Object Affordance Detection for Mobile Manipulation in Retail Environments

by

P.G. van Houtum

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday May 20, 2021 at 10:00 AM.

Student number:	4381424	
Project duration:	October 12, 2020 – May 20, 2021	
Thesis committee:	Ir. M. Spahn,	TU Delft, daily supervisor
	Dr. ir. J. Alonso-Mora,	TU Delft, supervisor
	Dr. ir. W. Pan,	TU Delft

TU Delft

Abstract

Modern retail stores are increasingly implementing automated systems with the aim of assisting both the customer and employee. Much research is conducted on robotic applications within such environments; one of these applications concerns deploying an autonomous mobile manipulator to assist humans in retail stores. Such a robot requires the ability to cope with unexpected environmental changes, like encountering obstructions in retail store aisles. Generally, the first step in the pipeline of an autonomous mobile robot is focused on gathering and processing environmental information using perception sensors like cameras and LiDARs. Subsequently, this information is used to plan the mobile robot's path, which is typically done without allowing interaction with the environment. For object detection techniques, objects are first localized, after which they are classified towards their object type category. Another approach is to classify objects towards their functional categories, which are commonly referred to as affordances. Research in the field of affordance classification mostly focuses on kitchen, garden and working tools due to the availability of affordance datasets for these objects. However, no work on affordances in retail store environments has been conducted to this date. More specifically, there is no dataset publicly available that allows mobile manipulators to identify how to interact with retail store related objects in such environments. This work has investigated the adaptation of an instance segmentation network to localize and classify objects on floors of retail stores into affordance classes. These affordance classes relate to the functional capabilities of mobile manipulators, like *graspable* or *pushable*. To achieve this, an affordance dataset consisting of retail store related objects is essential. To overcome the scarcity in this data, a novel dataset consisting of 3237 images with pixel-level affordance annotations was successfully created using an automated data generation approach. This work has shown that such an approach can be used to minimize human labour in terms of acquiring annotated data drastically. A state-of-the-art instance segmentation network was trained using this synthetically generated data and was tested on both synthetic and real image data. The evaluation revealed that the use of synthetic data for training allows for inference on real image data, yet this may compromise the localization and especially the classification performance. Further, the observation of objects that are assigned to both affordance classes has introduced the aspect of the subjectivity of affordances.

Nomenclature

AUC	Area Under Curve
DOF	Degree Of Freedom
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
GUI	Graphical User Interface
IoU	Intersection over Union
LiDAR	Light Detection And Ranging
mAP	mean Average Precision
R-CNN	Region-Convolutional Neural Network
RoI	Region of Interest
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
TN	True Negative
TP	True Positive

Contents

1 Introduction

1.1. AIRLab Delft

This master's thesis research was conducted within the AI for Retail (AIR) Lab Delft, which is a joint TU Delft-Ahold Delhaize industry lab focused on research in robotics in retail environments. The possibilities of implementing robots to assist humans in such a retail environment are endless. The lab has chosen to invest in a mobile manipulator robot named Albert, consisting of a mobile base with a 7-DOF manipulator arm mounted to it. Combining a mobile base with a manipulator arm offers a wide variety of options in performing actions autonomously. The possibility to use Albert in AIRLab's mock-up retail store environment enables realistic testing. Also, the lab provides high-performance desktops for computationally expensive tasks, like the training of a neural network. For more information on AIRLab Delft, please visit `https://icai.ai/airlab-delft/`.

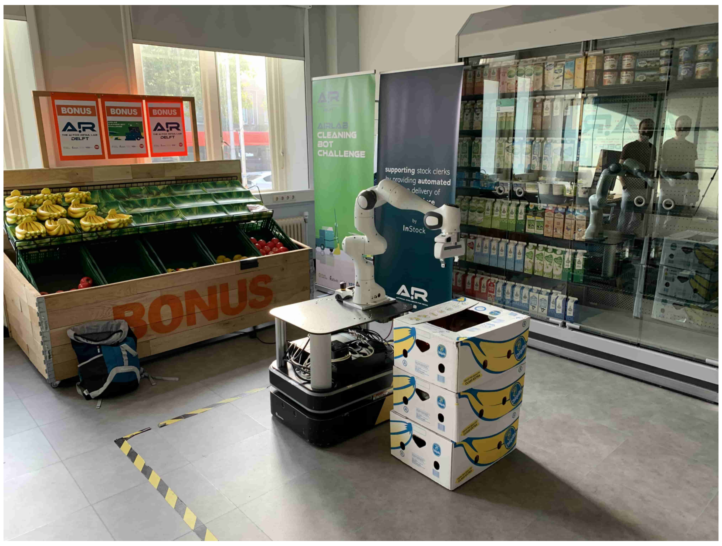


Figure 1.1: Albert, the 7-DOF mobile manipulator performing a banana-grasping action in AIRLab's mock-up store.

1.2. Motivation

Generally, the first step in the pipeline of an autonomous mobile robot is focused on gathering and processing environmental information. This is generally done using perception sensors, like mono cameras, stereo cameras or LiDAR sensors. The content of this research lies in the domain of computer vision, in which perception sensor data is processed to obtain specific information on the robot's environment.

The aim of Albert the robot is to autonomously drive through a retail environment and perform actions to assist customers and employees. This requires the ability to cope with unexpected changes, like encountering obstructions. Presumably, the robot would run across many people, shopping carts and baskets. However, obstructions like products that have fallen of shelves and other unpredictable encounters must also be tackled. In the domain of computer vision, much research has been conducted in the field of object detection, in which visual features are classified into object type categories. Another approach is to classify these features into functional categories, also known as affordances, which will shortly be explained below.

Affordances
The human brain is trained to correlate object appearances to its function. For instance, humans know that a soccer ball can be rolled by giving it a push. Or that a concave-shaped object, e.g. a bowl or a cup, can contain liquid substances. This brings up the biologically inspired concept of affordances, introduced by the American psychologist J.J. Gibson [11].

> "The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. The verb to afford is found in the dictionary, the noun affordance is not. I have made it up. I mean by it something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment."
>
> *J.J. Gibson*

This biologically inspired concept has been successfully applied to the field of robotics, which will be covered in the next chapter. Generally, these applications focus on the detection of affordances of kitchen, garden and working tools. To this date, the application of affordances in retail environments using mobile manipulation has not been explored to the best of my knowledge. This concept is believed to have great potential in detecting the interactability of encountered objects in a retail environment, which will therefore be investigated in this work. More specifically, the use of a synthetically generated affordance image dataset for retail products will be explored. The reason for this exploration roots in the lack of a suitable retail object image dataset that is labelled towards affordance classes in terms of interactability, e.g. *graspable* and *pushable*. The work conducted in this research is based on several research questions and assumptions, which will be introduced next.

1.3. Research Question

The main research question of this master's thesis is formulated as follows.

- *How can the concept of affordances be adopted to classify objects that are commonly found in retail environments into their ability to be interacted with, given the use of a mobile manipulator?*

This main research question can be divided into several sub-questions, whose answers aim to contribute to the answer of the main research question.

1. How can a conventional object detection model for 2D images be adapted to classify into affordance labels?
2. Can the use of a synthetically generated retail product image dataset solve the issue of the lack of appropriate affordance training data?
3. Is the resulting affordance model able to generalize objects into their level of interactability, e.g. being *graspable* or *pushable*?

1.4. Assumptions

- The objects that are commonly found in retail store environments consist of products, e.g. packaged food, drinks, fruits, vegetables, household products and shopping baskets.
- Only objects that are located on floors of retail environments are aimed to be detected.
- All objects are considered to be static, meaning that objects are not changing location over time.
- This work is aimed for mobile manipulators deployed in a human shared retail environment, e.g. supermarkets.
- There is a clear visual distinction between *graspable* and *pushable* objects.

1.5. Contributions

- The generation of a novel affordance dataset of 3237 annotated images that fills the gap in affordance datasets for retail store environments, publicly available at `https://doi.org/10.4121/14557965.v1`
- The design of a synthetic image data generation script that generates images accompanied with their semantic masks, consisting of object CAD models combined with real background images on the fly. A labelling application was created to annotate the created semantic masks towards object affordance classes efficiently.
- A state-of-the-art instance segmentation network was adopted to detect and classify retail store related objects located on floors into affordance classes using this novel dataset.

1.6. Thesis Outline

This finalizes the *Introduction* 1. In chapter 2, related work concerning various affordance applications in robotics will be discussed. In the *Methods* chapter (3), the content of this work's research will be covered by elaborating on the chosen object detector, the approach of synthetic data generation and the experimental setup that aims to answer the research question. Subsequently, the *Results* chapter (4) covers the experiment's outcome, which is analyzed in the *Discussion* (5). This master's thesis will be concluded in the *Conclusion* (6) by answering the main research question. Finally, several suggestions for further research are proposed in the *Future Work* chapter (7).

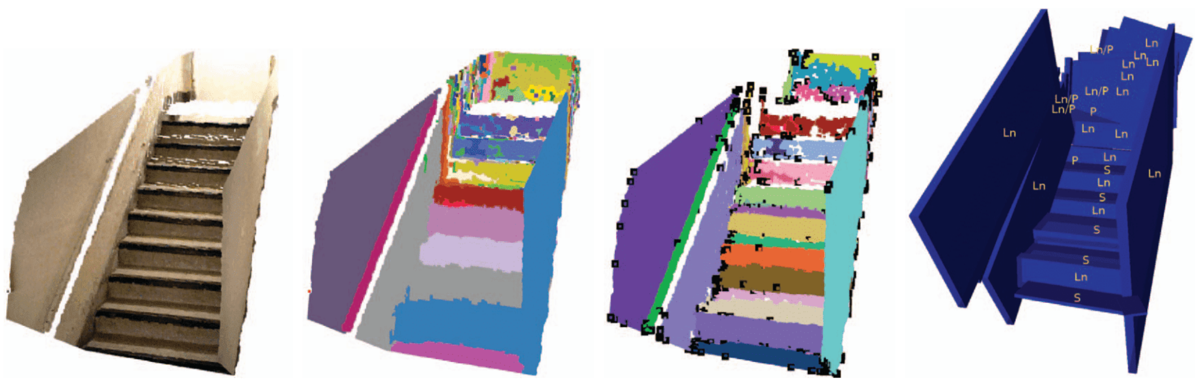
2
Affordance Detection in Robotics

The biologically inspired concept of affordances has been applied to robotic applications, reviewed in several surveys [15], [22], [36], [38]. Generally, these applications seek to detect what objects can afford, meaning that a set of actions may be taken, given the object and its environment. Imagine the case of a robot manipulator arm in a cluttered environment of random objects, one of which is a hammer. Classical object classification can be used to detect whether these objects can be manipulated or not by classifying objects by their type and use *a priori* knowledge of that type. In contrast, the approach of affordances aims to classify visual features that relate to an object's function, rather than its type. As an example, we may look at the hammer again, which consists of a relatively long stick with a specifically shaped head at its end. As humans, we perceive this stick as a grip that we know should be taken in our hand. Regardless of the object, all objects which have similarly shaped sticks, are known to be graspable by our hands. This is the basis of the idea of affordances in robotics and this is where it differs from classical object classification.

