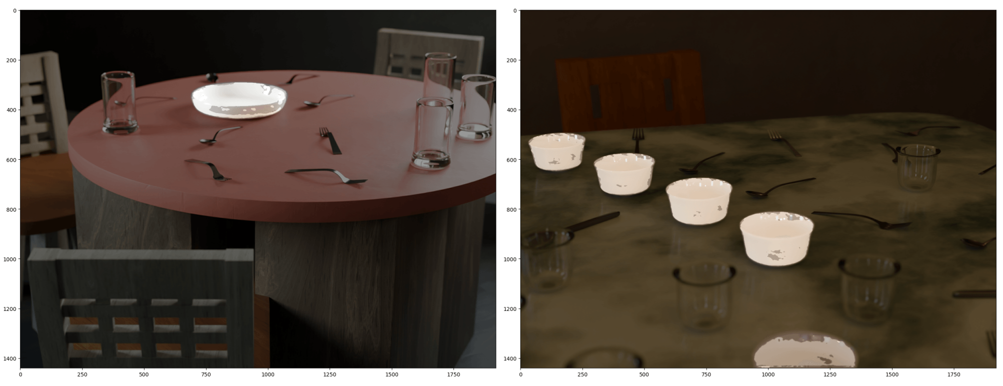


Figure 6.5 Soiled Model: Detection of Clean Plates

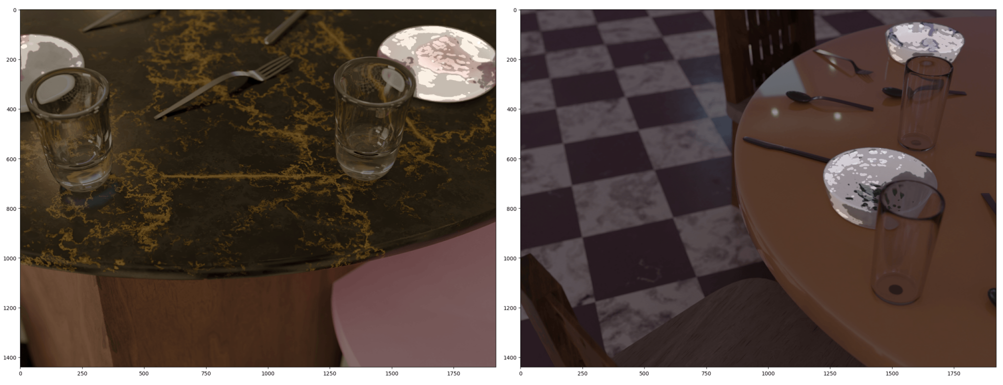


Figure 6.6 Soiled Model: Identification of Glass Occlusion

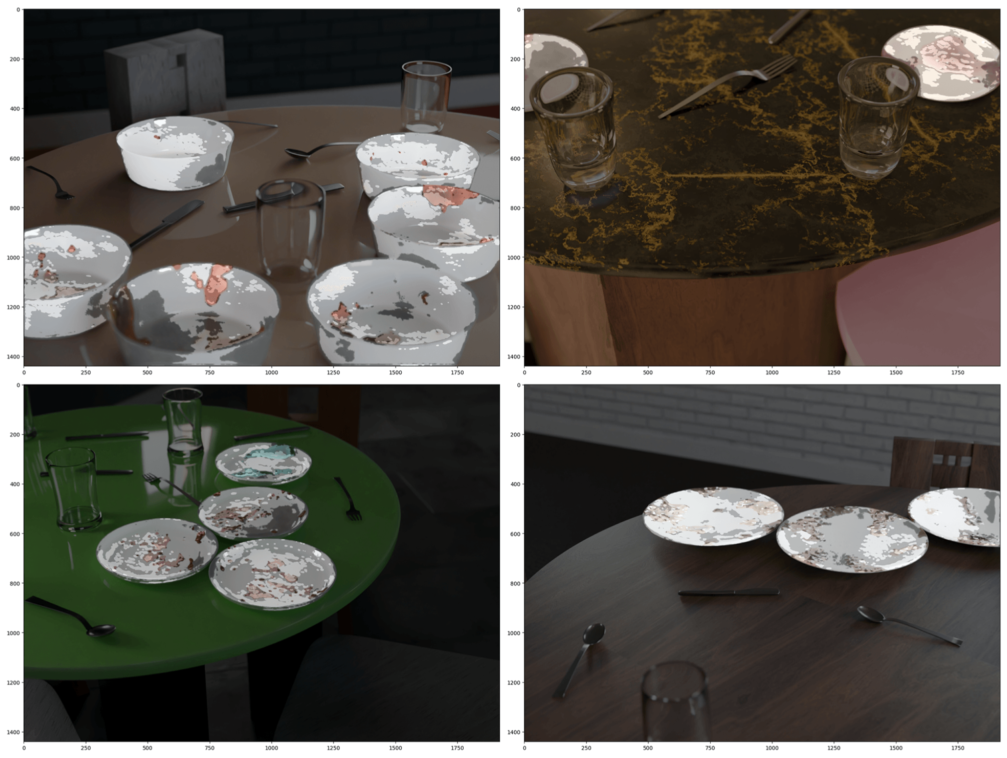


Figure 6.7 Clean Model: Soil classified as Plate

7. Discussion

In the following chapter, the results of the model training regarding the metrics, and the evaluation of the model are discussed.

Considering the precision curve in figure 5.5 and the achieved precision score, the trained model in both approaches can identify pixels that belong to a plate correctly in seen and unseen images. Regarding the recall curve in figure 5.6, the model is able to identify most of the pixels that belong to a plate in the training images, but not so well in comparison when predicting new images. The curve of the IoU metric shows that the model is slowly overfitting, because the IoU value increases in each epoch during training, but the validation IoU value decreases. The curve of the loss shows the same, but the training loss decreases and the validation loss increases. In addition, the IoU metric shows the accuracy of the heat map that overlaps with the ground truth. In both approaches, the model gets more accurate in aligning the heatmap output with the ground truth binary masks during training, but gets less accurate in unseen images.

The quantitative analysis in section 6.1 shows that the performance of tableware detectors, that are only trained on images with clean tableware, deteriorates when soiled tableware must be identified. This aspect lowers the real-world applicability of tableware detectors, that do not consider soiled tableware in their training data, and highlights that soiled-based occlusion should be taken into consideration in the model training process.

The qualitative analysis in section 6.2 shows potential problems of the implemented tableware detector that could be researched on in future work. As seen in figure 6.2 the tableware detector cannot differentiate between reflections on or of the plates and the plate itself. In addition, the receptive field of the model might be too small, which causes large plates or plates near the camera to be misclassified and undetected. Considering the heatmaps of the ground truth images (see figure 4.3), the lack of images in the dataset that contain large plates or plates near the camera might also be a reason, because there are fewer ground truth labels on the border of the images.

Regarding the performance of the tableware detector, after it is trained on images with soiled tableware, the model might perform better, if a larger dataset with more variety in the positions of the soiled tableware is created and used. The heatmap in figure 4.3 and the calculation in figure 5.2 show that there are less ground truth pixels in the soiled tableware dataset compared to the amount of ground truth pixels in the clean tableware dataset. The center of the heatmap of the soiled tableware dataset is smaller and there are approximately three-hundred million more ground truth pixels in the clean tableware dataset than in the soiled tableware dataset. For future work, this should be taken into consideration in the dataset creation. In addition, the mixture of images of clean and soiled tableware in the training dataset can be further researched, because this thesis only used fifty percent of images from the clean tableware dataset and fifty percent of images of from the soiled tableware dataset.

For future work, this thesis lacks comparison to other tableware detector models. Furthermore, in this thesis, real images were not considered and thus the domain gap between synthetic and real images in context of tableware detectors cannot be analyzed. For example, real images can be collected and mixed with the synthetic images to create a t-SNE visualization to analyze the domain gap of the real and synthetic data like it was done in Mill et al.'s work [56]. The dataset can also be extended by adding flying distractors or perform image augmentation such as sensor effects, random resized crop, random horizontal flip, random vertical flip, random rotation or color jitter. The effect of soiled-base occlusion can be further analyzed by comparing the dataset created in this work with a new dataset that uses random abstract patterns as occlusion on plates instead of soil. Last, the dataset generator is limited in the amount of the procedural generated objects and materials. In addition, the materials were not created in a physically-based approach, but were creative work, which could affect the realism of the image data.