2.1. Learning-Free

Most affordance detection methods are based on trained models; however, some work shows that positive results can be obtained from approaches without the use of machine learning techniques by using methods based on conditional approaches or geometric shape fitting, for instance. As no training procedure takes place, there is no need for *a priori* annotated training data. However, this simplistic approach tends to generalize object shapes and therefore is not applicable for more complex affordance detection scenarios.

Ten Pas and Platt [32] are roboticists who chose such a conditional approach in which they identify a set of geometric conditions for the existence of a grasp affordance using a 3D point cloud model. A quadratic curve fitting function is applied to the data to find handle-shaped and hole-shaped object areas, which imply a grasp affordance. Due to the nature of this general conditional approach, the authors claim that this model is well applicable to many real-world robot application scenarios. Also, the authors incorporated their work into a ROS package, which is made publicly available [31]. Similarly, Kaiser et al. [16] segment RGB-D point cloud data and match these segments to their geometric primitives, such as planes, cylinders and spheres, whereafter a rule-based system is employed to obtain affordance predictions.



(a) Registered Point Cloud (b) Segmented Point Cloud (c) Primitives/Grasp Points (d) Affordances

Figure 2.1: Kaiser et al.'s approach of obtaining affordances from RGB-D point cloud data, by comparing the data with geometric primitives (planes, cylinders and spheres). In (d), affordances such as, lean (Ln), support (S) and Push (P) are labeled together with its primitive shapes. [16]

Another non-trained approach is proposed by Yu et al. [37], who analyse thoroughly scanned container objects for the affordance of containability of liquid under specific angular poses with respect to the gravitational direction. They first voxelized the data, whereafter an iterative process is started from the bottom to the top part of the object to search for holes in the hollow structure. This voxelization may cause errors where small holes are not detected, depending on the voxel resolution. A significant benefit of this method, as well as the other methods that do not require trained machine learning models, is that there is no need for large amounts of prior training data.

2.2. Supervised Learning

The majority of the work found on affordances in robotics for the last years is based on affordance learning in a supervised way. The model is trained on *a priori* annotated data, generally obtained from publicly available affordance datasets of kitchen, garden and working tools, like the UMD affordance dataset [23] or the IIT-AFF dataset [25]. These supervised learning methods can be categorized into three model approaches. In *Probabilistic Network Models*, the relations between input sensor data and the desired affordance output are generally captured in probabilistic network models like Bayesian/Markov Networks or other types of graph models. Secondly, *Conventional Machine Learning Models* may be used to train manually defined features to be classified or regressed to an affordance class label. Finally, a similar but more end-to-end affordance detection approach using *Deep Learning Models* can be used. These methods generally have high training times and low run times, which makes them useful to real-time applications. Below, the last approach will be elaborated as it contains the basis of this work.

Deep Learning

In contrast to the conventional supervised machine learning approaches, deep learning approaches do not use manually defined features but rather learn deep, complex features using convolutional layers. Additionally, deep learning is able to combine multiple steps into a single network. Object localization and object classification may be combined into a single network to obtain better performance.

AffordanceNet [9], is an example of this end-to-end approach, where an image is fed to a Convolutional Neural Network (CNN) for feature extraction. Important Regions of Interest (RoIs) are selected using a Region Proposal Network (RPN) [26]. Finally, a sequence of convolutional-deconvolutional and a softmax layer are used to classify by affordances and output an affordance mask. The strength of this method lays in the end-to-end way that object masks and affordance classes are detected. Resulting in a runtime of 150ms per RGB image on a Titan X GPU, allowing it to run on real-time applications. AffordanceNet uses an instance segmentation network named Mask R-CNN [14], which is a proven and publicly accessible model that will be used in this work. Similarly, the method proposed by Roy and Todorovic [28], runs at 150ms per RGB image on an NVIDIA Tesla K80 GPU. An RGB image is fed to three multi-scale CNNs to obtain a depth map, surface normals and semantic object labels. A fourth multi-scale CNN uses these three maps, together with the input RGB image, to acquire a final affordance prediction mask. The affordance detector may fail in the presence of object clutter and partial occlusion. A different approach is proposed by Kokic et al. [17], who voxelized 3D point cloud data in a pre-processing step into a binary 3D occupancy grid. The voxel grid is fed to an affordance detection network (AFF-CNN), which outputs grasping locations. In parallel, the voxel grid is passed on to an object classification and orientation estimation network (CO-CNN). Object class, orientation and grasp locations are used to formulate grasp constraints, which are needed for higher-level affordance tasks, like poking using a screwdriver after having grasped it. The authors have shown that all three entities are essential to a good affordance execution by their manipulator arm. For the 2017 pick-and-place Amazon Robotics Challenge, the MIT-Princeton Team took first place in the stowing task, using their multi-affordance grasping system [39]. They use two Fully Convolutional Networks (FCNs) for the detection of suction and grasping affordances, given multi-view RGB and depth images (RGB-D). Both FCNs output an affordance heat map that shows successful affordance probabilities for suction and grasping, respectively. The major limitation of this pick-and-place approach is that the method is unable to pick up objects that are occluded by other objects. The authors praise reinforcement learning techniques, as a promising solution.

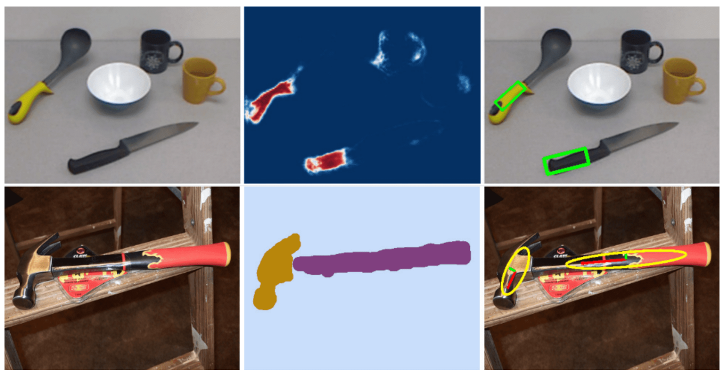


Figure 2.2: Examples of the output of affordance detection methods, where input images are shown in the first column and affordance predictions are visualized in the last two columns as; a heat map, a semantic mask, bounding boxes and as part pose frames. [24], [25]

Similarly to previously mentioned methods, Nguyen et al. [24] use a CNN. But before passing the RGB and depth image data on to the network, they decompose the depth image into a horizontal disparity image, a height above ground image and an image containing information on the angle between pixels' surface normals and the estimated gravity direction. With a runtime of 90ms on an NVIDIA Titan X GPU, CNN-HHA seems to be quite competitive. However, the same CNN but without depth image decomposition outperforms the method in terms of accuracy. The authors point to the nature of the used UMD dataset [23], for the inability of estimating a reliable gravity direction from a single depth image. As a consequence, the same authors have published follow-up work [25], in which their improved framework (BB-CNN-CRF) is described. Along with their improved framework, a newly published IIT-AFF affordance dataset was published, which has a broad variety of background scenes as opposed to the content of the UMD dataset. The authors show that object localization is paramount in such non-constant backgrounds, therefore they propose to pre-process images with an object detector network to narrow down the RoI. Nonetheless, this extra processing step may cause failure cases due to incorrect object localization. After the affordance detection network, the pixel-wise labelled output is post-processed, using dense Conditional Random Fields (CRFs). Affordance label probabilities and neighbourhood pixel labels are used in a dense CRF to remove inconsistencies and as a result improve accuracy.

2.3. Unsupervised and Self-Supervised Learning

Generally, supervised machine learning models increase in accuracy when trained on large amounts of annotated data. In areas where these large datasets may not be available, alternative approaches based on unsupervised and/or self-supervised learning may be useful. Here, the model finds patterns in data in an unsupervised way by clustering for instance. These clusters may be used as data to train a classifier, which in turn is referred to as self-supervising learning. However, if a comparison between unsupervised clustered data and supervised human-annotated data would be made, the latter would get the better hand in terms of accuracy.

An example of this approach is proposed by Ridge et al. [27], who use a multi-view learning method [30] to first cluster data from multiple feature spaces, into affordance category estimates in an unsupervised way.

The clusters are passed on to a classifier for training. One feature space consists of a large variety of pre-defined object shape features detected prior to interaction, whereas a second feature space contains effect features that are detected during and after interaction, using both 3D point cloud and 2D image data. Object detection limitations may occur due to the assumption that objects must lie on flat surfaces to be able to segment the object using a plane segmentation algorithm. Similar to this approach, Mar et al. [20] cluster Oriented Multi-Scale Extended Gaussian Image (OMS-EGI) geometric features, obtained from voxelized point cloud data, into tool-pose-based categories. Assuming that similar tool-poses have similar affordances, Generalized Regression Neural Networks (GRNNs) [26] are trained on these clusters to obtain a self-learned affordance detection model. Moreover, this method is an improvement of precedent work published by the same authors [21], but differs in the more simplistically used geometric feature representations and the use of a one-versus-all SVM model. The performance of this model is dependent on well clustered tool-pose data, in that regard the authors admit that clustering these kinds of categories is quite complex and may therefore be problematic. Ugur and Piater published work [33], [34], where feature extraction is performed on RGB-D point cloud data. More specifically, they differentiate between low-level human-defined geometric features and high-level self-learned features by exploration. Which in turn are clustered into basic effect categories at first, whereafter effect category complexity increases with the number of exploratory actions. A multiclass SVM model is self-trained using this unsupervised data generation approach. According to the authors, clustering into more complex effect categories is a challenging task, which may be solved by increasing exploratory actions, however this is a time-consuming process.

2.4. Reinforcement Learning

Finally, the concept of reinforcement learning in affordances will be introduced, where the robot is trained on its own successful or failed interaction with the environment, instead of relying on annotated examples. Authors point out that during early learning stages, these environments must be rather simplistic for the robot to be able to categorize its actions. As opposed to the other learning methods, reinforcement learning focuses on learning actions that can be performed using a specific robot and its environment, which is in line with Gibson's definition of affordances.

This concept is used by Ugur et al. in [35], where initially a robot discovers motion primitives by performing simple reach-and-enclose-on-contact actions on objects, whereafter it observes the caused effects visually and saves the action-effect combinations. In the next stage, the robot learns object affordances by applying specific actions to new objects and compares the result with the predicted effect. Whenever the observed result corresponds or does not correspond to the predicted effect, the action prediction is marked as a success or failure, respectively. An SVM is trained to map between an input object feature vector and the prediction of success or failure for certain actions. Cluttered environments may pose difficulties for this approach, as action-effect observations will be inconsistent and therefore, predictions may be less accurate. Also, in this particular approach the robot is not starting its learning curve from scratch, as it initially starts with simple pre-learned reach-and-enclose-on-contact actions. Another reinforcement learning method is proposed by Glover et al. [12], who use a Distributed Semi-Markov Decision Process (DSMDP). The robot uses this model together with an object as a goal position to learn the mapping between visually detected object positions and specific wheel movement actions (e.g. turn left, go forward). Also, the robot is equipped with a binary proximity sensor to detect whether an object is located between the robot's grippers. The affordance of being graspable is learned for specific objects. In the conducted experiments, objects were visually easily distinguishable by colour for simplicity, which would pose problems in real non-simplistic scenarios. The approach of detecting affordances using these reinforcement learning techniques are in line with Gibson's definition of affordances. Because, in contrast to both the learning-free and the other learning-based methods, reinforcement learning methods specifically focus on the robot and the object as twofold, instead of learning a model uniquely based on object information.