8. Conclusion

The aim of this thesis is to address the research gap of missing data of soiled tableware in the domain of tableware detectors and robotics, and analyze the effects of soil-based occlusion in object detection models if soil is considered or not in the training process of ANN models.

Related literature lack the implementation of images of soiled tableware in the training data of their tableware detection models and were limited in the available data and variability in the dataset such as the different shapes of plates. This research gap will be addressed by the synthetic tableware dataset generator, which generates synthetic image data of a procedural generated dining room scene with clean or soiled tableware for a pixel-wise binary classification task. Additionally, an analysis of soiled-based occlusion was conducted by training a CNN on a dataset with only images of clean tableware and then training the same CNN on a dataset with images of clean and soiled tableware.

The results of the analysis show that the detection performance of the model that is trained only on images of clean tableware, will deteriorate in case of soiled tableware detection and might affect the real-world applicability of the tableware detector models that are only trained on images of clean data.

The most serious limitation of this work is the lack of comparison to other tableware detector models. In addition, the inclusion of real image to analyze the domain gap between real and synthetic image in the domain of tableware detection.

Bibliography

[1] *3D design software - Adobe Substance 3D*. en-US. URL: *https://www.adobe.com/products/substance3d.html* (visited on 10/20/2024).

[2] *Amazon announces 2 new ways it's using robots to assist employees and deliver for customers*. en. Section: Operations. Oct. 2023. URL: *https://www.aboutamazon.com/news/operations/amazon-introduces-new-robotics-solutions* (visited on 11/24/2023).

[3] Zap Andersson, Paul Edmondson, Julien Guertault, Adrien Herubel, Alan King, Peter Kutz, Andréa Machizaud, Jamie Portsmouth, Frédéric Servant, and Jonathan Stone. *OpenPBR Surface Specification*. Tech. rep. Academy Software Foundation (ASWF), 2024. URL: *https://academysoftwarefoundation.github.io/OpenPBR/*.

[4] Architecture Topics. *Best Glass Material in Blender Tutorial 2023*. Sept. 2023. URL: *https://www.youtube.com/watch?v=MzIreMJRhqk* (visited on 10/20/2024).

[5] Vince Austria, Edwin Arboleda, and Elbert Galas. "Image Processing Of Clean And Dirty Dishes To Design And Construct A Fuzzy Logic Dishwasher". In: *International Journal of Scientific & Technology Research* 8 (Dec. 2019), pp. 2470–73.

[6] F. O. Bartell, E. L. Dereniak, and W. L. Wolfe. "The Theory And Measurement Of Bidirectional Reflectance Distribution Function (Brdf) And Bidirectional Transmittance Distribution Function (BTDF)". In: *Radiation Scattering in Optical Systems*. Vol. 0257. SPIE, Mar. 1981, pp. 154–160. DOI: *10.1117/12.959611*. URL: *https://www.spiedigitallibrary.org/conference-proceedings-of-spie/0257/0000/The-Theory-And-Measurement-Of-*

Bidirectional-Reflectance-Distribution-Function-Brdf/10.1117/12.959611.full (visited on 10/20/2024).

[7] Kim Benzinger. *Was sind Lumen und wie viel braucht man?* de-DE. Nov. 2023. URL: *https://lampify.de/lichtlexikon/was-sind-lumen-und-wie-viel-braucht-man/* (visited on 10/20/2024).

[8] blenderbitesize. *CREATE A PROCEDURAL GLAZED CERAMIC MATERIAL IN BLENDER V4*. Nov. 2024. URL: *https://www.youtube.com/watch?v=5c-ba0vyNmg* (visited on 10/20/2024).

[9] Richard Bormann, Xinjie Wang, Jiawen Xu, and Joel Schmidt. "DirtNet: Visual Dirt Detection for Autonomous Cleaning Robots". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2020, pp. 1977–1983. DOI: *10.1109/ICRA40945.2020.9196559*. URL: *https://ieeexplore.ieee.org/abstract/document/9196559* (visited on 10/04/2023).

[10] *Borosilicate Glass vs. Soda Lime Glass? - Rayotek News*. Apr. 2017. URL: *https://web.archive.org/web/20170423152846/https://rayotek.com/wpnews/borosilicate-glass-vs-soda-lime-glass/* (visited on 10/20/2024).

[11] David Brunner and Fabian Schmid. "Synthetic Data in Automatic Number Plate Recognition". en. In: *Database and Expert Systems Applications - DEXA 2022 Workshops*. Ed. by Gabriele Kotsis, A. Min Tjoa, Ismail Khalil, Bernhard Moser, Alfred Taudes, Atif Mashkoor, Johannes Sametinger, Jorge Martinez-Gil, Florian Sobieczky, Lukas Fischer, Rudolf Ramler, Maqbool Khan, and Gerald Czech. Communications in Computer and Information Science. Cham: Springer International Publishing, 2022, pp. 112–118. ISBN: 978-3-031-14343-4. DOI: *10.1007/978-3-031-14343-4_11*.

[12] Shehan Caldera, Alexander Rassau, and Douglas Chai. "Review of Deep Learning Methods in Robotic Grasp Detection". en. In: *Multimodal Technologies and Interaction* 2.3 (Sept. 2018). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 57. ISSN: 2414-4088. DOI: *10.3390/mti2030057*. URL: *https://www.mdpi.com/2414-4088/2/3/57* (visited on 11/24/2023).

[13] *Camera Sensor Size in Photography - Why it Matters!* URL: *https://capturetheatlas.com/camera-sensor-size/* (visited on 10/20/2024).

[14] Daniel Canedo, Pedro Fonseca, Petia Georgieva, and António J. R. Neves. "A Deep Learning-Based Dirt Detection Computer Vision System for Floor-Cleaning Robots with Improved Data Collection". en. In: *Technologies* 9.4 (Dec. 2021). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 94. ISSN: 2227-7080. DOI: *10.3390/technologies9040094*. URL: *https://www.mdpi.com/2227-7080/9/4/94* (visited on 10/04/2023).

[15] Yuhao Chen, Yue Luo, and Boyi Hu. "Towards Next Generation Cleaning Tools: Factors Affecting Cleaning Robot Usage and Proxemic Behaviors Design". In: *Frontiers in Electronics* 3 (2022). ISSN: 2673-5857. URL: *https://www.frontiersin.org/articles/10.3389/felec.2022.895001* (visited on 11/24/2023).

[16] Tae Eun Choe, Jane Wu, Xiaolin Lin, Karen Kwon, and Minwoo Park. *HazardNet: Road Debris Detection by Augmentation of Synthetic Models*. Issue: arXiv:2303.07547 arXiv:2303.07547 [cs]. Mar. 2023. DOI: *10.48550/arXiv.2303.07547*. URL: *http://arxiv.org/abs/2303.07547* (visited on 10/04/2023).

[17] Michael Chui, James Manyika, and Mehdi Miremadi. "Where machines could replace humans—and where they can't (yet)". en. In: (2016).

[18] *Cleaned vs Dirty V2*. en. URL: *https://kaggle.com/competitions/platesv2* (visited on 10/20/2024).

[19] Guillem Delgado, Andoni Cortés, Sara García, Estíbaliz Loyo, Maialen Berasategi, and Nerea Aranjuelo. "Methodology for generating synthetic labeled datasets for visual container inspection". In: *Transportation Research Part E: Logistics and Transportation Review* 175 (July 2023), p. 103174. ISSN: 1366-5545. DOI: *10.1016/j.tre.2023.103174*. URL: *https://www.sciencedirect.com/science/article/pii/S136655452300162X* (visited on 10/04/2023).

[20] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering". In: *Journal of Open Source Software* 8.82 (2023), p. 4901. DOI: *10.21105/joss.04901*. URL: *https://doi.org/10.21105/joss.04901*.

[21] *Dimensions | Database of Dimensioned Drawings*. en. URL: *https://www.dimensions.com* (visited on 10/20/2024).

[22] *Dining Room - Square, Formal (Medium) Dimensions & Drawings | Dimensions.com*. en. URL: *https://www.dimensions.com/element/dining-room-square-formal-medium* (visited on 10/20/2024).

[23] Trung Huy Duong, Mohsen Emami, and Lawrence Larry Hoberock. "Automatic Dishware Inspection: Applications and Comparisons of Two New Methods". In: *2011 10th International Conference on Machine Learning and Applications and Workshops*. Vol. 1. Dec. 2011, pp. 25–30. DOI: *10.1109/ICMLA.2011.40*.

[24] Jon Dykstra. *Parts of a Table (Dining Room and Coffee Table Diagrams)*. en-US. Feb. 2018. URL: *https://www.homestratosphere.com/parts-of-table/* (visited on 10/20/2024).