2.5. Summary

The state-of-the-art affordance approaches implemented for robotic use, may be categorized into four different types of learning approaches. *Learning-Free* approaches are the most simplistic and may be based on conditional or geometric shape fitting approaches. The second and most commonly used approach is based on *Supervised Learning*, in which a training stage processes large batches of labelled examples before performing inference on unseen data. *Deep Learning* is a promising sub-field in this category. Thirdly, *Unsu-*

pervised and Self-Supervised Learning methods obtain labelled data by finding patterns in an unsupervised way (e.g. clustering), which is then used as training data, hence the name Self-Supervised Learning. Lastly, *Reinforcement Learning* applications were discussed, in which an agent learns affordances by following an action reward-based system.

In conclusion, the *Supervised Learning* approach and more specifically the sub-category of *Deep Learning*, has been proven to be a successful end-to-end approach in detecting and classifying objects and their affordances. Generally, these approaches aim to classify objects related to kitchen, garden and working tools due to the availability of affordance datasets for these object categories. However, no work on affordances in retail store environments has been conducted to this date. More specifically, there is no dataset publicly available that allows mobile manipulators to identify how to interact with retail store related objects in such environments. This work will investigate the approach of synthetically generating images to overcome this scarcity in data. Subsequently, the data will be used to train an instance segmentation model with the aim of detecting retail store related objects and their affordance classes. In the next chapter, this approach will be explained in more detail.

3 Methods

3.1. Model Choice: Mask R-CNN

3.1.1. Affordances using Mask R-CNN

For an autonomous mobile manipulator in a retail environment, the knowledge of the ability of objects to be grasped by the robot has great value. Products may fall from shelves and end up lying on the floor, which in many aspects poses problems. In terms of customer and employee safety but also in terms of the aesthetics of a retail store interior. A Region-based Convolutional Neural Network for instance segmentation (Mask R-CNN) [14] was used in this work to first localize these objects within an image and then classify these objects into a graspable or pushable affordance class. This approach differs from the conventional application in which objects are classified towards their object type labels after being localized in the image. As an example, we take the classification of apples versus pears which have two different classes in the conventional object type classification, yet share the same graspable affordance label in this work's affordance classification. Do et al. showed in their work on AffordanceNet [9], that the use of Mask R-CNN in combination with affordance datasets resulted in strong results. But also the availability and documentation of the model were reasons to choose Mask R-CNN. This instance segmentation network takes a 2D image as input and is able to predict labels on pixel-level, which results in labelled semantic masks as output. In contrast to bounding box predictions, these masks allow object shapes to be described precisely. As this work aims to classify objects towards their ability to be manipulated, knowledge of their precise shape is of great importance for the actual manipulation step. This subsequent manipulation step of the robot lies outside of the scope of this work. The working principle of Mask R-CNN is elaborated below.

3.1.2. Network Architecture

Mask R-CNN is an instance segmentation network that uses a Region Proposal Network (RPN) [26] to select Regions of Interest (RoIs) from feature maps generated by a backbone network. These initial object localization steps are also referred to as the first stage of Mask R-CNN. Subsequently, the second stage or the head part of the network aims to obtain the object masks, refine the bounding box coordinates and classify objects for every RoI. Below, the process of each part of Mask R-CNN will be explained in more detail and the overall network architecture can be seen in Figure 3.1.

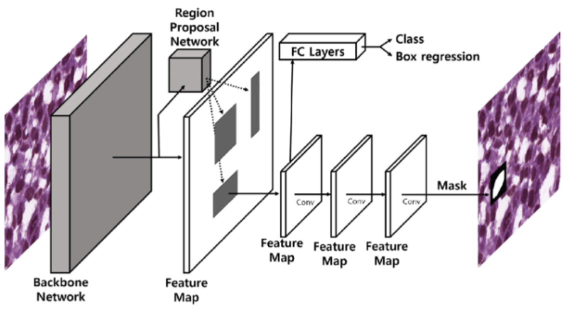


Figure 3.1: Mask R-CNN network architecture. [7]

Backbone: Feature Pyramid Network

The Feature Pyramid Network (FPN) [19] is the backbone network of Mask R-CNN that extracts high-level semantic feature maps at different scales. This is beneficial in detecting objects that differ in size due to the difference in distance from the camera. This backbone may be pre-trained on large datasets like COCO [18] or ImageNet [8], which saves training time by bypassing the part where initial features are learned by the network. The network uses a deep Residual Network with 101 layers (ResNet-101) [13] that processes the input image through convolutional layers for feature map extraction at different pyramid scales (C2-C5 in Figure 3.2). As a consequence of this pathway, the resolution decreases which results in less detailed feature maps and therefore the semantic value of the feature map increases. At every pyramid level, these feature maps are added through a lateral connection to its corresponding up-scaled feature map which contains a higher semantic value. This lateral connection allows the final feature maps (P2-P5 in Figure 3.2) to have value in both the semantic and the resolution aspect at different scales.

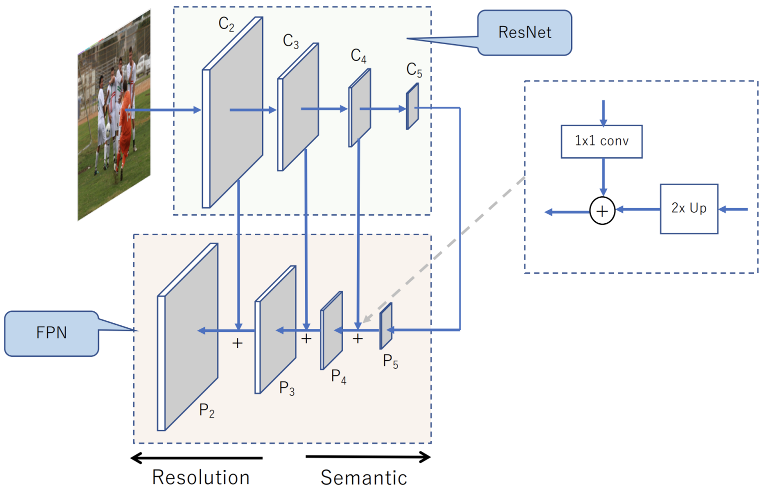


Figure 3.2: Feature Pyramid Network using a deep Residual Network (ResNet) as backbone image feature extractor. [19]

Region Proposal Network

After the set of feature maps have been computed, the Region Proposal Network (RPN) object detector [26] is run on these maps to obtain a set of RoIs from the image. This method applies sliding windows with multiple scales and aspect ratios, known as anchor boxes, on the set of feature maps. Each 2D window will be flattened into a vector, whereafter it is passed through a binary classification layer. This layer outputs two confidence scores, which aim to quantify the confidence of the window containing an object or not. Next to this classification layer, the window will be passed to a regression layer that aims to refine the pre-defined window into a more precise bounding box around the object. For the definition of a bounding box four values are required, the x and y locations and the width and height of the box.

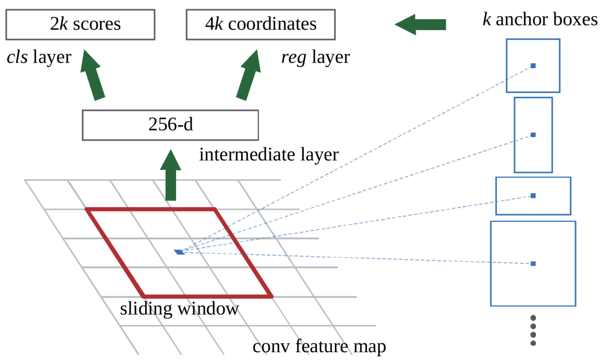


Figure 3.3: Region Proposal Network that applies a sliding window on the obtained multi-scale feature maps. [26]

RoI Align
The bounding boxes (RoIs) obtained from the windows that were classified as an object, are sized relative to their corresponding feature map. The next step in the Mask R-CNN pipeline is to align these RoIs to have identical dimensions. This is required for the head part of the network which will infer the object's mask and class from these fixed-sized RoIs. The size of all the RoIs is reduced to a squared shaped matrix of $n \times n$ using bilinear interpolation, more details on this can be found in the published work of Mask R-CNN [14].

Head Network
After the alignment, the network starts its second stage. All RoIs along with their corresponding feature map are passed to the network's head, which is visualized in Figure 3.4. The upper branch includes fully connected layers to derive the exact bounding box in terms of the x and y locations and the width and height of the box. The upper branch also identifies the class for each RoI in terms of a probability ranging from 0 to 1 for each of the $K+1$ possible classes; K being the desired number of classes plus one background class. The lower part of the head uses a series of convolutional layers to infer the masks corresponding to each class, the mask output will take the shape of $m \times m \times (K+1)$.

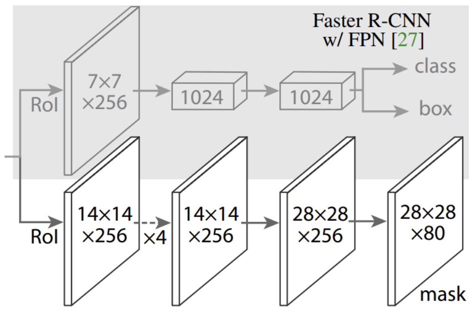


Figure 3.4: Mask R-CNN's head part. [14]

3.1.3. Network Training

Before the network can be used for object affordance detection, it needs to be trained to recognize the desired objects and classify them into affordance labels. For this, a training dataset with annotated ground truth examples is required, which is generated using a data generation approach explained in the next section. The first stage of the network is not trained due to the availability of the pre-trained ResNet-101 on the COCO dataset, which consists of around 2,000,000 annotated objects from 200,000 images. The second stage of the network is trained by adjusting the network weights towards a minimization of the difference between the predicted network output and the annotated ground truth. This difference is commonly referred to as a training *Error* or a *Loss*, which is calculated for the predicted object affordance class, bounding box and mask. These individual *Losses* are combined into a general *Loss* function (Equation 3.1), which will be minimized using a Stochastic Gradient Descent (SGD) optimization approach [29]. This training process will adjust the network weights iteratively for a set of mini-batches obtained from the total training data.

$$L = L_{cls} + L_{box} + L_{mask} \tag{3.1}$$

The *Loss* that relates to an RoI's class (L_{cls}) is defined as the average cross-entropy *Loss* between the predicted class labels and their ground truth labels for all RoI's. A probability of $p_i = 1$ is optimal and must result in a low *Loss* value, which explains the suitability of this logarithmic *Loss* function. The -1 serves to compensate for the resulting negative values for all $p_i < 1$.

$$L_{cls} = -\frac{1}{N_{cls}} \sum_{i=1}^{N_{box}} [p_i \log(\hat{p}_i) + (1-p_i)\log(1-\hat{p}_i)] \tag{3.2}$$

$$i = \text{Index of an RoI in a mini-batch.} \tag{3.3}$$

$$N_{cls} = \text{Normalization term set to the mini-batch size.} \tag{3.4}$$

$$N_{box} = \text{Normalization term set to the number of RoIs in a mini-batch.} \tag{3.5}$$

$$\hat{p}_i = \text{Predicted probability of RoI } i \text{ being an object of a given class.} \tag{3.6}$$

$$p_i = \text{Ground truth binary label of RoI } i \text{ being an object of a given class.} \tag{3.7}$$