[25] *Each type of camera lens explained | photoGuard*. en. URL: *https://www.photoguard.co.uk/camera-lens-guide* (visited on 10/20/2024).

[26] F. H. A. Editor. *Wine Glasses Guide: Different Types and Their Uses*. en-US. May 2024. URL: *https://fhafnb.com/blog/types-of-wine-glasses/* (visited on 10/20/2024).

[27] *Employment by major industry sector : U.S. Bureau of Labor Statistics*. en. URL: *https://www.bls.gov/emp/tables/employment-by-major-industry-sector.htm* (visited on 11/23/2023).

[28] *Esstischbeleuchtung - Tipps für gutes Licht von LAMPADA*. de-DE. Sept. 2021. URL: *https://lampada.de/tipps-zur-esstischbeleuchtung/* (visited on 10/20/2024).

[29] *Fork image - Visual Dictionary Online*. URL: *https://www.visualdictionaryonline.com/food-kitchen/kitchen/silverware/fork.php* (visited on 10/20/2024).

[30] Yudai Fukuzawa, Zhongkui Wang, Yoshiki Mori, and Sadao Kawamura. "A Robotic System Capable of Recognition, Grasping, and Suction for Dishwashing Automation". In: *2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. Nov. 2021, pp. 369–374. DOI: *10.1109/M2VIP49856.2021.9665169*.

[31] Juan Miguel Garcia-Haro, Edwin Daniel Oña, Juan Hernandez-Vicen, Santiago Martinez, and Carlos Balaguer. "Service Robots in Catering Applications: A Review and Future Challenges". en. In: *Electronics* 10.1 (Jan. 2021). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 47.

ISSN: 2079-9292. DOI: *10.3390/electronics10010047*. URL: *https://www.mdpi.com/2079-9292/10/1/47* (visited on 09/21/2023).

[32] *Grundriss planen - Richtwerte für Raumgrößen*. de. URL: *https://www.fertighaus.de/ratgeber/hausbau/grundriss-planen-richtwerte-fuer-raumgroessen/* (visited on 10/20/2024).

[33] Shangbin Guan and Gang Peng. "Dirt Detection and Segmentation Network for Autonomous Washing Robots". en. In: *Pattern Recognition and Computer Vision*. Ed. by Shiqi Yu, Zhaoxiang Zhang, Pong C. Yuen, Junwei Han, Tieniu Tan, Yike Guo, Jianhuang Lai, and Jianguo Zhang. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2022, pp. 691–703. ISBN: 978-3-031-18913-5. DOI: *10.1007/978-3-031-18913-5_53*.

[34] Frank E. Harrell Jr., Kerry L. Lee, Robert M. Califf, David B. Pryor, and Robert A. Rosati. "Regression modelling strategies for improved prognostic prediction". en. In: *Statistics in Medicine* 3.2 (1984), pp. 143–152. ISSN: 1097-0258. DOI: *10.1002/sim.4780030207*. URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4780030207* (visited on 10/20/2024).

[35] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask R-CNN*. Jan. 2018. DOI: *10.48550/arXiv.1703.06870*. URL: *http://arxiv.org/abs/1703.06870* (visited on 11/24/2023).

[36] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. *On Pre-Trained Image Features and Synthetic Images for Deep Learning*. Nov. 2017. URL: *http://arxiv.org/abs/1710.10710* (visited on 10/04/2023).

[37] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Martina Marek, and Martin Bokeloh. *An Annotation Saved is an Annotation Earned: Using Fully Synthetic Training for Object Instance Detection*. Issue: arXiv:1902.09967 arXiv:1902.09967 [cs]. Feb. 2019. DOI: *10.48550/arXiv.1902.09967*. URL: *http://arxiv.org/abs/1902.09967* (visited on 10/04/2023).

[38] Vedhus Hoskere, Yasutaka Narazaki, and Billie F. Spencer. "Physics-Based Graphics Models in 3D Synthetic Environments as Autonomous Vision-Based Inspection Testbeds". en. In: *Sensors* 22.2 (Jan. 2022). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 532. ISSN: 1424-8220. DOI: *10.3390/s22020532*. URL: *https://www.mdpi.com/1424-8220/22/2/532* (visited on 10/04/2023).

[39] *Ideale Raumgrößen*. de. URL: *https://www.fertighauswelt.de/magazin/ratgeber/ideale-raumgroessen* (visited on 10/20/2024).

[40] *Industrial robotics market revenue worldwide 2018-2028*. en. URL: *https://www.statista.com/statistics/760190/worldwide-robotics-market-revenue/* (visited on 11/23/2023).

[41] *IOR / Index of Refraction List - Pixel and Poly*. URL: *https://pixelandpoly.com/ior.html* (visited on 10/20/2024).

[42] *IOR List for PBR Materials*. en. Nov. 2017. URL: *http://blendersauce.com/ior-list/* (visited on 10/20/2024).

[43] Joey Carlino. *Splatter Texture Generator // Blender 2.93*. July 2021. URL: *https://www.youtube.com/watch?v=_shVogE4t3o* (visited on 10/20/2024).

[44] JRoss 3D. *Household Carpet Material | Blender Tutorial*. Sept. 2022. URL: *https://www.youtube.com/watch?v=OuH_1BO7Ti4* (visited on 10/20/2024).

[45] Katherine. *Collector's Notes: Antique Plates*. en-US. Apr. 2021. URL: *https://penderandpeony.com/2021/04/collectors-notes-antique-plates/* (visited on 10/20/2024).

[46] Jaeseok Kim, Anand Kumar Mishra, Raffaele Limosani, Marco Scafuro, Nino Cauli, Jose Santos-Victor, Barbara Mazzolai, and Filippo Cavallo. "Control strategies for cleaning robots in domestic applications: A comprehensive review". In: *International Journal of Advanced Robotic Systems* 16.4 (July 2019). Number: 4 Publisher: SAGE Publications, p. 1729881419857432. ISSN: 1729-8806. DOI: *10.1177/1729881419857432*. URL: *https://doi.org/10.1177/1729881419857432* (visited on 09/21/2023).

[47] Jun Kinugawa, Hiroki Suzuki, Junya Terayama, and Kazuhiro Kosuge. "Underactuated robotic hand for a fully automatic dishwasher based on grasp stability analysis*". In: *Advanced Robotics* 36.4 (Feb. 2022), pp. 167–181. ISSN: 0169-1864. DOI: *10.1080/01691864.2021.2011778*. URL: *https://doi.org/10.1080/01691864.2021.2011778* (visited on 09/22/2023).

[48] Lachlan Sarv. *Procedural Wood Material - Blender Tutorial*. Oct. 2022. URL: *https://www.youtube.com/watch?v=jDEijCwz6to* (visited on 10/20/2024).

[49] *Learn How to Identify the Different Types of Dinnerware*. en. Section: The Spruce Eats. URL: *https://www.thespruceeats.com/dinnerware-materials-908883* (visited on 10/20/2024).

[50] *Light Objects - Blender 4.2 Manual*. URL: *https://docs.blender.org/manual/en/latest/render/lights/light_object.html#power-of-lights* (visited on 10/20/2024).

[51] Bernard Marr. *The Future Of Delivery Robots*. en. Section: Enterprise Tech. URL: *https://www.forbes.com/sites/bernardmarr/2021/11/05/the-future-of-delivery-robots/* (visited on 11/24/2023).

[52] Nestor Maslej, Loredana Fattorini, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Helen Ngo, Juan Carlos Niebles, Vanessa Parli, Yoav Shoham, Russell Wald, Jack Clark, and Raymond Perrault. *Artificial Intelligence Index Report 2023*. arXiv:2310.03715 [cs]. Oct. 2023. DOI: *10.48550/arXiv.2310.03715*. URL: *http://arxiv.org/abs/2310.03715* (visited on 11/23/2023).

[53] *Material Index of Refraction*. URL: *https://www.spacekdet.com/tutorials/IOR.html* (visited on 10/20/2024).

[54] *Materials Introduction - Blender 4.2 Manual*. URL: *https://docs.blender.org/manual/en/latest/render/materials/introduction.html#physically-based-shading* (visited on 10/20/2024).

[55] Jeffery C. Mays. "400-Pound N.Y.P.D. Robot Gets Tryout in Times Square Subway Station". en-US. In: *The New York Times* (Sept. 2023). ISSN: 0362-4331. URL: *https://www.nytimes.com/2023/09/22/nyregion/police-robot-times-square-nyc.html* (visited on 11/24/2023).