The second part of the general *Loss* function (L_{box}) concerns the *Loss* in terms of the predicted RoI bounding boxes compared with their ground truth bounding boxes. The difference between the bounding box coordinates is given to an L_1^{smooth} *Loss* function, which increases the *Loss* value if the deviation from the ground truth is large. The binary p_i parameter aims to only consider correctly detected bounding boxes.

$$L_{box} = \lambda \frac{1}{N_{box}} \sum_{i=1}^{N_{box}} p_i L_1^{smooth}(\hat{t}_i - t_i) \tag{3.8}$$

$$i = \text{Index of an RoI in a mini-batch.} \tag{3.9}$$

$$\lambda = \text{Balancing factor of } L_{box} \text{ for the overall } Loss \text{ function.} \tag{3.10}$$

$$N_{box} = \text{Normalization term set to the number of RoIs in a mini-batch.} \tag{3.11}$$

$$p_i = \text{Ground truth binary label of RoI } i \text{ being an object of a given class.} \tag{3.12}$$

$$\hat{t}_i = (\hat{t}_i^x, \hat{t}_i^y, \hat{t}_i^w, \hat{t}_i^h), \text{ predicted coordinates of RoI } i. \tag{3.13}$$

$$t_i = (t_i^x, t_i^y, t_i^w, t_i^h), \text{ ground truth coordinates of RoI } i. \tag{3.14}$$

$$L_1^{smooth}(t) = \begin{cases} 0.5t^2 & \text{if } |t| < 1 \\ |t| - 0.5 & \text{otherwise} \end{cases} \tag{3.15}$$

The final term (L_{mask}), catches the *Loss* between the ground truth mask pixels and the predicted mask pixels for a given class. The average cross-entropy *Loss* is calculated for the predicted binary mask labels and their corresponding ground truth labels. The *Loss* function calculates the cross-entropy for all pixels in the predicted mask, the more similarity between the masks the lower the *Loss* value.

$$L_{mask} = -\frac{1}{m^2} \sum_{i,j=1}^{m} [y_{ij} \log(\hat{y}_{ij}^u) + (1 - y_{ij}) \log(1 - \hat{y}_{ij}^u)] \tag{3.16}$$

$$u = \text{Ground truth class label of RoI} \tag{3.17}$$

$$m = \text{Mask size in pixels.} \tag{3.18}$$

$$i, j = \text{Pixel coordinates.} \tag{3.19}$$

$$\hat{y}_{ij}^u = \text{Predicted binary pixel label of mask for ground truth class.} \tag{3.20}$$

$$y_{ij} = \text{Ground truth binary pixel label of ground truth mask.} \tag{3.21}$$

3.2. Data Generation

3.2.1. Synthetic Image Data

As explained earlier in this report, the choice of a *Supervised Learning*-based approach requires a significant amount of annotated images to train the Mask R-CNN model. The work of Do et al. in AffordanceNet [9], used a totality of 8,835 annotated images for 9 different affordance classes related to kitchen and working tools. Given that AffordanceNet performed well using around 1,000 training images per class, this work aims to generate a dataset with a minimum of 2,000 annotated training images for 2 classes, i.e. *graspable* and *pushable*. To avoid the intensive human labour of creating and annotating such an amount of images, the choice was made to generate a so-called synthetic image dataset for training. This automated data generation process is described next.

Image and Mask Generation Pipeline

The process of generating synthetic images and their corresponding masks is implemented in a Blender script that is able to iteratively load and render 3D models in a custom environment. The pipeline of the Blender script created for this work is visualized using a simplified pseudocode-style structure in Algorithm 1. In short, the script takes as input a list of 3D models and a list of background images captured in a retail store environment. It loads a 3D model, randomizes parameters, adds realistic lighting conditions and finally renders and saves the synthetic image and mask, which is done iteratively for all 3D models. This process is repeated N times, depending on the required size of the dataset. Below, the data generation script will be elaborated in more detail.

Algorithm 1 Blender data generation script in pseudocode.

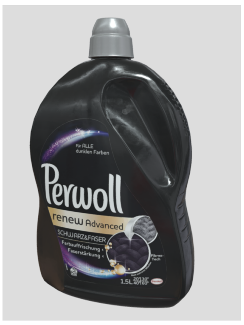
```
Input: list of 3D models, list of background images
Output: synthetic images, semantic masks
for $i = 1, 2, \ldots, N$ do                                  ▷ Variable to control resulting dataset size
    for $model$ in $list\_models$ do                          ▷ Loop through all 3D models
        $env = new\_environment()$                            ▷ Create new Blender environment

        $background\_image = random()$                        ▷ Randomize parameters for environment
        $orientation = random()$
        $location = random()$

        $env.model = load\_model(model, location, orientation)$
        $env.light = add\_lighting()$
        $env.plane = add\_plane(location)$
        $env.cam = add\_camera()$

        $render(env, background\_image)$                      ▷ Render and save image + mask
    end for
end for
```

Input and Output
The 3D models are configured using the *glTF* file format and contain an added file called *height_estimate.txt*, which contains an integer number that represents an estimate of the length of the object along its largest axis. This is required to correctly define the 3D model's relative size towards the other models. The list of images captured in a retail store environment will serve as a background image for the synthetic image. The final affordance dataset was created using 121 3D models of commonly found objects in supermarkets, varying from groceries to cardboard boxes and 311 background images.



(a) 3D Model

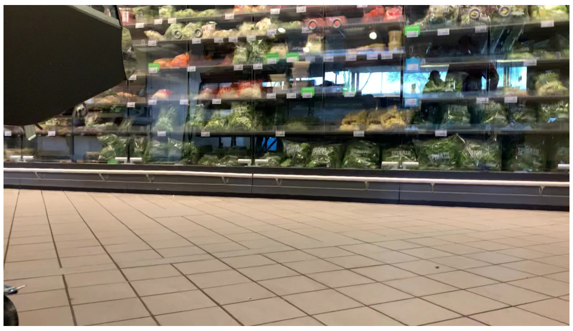
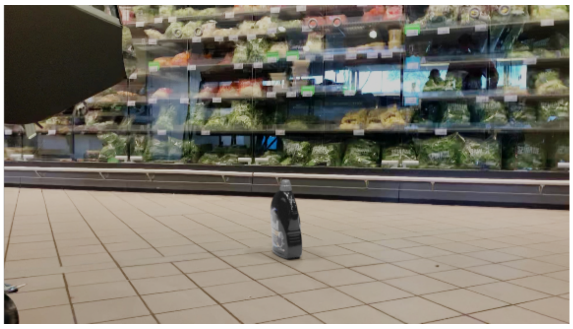

(b) Background image

Figure 3.5: An example of a possible set of **inputs** to the Blender script.


(a) Synthetic image

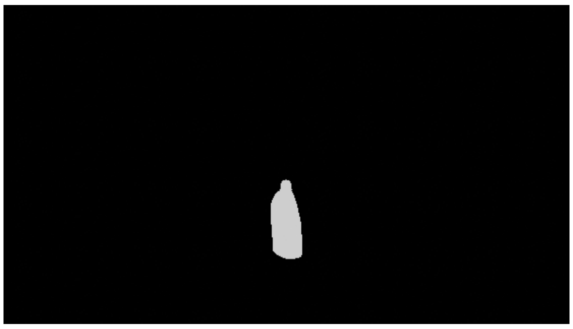

(b) Mask

Figure 3.6: The resulting **outputs** from the example inputs shown in Figure 3.5.

Randomizing Environmental Parameters

For the generation of the affordance dataset, the requirement of variety in images is an important aspect. To obtain such variety, randomized parameters are initialized. First, for every iteration a random background image is chosen from the list of background images, after which the location and orientation of the 3D model are randomized. The horizontal location of the object is limited to the area in the centre of the image, due to the nature of the background images. These generally have obstructions, like shelves, at both sides of the image. The vertical location is randomly chosen from a range of numbers that still allow for a realistic setting with respect to the object sizes compared to the background image. Orientations with respect to the x- and y-axis are limited to increments of 90°, to maintain realistic object orientation w.r.t. gravity. The possible sets of orientations are mathematically defined below.

$$\theta_x \in \mathbb{Z} \mid \theta_x = \{0, 90, \dots, 270\} \tag{3.22}$$
$$\theta_y \in \mathbb{Z} \mid \theta_y = \{0, 90, \dots, 270\} \tag{3.23}$$
$$\theta_z \in \mathbb{Z} \mid 0 \leq \theta_z < 360 \tag{3.24}$$

3D Model on Plane

After the initialization of the parameters, the 3D model corresponding to the current iteration is loaded into the newly created Blender environment, given the initialized location and orientation. To enhance the synthetic image in terms of realism, lighting conditions are added to obtain reflections and shadows. To simulate the indoor lighting conditions of a retail store, a 2x2 grid of virtual light spots are added around the object at a ceiling height of 3 meters. A horizontal plane is added to the bottom of the 3D model to replicate the retail store's floor surface on which the object is located and on which the object's shadow will appear. This Blender environment can be seen in Figure 3.7.

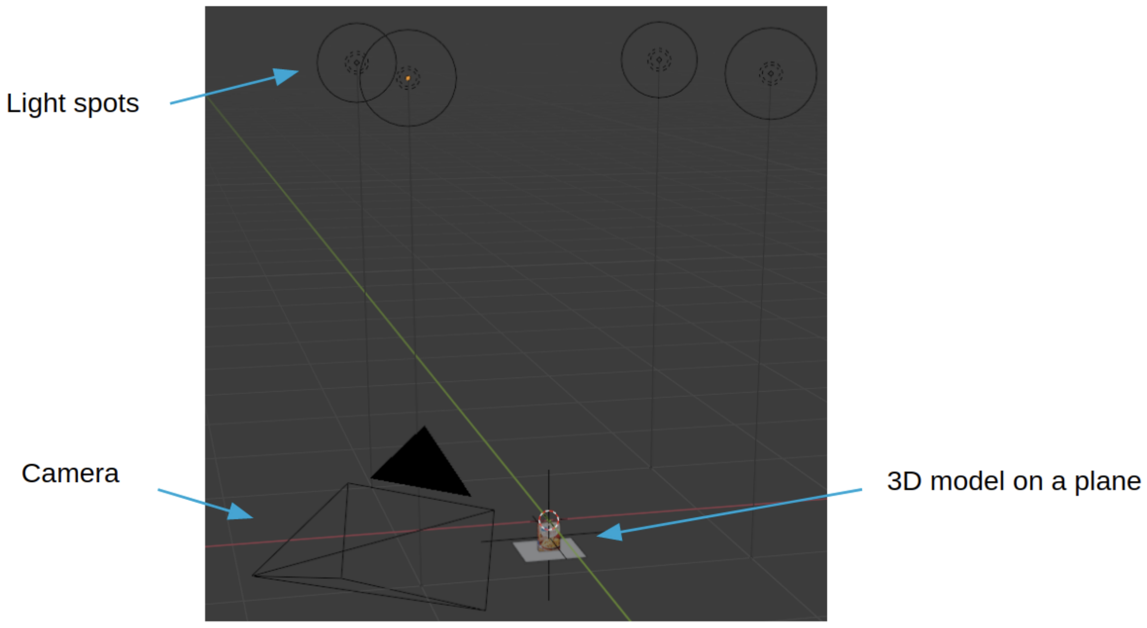


Figure 3.7: Example of the Blender environment created by the script and visualized in the Blender GUI.