[56] Leonid Mill, David Wolff, Nele Gerrits, Patrick Philipp, Lasse Kling, Florian Vollnhals, Andrew Ignatenko, Christian Jaremenko, Yixing Huang, Olivier De Castro, Jean-Nicolas Audinot, Inge Nelissen, Tom Wirtz, Andreas Maier, and Silke Christiansen. "Synthetic Image Rendering Solves Annotation Problem in Deep Learning Nanoparticle Segmentation". en. In: *Small Methods* 5.7 (2021), p. 2100223. ISSN: 2366-9608. DOI: *10.1002/smtd.202100223*. URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/smtd.202100223* (visited on 10/04/2023).

[57] *Parts of a Chair: An In-Depth Look at Chair Components*. Sept. 2023. URL: *https://vaseat.com/parts-of-a-chair/* (visited on 10/20/2024).

[58] Peter Peduzzi, John Concato, Elizabeth Kemper, Theodore R. Holford, and Alvan R. Feinstein. "A simulation study of the number of events per variable

in logistic regression analysis". English. In: *Journal of Clinical Epidemiology* 49.12 (Dec. 1996), pp. 1373–1379. ISSN: 0895-4356, 1878-5921. DOI: *10.1016/S0895-4356(96)00236-3*. URL: *https://www.jclinepi.com/article/S0895-4356(96)00236-3/abstract* (visited on 10/20/2024).

[59] Philip Kriegel. *Common Mistakes When Making Glass / How To Make A Glass In Blender (Beginner Tutorial)*. Nov. 2020. URL: *https://www.youtube.com/watch?v=w_mYYy4-5PU* (visited on 10/20/2024).

[60] *PHOTOGRAPHER 5*. URL: *https://sites.google.com/view/photographer-5-documentation* (visited on 10/20/2024).

[61] *PHOTOGRAPHER 5 - Exposure*. URL: *https://sites.google.com/view/photographer-5-documentation/cameras/exposure* (visited on 10/20/2024).

[62] Pixel Adventures. *Tutorial: Procedural blood splatters in Blender*. Apr. 2021. URL: *https://www.youtube.com/watch?v=5VdYP6q0Fmg* (visited on 10/20/2024).

[63] Param S. Rajpura, Hristo Bojinov, and Ravi S. Hegde. *Object Detection Using Deep CNNs Trained on Synthetic Images*. Issue: arXiv:1706.06782 arXiv:1706.06782 [cs]. Sept. 2017. DOI: *10.48550/arXiv.1706.06782*. URL: *http://arxiv.org/abs/1706.06782* (visited on 10/04/2023).

[64] *Random Room Constructor — BlenderProc 2.7.0 documentation*. URL: *https://dlr-rm.github.io/BlenderProc/examples/advanced/random_room_constructor/README.html#implementation* (visited on 10/20/2024).

[65] Rodrigo Rill-García, Eva Dokladalova, and Petr Dokladal. "Syncrack: Improving Pavement and Concrete Crack Detection through Synthetic Data Generation". In: Jan. 2022, pp. 147–158. DOI: *10.5220/0010837300003124*.

[66] *Ripe for the future – with our harvesting robot*. en-US. URL: *https://www.iav.com/en/full-berry-into-the-future-with-our-new-harvesting-robot/* (visited on 11/24/2023).

[67] Ryan King Art. *Procedural Biscuit Material (Blender Tutorial)*. Dec. 2022. URL: *https://www.youtube.com/watch?v=52dC0yBS35I* (visited on 10/20/2024).

[68] Ryan King Art. *Procedural Black Veined Marble Material (Blender Tutorial)*. June 2024. URL: *https://www.youtube.com/watch?v=NZrWP9SJ3Kc* (visited on 10/20/2024).

[69] Ryan King Art. *Procedural Brick Material (Blender Tutorial)*. July 2021. URL: *https://www.youtube.com/watch?v=2MxQUaMlk3A* (visited on 10/20/2024).

[70] Ryan King Art. *Procedural Checkered Kitchen Marble Floor Material (Blender Tutorial)*. June 2023. URL: *https://www.youtube.com/watch?v=r5rwaimsZs0* (visited on 10/20/2024).

[71] Ryan King Art. *Procedural Dark Wood Floor Boards Material (Blender Tutorial)*. Apr. 2024. URL: *https://www.youtube.com/watch?v=rHjbMVGDqGQ* (visited on 10/20/2024).

[72] Ryan King Art. *Procedural Marble Material (Blender Tutorial)*. July 2021. URL: *https://www.youtube.com/watch?v=wTzk9T06gdw* (visited on 10/20/2024).

[73] Ryan King Art. *Procedural Meatball Material (Blender Tutorial)*. Oct. 2022. URL: *https://www.youtube.com/watch?v=qdVOXc7zs24* (visited on 10/20/2024).

[74] Ryan King Art. *Procedural Metal Materials (Blender Tutorial)*. Sept. 2024. URL: *https://www.youtube.com/watch?v=U_jjy8zmXMY* (visited on 10/20/2024).

[75] Ryan King Art. *Procedural Plaster Material (Blender Tutorial)*. Sept. 2021. URL: *https://www.youtube.com/watch?v=EwB3HWcUdEk* (visited on 10/20/2024).

[76] Ryan King Art. *Procedural Rough Metal Material (Blender Tutorial)*. Mar. 2022. URL: *https://www.youtube.com/watch?v=Niech2KIWgI* (visited on 10/20/2024).

[77] Ryan King Art. *Procedural Scratched Plastic (Blender Tutorial)*. Mar. 2022. URL: *https://www.youtube.com/watch?v=l0whu3494_c* (visited on 10/20/2024).

[78] Ryan King Art. *Procedural Smooth Concrete Material (Blender Tutorial)*. Feb. 2022. URL: *https://www.youtube.com/watch?v=uCyUt1Jaufk* (visited on 10/20/2024).

[79] Ryan King Art. *Procedural Tile Floor Material (Blender Tutorial)*. Jan. 2023. URL: *https://www.youtube.com/watch?v=BtUDs6sxgaw* (visited on 10/20/2024).

[80] *Spoon image - Visual Dictionary Online*. URL: *https://www.visualdictionaryonline.com/food-kitchen/kitchen/silverware/spoon.php* (visited on 10/20/2024).

[81] Mingxing Tan, Ruoming Pang, and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. arXiv:1911.09070 [cs, eess]. July 2020. DOI: *10.48550/arXiv.1911.09070*. URL: *http://arxiv.org/abs/1911.09070* (visited on 11/24/2023).

[82] *The best camera settings for indoor photography - Adobe*. en-US. URL: *https://www.adobe.com/creativecloud/photography/hub/guides/camera-settings-indoor-photography.html* (visited on 10/20/2024).

[83] 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*. Issue: arXiv:1804.06516 arXiv:1804.06516 [cs]. Apr. 2018. DOI: *10.48550/arXiv.1804.06516*. URL: *http://arxiv.org/abs/1804.06516* (visited on 10/04/2023).

[84] *Types of Flatware*. en. URL: *https://www.webstaurantstore.com/guide/585/different-types-of-flatware.html* (visited on 10/20/2024).

[85] *Volume Absorption - Blender 4.2 Manual*. URL: *https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/volume_absorption.html* (visited on 10/20/2024).

[86] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. arXiv:2207.02696 [cs]. July 2022. DOI: *10.48550/arXiv.2207.02696*. URL: *http://arxiv.org/abs/2207.02696* (visited on 11/24/2023).

[87] Jianfeng Wang and Xiaolin Hu. “Convolutional Neural Networks with Gated Recurrent Connections”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). arXiv:2106.02859 [cs], pp. 1–1. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: *10.1109/TPAMI.2021.3054614*. URL: *http://arxiv.org/abs/2106.02859* (visited on 11/24/2023).

[88] *Wie groß sollten Räume sein? - 25 wichtige Richtwerte*. de-DE. URL: *https://plan7architekt.com/i/wie-gross-sollten-raeume-sein-richtwerte-ai51/* (visited on 10/20/2024).

[89] Xuebin Yue, Hengyi Li, Masao Shimizu, Sadao Kawamura, and Lin Meng. "YOLO-GD: A Deep Learning-Based Object Detection Algorithm for Empty-Dish Recycling Robots". en. In: *Machines* 10.5 (May 2022). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 294. ISSN: 2075-1702. DOI: *10.3390/machines10050294*. URL: *https://www.mdpi.com/2075-1702/10/5/294* (visited on 09/22/2023).