Rendering and Saving

One of the final steps of the script is the addition of a virtual camera, from which the synthetic image will be captured. This camera with a focal length of $50mm$ is located at a fixed position of $(2, 2, \frac{1}{2})m$ with respect to the 3D object and is pointing towards the 3D object. The synthetic image can now be rendered using the camera's perspective of the environment, together with the chosen background image that is set to be rendered behind the object. Next to this synthetic image, a binary mask is rendered, which saves pixels corresponding to the object as HIGH and the background pixels to LOW. This allows for semantic distinction between the object and its background on pixel-level accuracy. Some examples of the output images and masks are shown below.

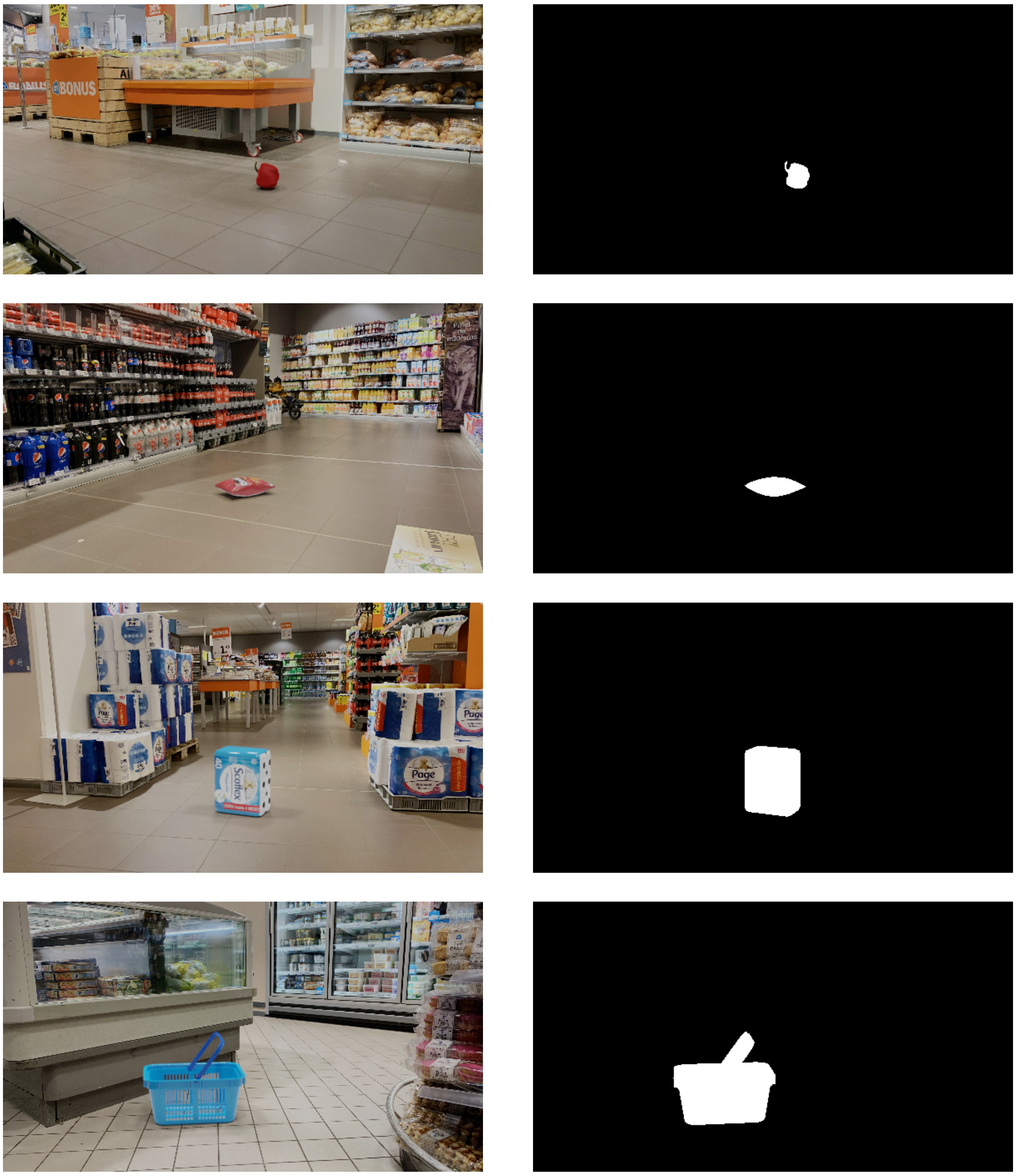


Figure 3.8: Examples of the Blender data generation script output with the synthetic images on the left and the corresponding masks on the right.

Data Labeling

For this work, the choice was made to distinguish objects between *graspable* and *pushable* affordance classes, yet the implementation of other affordance classes might also be possible. In terms of applications for mobile manipulation, these specific affordance classes are of great importance, think of obstacle clearance or even product restocking. The distinction between the two object affordance classes for the synthetic image training dataset is made through the manual labelling of the image masks. The classes are distinguished by the definition of a human interpretation of an object being able to be grasped using a single hand. If an object does not satisfy this condition, then it is assigned the *pushable* class. This decision was made with the purpose of having a general distinction between *graspable* and *pushable* objects, instead of focussing on a specific manipulator gripper. However, this assumption comes with the limitation of having subjectivity in the affordance labels, as each person has another interpretation of an object being *graspable* by a single hand or not. Therefore, a grey area between the two classes is expected, in which an object may be considered to be *graspable* or *pushable* depending on a person's interpretation of the scenario. This aspect will be discussed later in this report. Some examples of *graspable* and *pushable* labelled images are displayed in Figure 3.9.

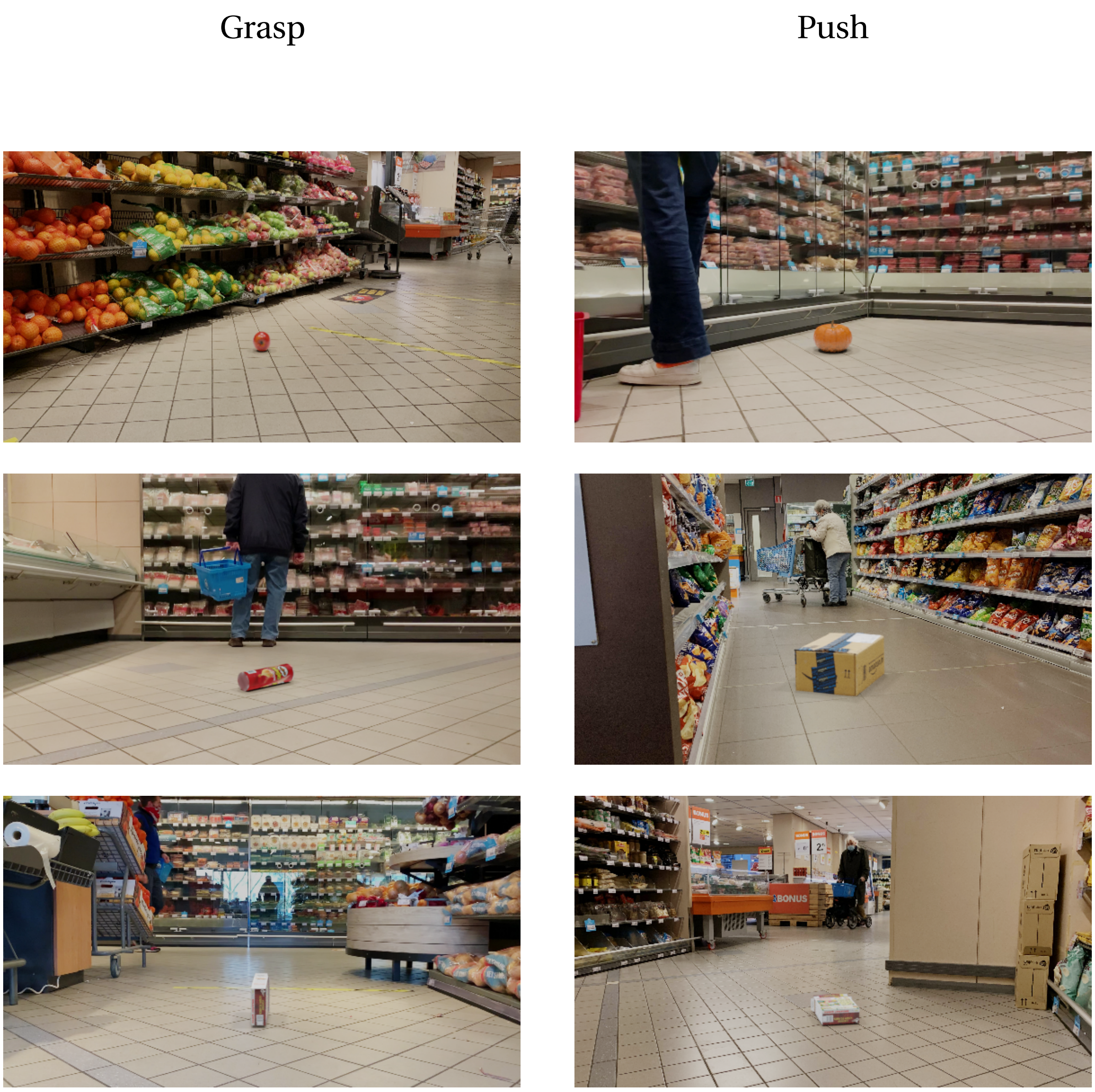
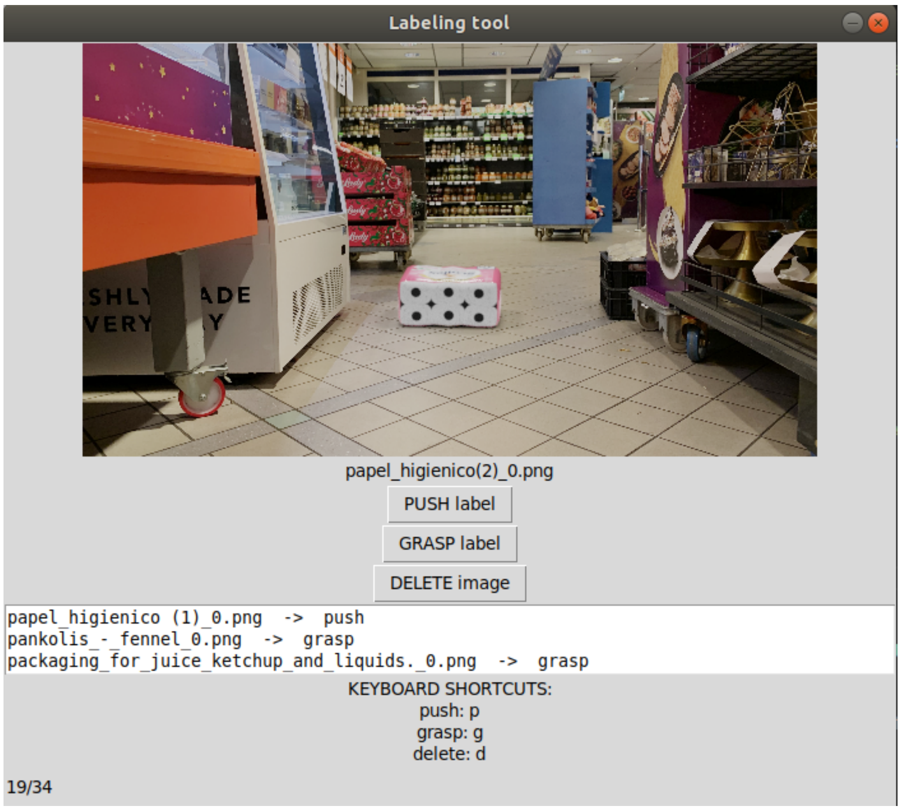


Figure 3.9: Three synthetic images labelled as *graspable* can be seen in the *left column*. In the *right column*, objects with a *pushable* affordance label are displayed. Note that the same object may be labelled differently according to the scenario (see *last row*).

Another notable difference in classifying affordance labels compared to object type labels can be observed in the last row of the examples. A particular object can be labelled as either of the affordance classes depending on the situation, this is generally caused by the orientation of the object. As an example, such a situation was added to the last row of Figure 3.9. The box of cereals can easily be grasped when it is positioned in an upright orientation, but cannot if it is lying flat on the floor and thus only affords to be pushed.



```
papel_higienico (1)_0.png  ->  push
pankolis_-_fennel_0.png  ->  grasp
packaging_for_juice_ketchup_and_liquids._0.png  ->  grasp
```

KEYBOARD SHORTCUTS:
push: p
grasp: g
delete: d

19/34

Figure 3.10: Affordance labeling tool for the synthetically generated images.

The ability to process data fast and efficient is key when labelling large amounts of images. For this reason, a labelling tool application was created to allow for fast data labelling. The Graphical User Interface (GUI) can be seen in Figure 3.10. After selecting a directory that contains synthetic images with their corresponding masks, the application will iterate through the images. At every displayed image, two label options are given to the user as well as a delete button that removes the current image and its mask from the directory. When a label button is pressed, a text file containing the label string will be generated and saved in the directory. This results in three files per labelled image; the synthetic image, the binary mask and the file containing the label. These three files are now ready to be loaded into the Mask R-CNN dataset structure.

Final Synthetic Image Dataset
After the synthetic data generation and labelling, a dataset of purely synthetic images was obtained that is ready to be used for training. This dataset was generated using 121 3D models of objects that are commonly found in supermarkets, which were obtained from 3D model providers like *Sketchfab* [6] and *TurboSquid* [1]. The 311 acquired background images were manually taken in two different supermarkets. The Blender script generated a final dataset that contains 3237 synthetic images with their corresponding masks and affordance label files. In Figures 3.8 and 3.9 various examples of these images are visualized.

3.2.2. Real Image Data

These synthetic images may be a good approach to obtain a large amount of data for the training stage of the model. However, the model may learn features that are unique to these synthetic images and do not occur in real images. Therefore, the performance of the model must be evaluated on real image data to replicate a real scenario as accurately as possible. This data was manually obtained by making images of scenarios with one or more objects on the ground. These images were obtained in three different supermarkets to obtain some variation in objects and backgrounds. The required masks and corresponding labels were annotated with the use of the Visual Geometry Group's (VGG) image annotator [10]. The final real image dataset consists of 204 images and will be used to validate the model. Some examples from the dataset are shown below in Figure 3.11.

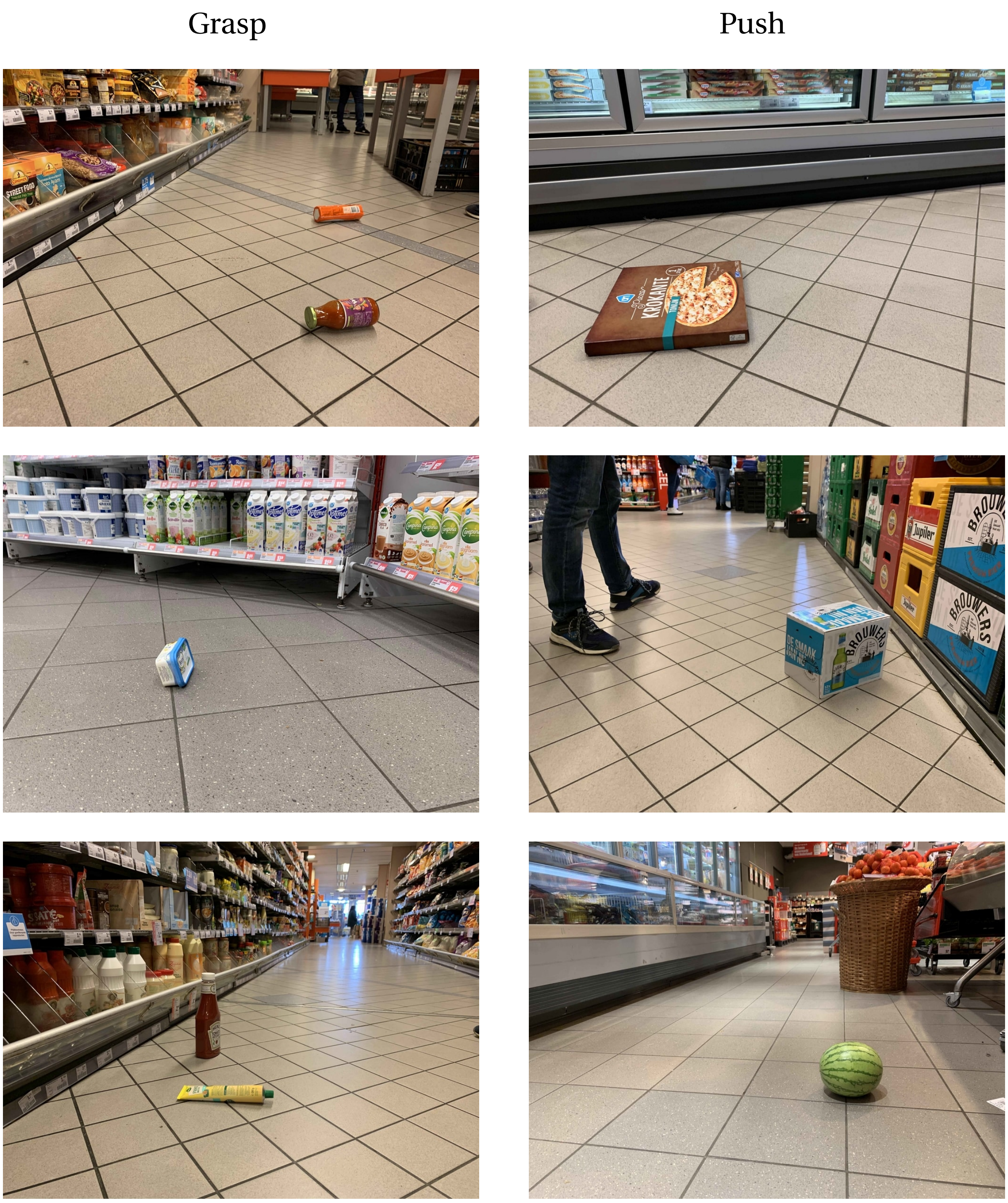


Figure 3.11: Examples of images from the real image dataset. In the *left column*, *graspable* objects are displayed and *pushable* objects are located in the *right column*. Note that the real image dataset can contain images with multiple objects in a single image, unlike in the synthetically generated dataset.

3.3. Model Setup

3.3.1. Default Model Parameters

The hyperparameters of a neural network determine the learning process of the model. For this work, most hyperparameters were set to the default values obtained from the Mask R-CNN repository [4]. The optimization of the performance of the model is not a goal of this work, but a performance evaluation of the generated affordance dataset is. Therefore the default model parameters are assumed to perform sufficiently well to compare performance between the different datasets. The parameter configuration is listed in Table 3.1.

Parameters	Values		
BACKBONE	resnet101	MASK_POOL_SIZE	14
BACKBONE_STRIDES	[4, 8, 16, 32, 64]	MASK_SHAPE	[28, 28]
BATCH_SIZE	**28**	MAX_GT_INSTANCES	100
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]	MEAN_PIXEL	[123.7 116.8 103.9]
COMPUTE_BACKBONE_SHAPE	None	MINI_MASK_SHAPE	(56, 56)
DETECTION_MAX_INSTANCES	100	**NAME**	**affordances**
DETECTION_MIN_CONFIDENCE	0.5	**NUM_CLASSES**	**3**
DETECTION_NMS_THRESHOLD	0.3	**NUM_EPOCHS**	**28**
FPN_CLASSIF_FC_LAYERS_SIZE	1024	POOL_SIZE	7
GPU_COUNT	1	POST_NMS_ROIS_INFERENCE	1000
GRADIENT_CLIP_NORM	5.0	POST_NMS_ROIS_TRAINING	2000
IMAGES_PER_GPU	1	PRE_NMS_LIMIT	6000
IMAGE_CHANNEL_COUNT	3	ROI_POSITIVE_RATIO	0.33
IMAGE_MAX_DIM	1024	RPN_ANCHOR_RATIOS	[0.5, 1, 2]
IMAGE_META_SIZE	15	RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
IMAGE_MIN_DIM	800	RPN_ANCHOR_STRIDE	1
IMAGE_MIN_SCALE	0	RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
IMAGE_RESIZE_MODE	square	RPN_NMS_THRESHOLD	0.7
IMAGE_SHAPE	[1024 1024 3]	RPN_TRAIN_ANCHORS_PER_IMAGE	256
INTERSECTION_OVER_UNION	**0.5**	**STEPS_PER_EPOCH**	**100**
LEARNING_MOMENTUM	0.9	TOP_DOWN_PYRAMID_SIZE	256
LEARNING_RATE	0.001	TRAIN_BN	False
LOSS_WEIGHTS		TRAIN_ROIS_PER_IMAGE	200
rpn_class_loss	1.0	USE_MINI_MASK	True
rpn_bbox_loss	1.0	USE_RPN_ROIS	True
mrcnn_class_loss	1.0	VALIDATION_STEPS	50
mrcnn_bbox_loss	1.0	WEIGHT_DECAY	0.0001
mrcnn_mask_loss	1.0		

Table 3.1: Overview of the parameter configuration for the Mask R-CNN model. This configuration is based on the default parameters from the Mask R-CNN repository and includes parameter changes shown in bold, which will be discussed in the subsequent section.

3.3.2. Tuned Model Parameters

Batch size, Steps per Epoch

When training a neural network like Mask R-CNN, the training data passes multiple times through the network while updating its weights according to an optimization process of a predefined *Loss* function. This training process was explained in section 3.1.3. A single cycle through the whole training dataset is referred to as an *Epoch*. Due to the size of the training dataset, one *Epoch* is too large to be processes at once and therefore is divided into several smaller mini-batches with a certain *Batch Size*. Finally, the *Steps per Epoch* or the number of *Iterations* tell us how many batches are needed to finish a single *Epoch*.

$$\textit{Steps per Epoch} = \frac{\textit{Training Dataset Size}}{\textit{Batch Size}} \tag{3.25}$$

$$\textit{100} = \frac{\textit{2800}}{\textit{Batch Size}} \tag{3.26}$$

$$\hookrightarrow \textit{Batch Size} = 28 \tag{3.27}$$

Due to the relatively small size of ±2800 images in the training dataset, the *Steps per Epoch* were set to 100 steps, as was done in a similarly sized training dataset example from the Mask R-CNN repository [4]. This choice results in a *Batch Size* of 28 samples per batch.

Number of Classes

The number of classes for this network equals the amount of affordance classes plus one. This is due to the background, which is configured as a class as well. In this case the total number of classes is 3, given the *graspable* and *pushable* affordances classes along with the background class.

Number of Epochs

The number of *Epochs* is a critical parameter to tune for the avoidance of network under- or overfitting. When training with a low amount of *Epochs*, the network will not capture the pattern from the training data enough to predict reliable results and therefore the model will experience underfitting. On the other hand, when training with too many *Epochs* the network will learn patterns that are too specific to the training data, consequently experiencing overfitting. To obtain the optimal amount of *Epochs*, which is positioned between under- and overfitting, the graph of the *Loss* vs. the number of *Epochs* will be examined, which typically looks like the example in Figure 3.12. During the training stage, the *Loss* parameter quantifies the error between the predicted output of the network and the ground truth. For Mask R-CNN, the *Loss* parameter covers the overall *Loss* in terms of the network's affordance class, bounding box and mask predictions, as discussed earlier in this chapter (3.1.3). It is calculated during the training stage for both the train and validation sets after every *Epoch*. The optimal amount of *Epochs* is referred to as the *Early Stopping* point in this example and it occurs when the *Loss* of the validation data reaches a global minimum.

With this theory in mind, the optimal amount of *Epochs* for this work's affordance model will be determined by training the model following two scenarios. In both scenarios the network is trained on the same split of the synthetic image dataset, resulting in an 87%/13% distribution of training and validation data. The training data thus consists of 2816 images. The first validation set consists of the validation set corresponding to the previously mentioned split from the synthetic image dataset resulting in 421 images. The second validation set will fully consist of the real image dataset, which contains 204 images. The reason to train the network according to these two scenarios is that synthetic image data may perform differently than real image data. Therefore, both should be considered when determining the optimal amount of *Epochs*. The *Loss* vs. *Epochs* graph will be plotted, after which the optimal number of *Epochs* will be determined by visually finding the global minimum.

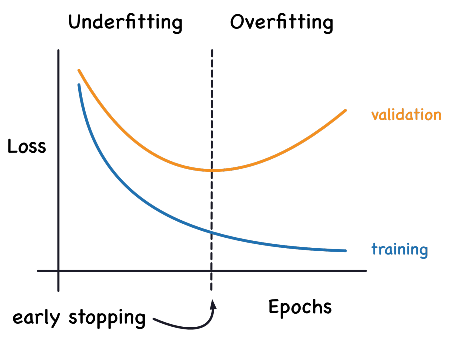


Figure 3.12: Visualization of the *Loss* plotted against the number of training *Epochs*. [5]

3.4. Model Evaluation

3.4.1. Evaluation Metrics

In summary, the following evaluation metrics will be used for the evaluation of the model performance on different validation sets.

Metrics
True Positives, False Positives, False Negatives
Precision
Recall
(mean) Average Precision

Table 3.2: Overview of the used metrics for the model evaluation.

In the next part, these metrics are explained in more detail. For this object affordance detection model the pipeline can be globally split up into two parts. First, the correct RoIs are obtained, whereafter those RoIs are classified into an affordance class. To evaluate the performance of the trained network, several commonly used metrics for object detection evaluation are introduced. The first step in evaluating the model is done by calculating the amount of *True/False Positives* and *True/False Negatives*. *True Positive* and *True Negative* being the parameters that correlate to the correct predictions considering the ground truth, either positive or negative detections, respectively. On the other hand, the parameters *False Positive* and *False Negative* aim to quantify the number of incorrect predictions that are either positive or negative, respectively. The sum of the *True Positives* and *False Negatives* result in the total amount of ground truth objects. More concisely, the parameters are defined as follows.

- **TP** = Positive detection with a correctly detected RoI and affordance class.
- **FP** = Positive detection with an incorrectly detected RoI and/or affordance class.
- **FN** = Objects that were not detected, but should have been.
- **TN** = Objects that were not detected and should not have been. *(Since this holds for the whole background of the image, this parameter is generally not considered for the evaluation of object detection methods.)*

An important model parameter that influences the number of *Positives* and *Negatives* is the *Minimum Level of Confidence* of the network, which is referred to as the MINIMUM_DETECTION_CONFIDENCE in Table 3.1. This is a threshold that determines the minimum *Level of Confidence* for the resulting detections to be accepted. The *Level of Confidence* is defined as the probability that a detection falls into an assigned class. According to this *Minimum Level of Confidence*, a detection is labelled as *Positive* when its *Level of Confidence* is positioned above the threshold and *Negative* otherwise.

The *Intersection over Union* (*IoU*) is a geometric measure used to compare the area of the predicted mask with respect to the ground truth mask and thus decides whether an RoI is correctly detected or not. This metric is computed by dividing the *Area of Overlap* by the *Area of Union*. Typically, this metric is used to distinguish predictions in terms of being *True Positive* or *False Positive*, by putting a minimum threshold on the metric.

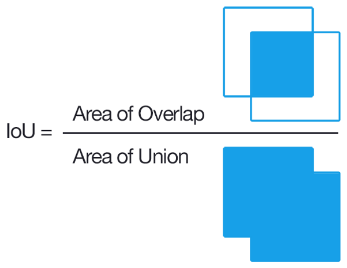


Figure 3.13: Visual definition of the *Intersection of Union* metric. [2]

Next, the metrics of *Precision* and *Recall* will be defined. The *Precision* metric quantifies the proportion of all positive predictions that are actually true based on the ground truth. The proportion of all ground truth objects that were identified correctly, is referred to as the *Recall*.

$$Precision = \frac{TP}{TP + FP} \tag{3.28}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.29}$$

These metrics are used for the evaluation of test data at a specific minimum *Level of Confidence*, but can be plotted in a *Precision* vs. *Recall* curve for an overall model performance comparison. This is done by calculating the *Precision* and *Recall* values across all minimum *Level of Confidence* thresholds. Also, this curve serves as a graphical explanation for the *Average Precision (AP)*, which is calculated using a smoothed version of the curve (Figure 3.14) and will be explained below.

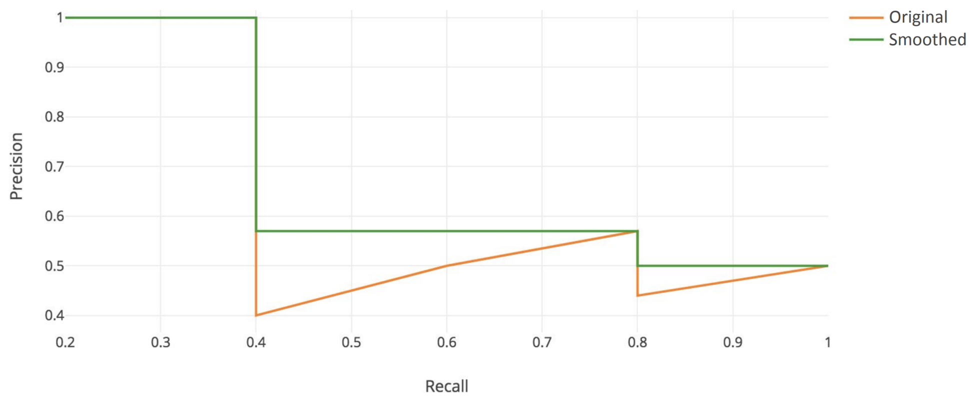


Figure 3.14: Example of a original *Precision* vs. *Recall* curve and it's interpolated version. [3]

Using the parameters explained above, the *Average Precision* (*AP*) is calculated for a given *IoU* threshold and class by computing the *Area Under Curve* (*AUC*) of the *Precision* vs. *Recall* curve, given a validation set. For convenience, the original noisy pattern of the curve is transformed into a more smooth version, which is used to calculate the *AUC*. The *mean Average Precision* (*mAP*) is obtained by averaging all *APs* over the number of classes, which summarizes the overall performance of the model in a single value. This can be mathematically described as the following.

$$mAP_{IoU} = \frac{\sum_{n=0}^{N} AP_{IoU}}{N} \tag{3.30}$$

$$AP_{IoU} = \int_0^1 p(r)dr \tag{3.31}$$

$$p, r = \textit{Precision, Recall @ IoU} \tag{3.32}$$

$$N = \textit{Number of Classes} \tag{3.33}$$

3.4.2. Evaluation on Synthetic and Real Image Data

The model was trained purely on synthetically generated images to overcome the problem of the acquisition of large amounts of real image data for training. However, for manipulator robots to be able to use this model it requires to perform on real image data. A performance comparison is made by comparing the evaluation metrics for both the synthetic and real image validation set. Also, a performance analysis on the first step of the network, i.e. the object localization step, will be conducted. The number of *True Positive*, *False Positive* and *False Negative* mask detections, regardless of the assigned affordance class, will be counted. This enables a comparison between the performance of the object localization part and the object classification part. For all calculations of the number of *True Positives*, *False Positives* and *False Negatives*, the minimum *Confidence Level* was set to 0.5 to allow comparisons between these metrics for different tests. The overall differences in performance between the real and synthetic validation set will be analyzed by introducing hypotheses for possible causes of these differences. For each hypothesis, an experiment will be conducted, which will yield an acceptance or rejection of the corresponding hypothesis.

3.5. Summary

In short, this work will investigate the use of synthetically generated image data for object affordance classification in retail environments. More specifically, obstructions that are found on the floor in a supermarket will be detected and classified to their possibility to be moved by a mobile manipulator. The obstructions will either have a *pushable* or *graspable* affordance label, depending on whether an object is able to be grasped by a single human hand. In this chapter, the chosen instance segmentation model Mask R-CNN was introduced. Several reasons for the choice of this network were given, after which the network architecture was explained in more detail. Also, the *Loss* function that is minimized during the training stage was enlightened. Next, the synthetic data generation approach was explained in detail along with several generated image and mask examples. In addition, the acquisition of annotated real image data for the evaluation of the trained model was shortly described. Thirdly, the model setup in terms of the default and tuned parameters were listed. Finally, the evaluation process of the model that will be trained using synthetically generated images, was described.

4 Results

4.1. Model Setup

4.1.1. Number of Epochs

In Figure 4.1, the results of the two scenarios that were described in the previous chapter are visualized. In short, both scenarios were trained on the same synthetic image dataset. However, one scenario was validated using synthetic images and the other using real images. In the next part, these scenarios will be referred to as synthetic and real scenarios, respectively.

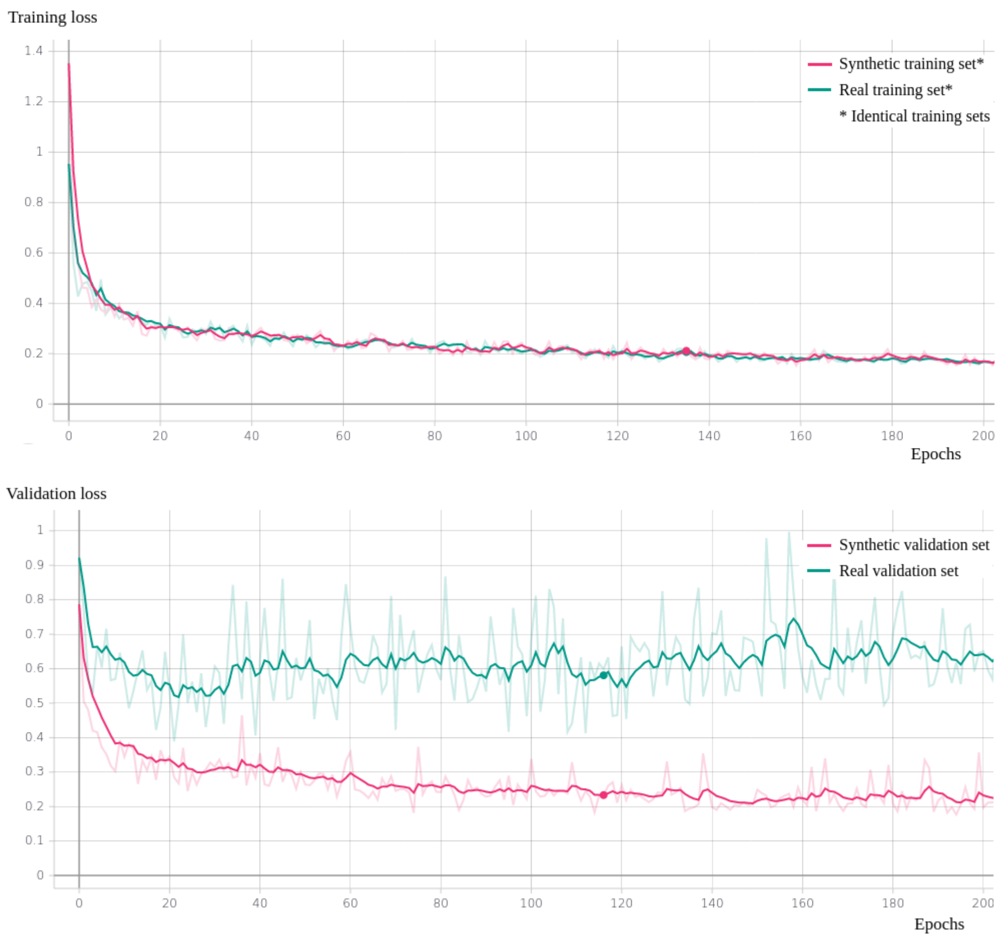


Figure 4.1: The top and bottom figure visualize the training and validation *Loss* vs. the number of *Epochs*, respectively. The pink-coloured lines correspond to the synthetic image validation scenario and the green-coloured lines to the real image validation scenario. The bold lines represent a smoothed approximation of the faded lines.

An initial observation of these graphs shows a large number of fluctuations in the validation *Loss*. The actual *Loss* is plotted as the faded lines, and a bold line visualizes a smoothed approximation. Also, higher fluctuations are found in the validation *Loss* for the real validation set compared to the synthetic validation set. Further, nearly identical decreasing lines are observed in the training data *Loss*, which is expected as the training data is identical in this case. At approximately 20 *Epochs*, the validation *Loss* curves start to decrease at a lower rate and tend to start converging to a *Loss* of 0.55 and 0.3 for the real and synthetic image validation sets, respectively. For the validation *Loss*, considering the smoothed line, the global minimum of the real validation set's curve occurs at 28 *Epochs*, whereafter the *Loss* slowly starts to increase. This convergence is not as noticeable for the synthetic validation set, which slightly keeps on decreasing over the number of *Epochs*, yet a local minimum is found around 28 *Epochs*. These minima point out that the optimal amount of *Epochs* is equal to 28, which is used for all further model training purposes.

4.2. Model Evaluation on Synthetic and Real Image Data

4.2.1. Object Localization Performance

In this section, the model performance in terms of localizing the object in the image was evaluated. More specifically, the classified affordance label was temporally ignored to determine the extent to which the network can correctly localize objects of interest and infer their shape using a mask representation. In Table 4.1 the results in terms of *True Positives*, *False Positives* and *False Negatives* are listed for both the synthetic and real image validation sets. The metrics do not take into account an object's class, thus a predicted mask that has significant overlap with the ground truth mask ($IoU > 0.5$) is captured in the *True Positives*, whereas detected masks without significant overlap are quantified in the *False Positives*. Further, the *False Negatives* capture the number of undetected objects. An initial observation of these results shows that the trained network obtains a perfect *Recall* score of 100%. In other words, it can detect all objects in the synthetic validation set for a minimum *Confidence Level* of 0.5. In the real image validation set, a total of 31 out of 391 (360+31) ground truth objects were missed, resulting in a *Recall* of 92.1%. Further, the results indicate that 55.0% and 68.0% of the predicted masks concern true detections for the synthetic and real image validation sets, respectively.

Confidence level > 0.5	**Synthetic**	**Real**
TP_{loc}	465	360
FP_{loc}	380	170
FN_{loc}	0	31
Precision [%]	55.0	68.0
Recall [%]	100	92.1

Table 4.1: An overview of the number of *True Positives*, *False Positives* and *False Negatives*, along with the resulting *Precision* and *Recall* in terms of the localization of objects in the synthetic and real image validation sets.

4.2.2. Overall Performance

Next to global minima, there are other observations to be made from the *Loss* vs. *Epochs* graph plotted in Figure 4.1. First, the *Loss* of the real image validation is higher than the *Loss* of the synthetic scenario at any *Epoch*. This translates to a higher error for real image data when comparing the network prediction with the ground truth. Furthermore, a larger amount of fluctuations is found in the real image validation data *Loss*. The categorizing of the detections into *True Positive*, *False Positive* and *False Negative* detections results in the distribution listed in Table 4.2. It covers all obtained detections except for detections that have a *Confidence Level* lower than 0.5. The first notable difference between the synthetic and real validation sets is the significant difference in total *Recall*. This difference is caused by the grasp class, which scores strongly for the synthetic image validation set and notably less in the real image validation set. The *Recall* values for the push class are lower for both validation sets but almost have identical values. The *Precision* is considerably lower, which is expected due to the high amount of detections that also include *False Positives*.

A comparison between the object localization performance (Table 4.1) and the overall performance (Table 4.2), results in the performance in terms of the object classification. Considerably more *False Negative* or missed detections are observed in the second table due to wrongly classified affordance labels. For the synthetic image validation set, all 23 *False Negatives* are caused by misclassification and for the real image validation set, this number has doubled due to misclassification. Further, the distribution of *True* and *False Positives* shifted towards the *False Positives* having the overhand caused by misclassification.

Confidence level > 0.5	**Synthetic**			**Real**		
	push	grasp	**total**	push	grasp	**total**
TP	63	335	**398**	100	138	**238**
FP	134	313	**447**	147	145	**292**
FN	20	3	**23**	32	31	**63**
Precision [%]	32.0	51.7	**47.1**	40.5	48.8	**44.9**
Recall [%]	75.9	99.1	**94.5**	75.8	81.7	**79.1**

Table 4.2: An overview of the amount of *True Positives*, *False Positives* and *False Negatives* per affordance class, along with the resulting *Precision* and *Recall*. The synthetic and real image validation sets are evaluated in terms of both the localization and classification performance, resulting in the overall model performance.

The difference in performance is confirmed by comparing the *mAP* values for both validation sets, where the synthetic validation set outperforms the real validation set with a difference of 19.4% in *mAP* (Table 4.3). When comparing the individual classes, a significant performance difference is found in the grasp affordance class. The corresponding curve in Figure 4.2, confirms this observation and shows that for this scenario, the *Precision* maintains a higher value than others at higher *Recall* values.

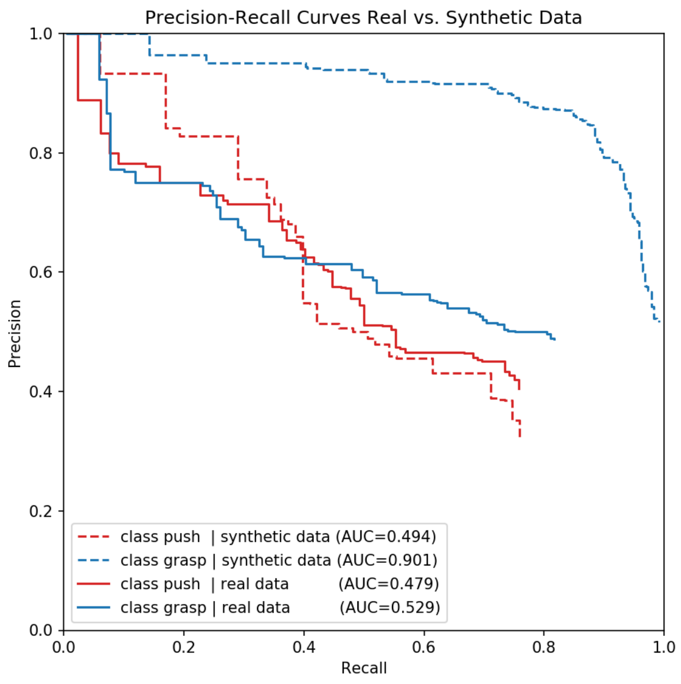


Figure 4.2: A comparison of the *Precision-Recall* curves for synthetic and real image validation sets, accompanied with their corresponding *Area Under Curve* value. These lines show the ratio between the *Precision* and *Recall* values at different *Confidence Level* thresholds.

	Synthetic			**Real**		
	push	grasp	**mean**	push	grasp	**mean**
$AP_{0.5}$ *[%]*	49.4	90.1	**69.8**	47.9	52.9	**50.4**

Table 4.3: A comparison of the model's overall performance for synthetic and real image validation sets, in terms of *Average Precision* at an *IoU* of 0.50.

4.2.3. Performance Analysis

Having analyzed the object localization and overall performance of the model for both synthetic and real validation sets, the next part will focus on particular aspects that influence this performance. Two hypotheses were set to analyze observed differences in performance, with the aim of highlighting areas in which the approach may be improved. First, the hypothesis in question will be explained, after which an experiment is set up to evaluate the ability to reject or accept this hypothesis. The following hypotheses were set up.

1. The subjective nature of affordances has an influence on the ability to generalize into pushable and graspable objects, which causes duplicate detections.

2. The relatively lower performance on real images compared to synthetic images is partly due to the more complex scenarios found in real images, i.e. objects that are located against shelves, located close to another or stacked on top of each other.

Hypothesis 1
The subjective nature of affordances has an influence on the ability to generalize into pushable and graspable objects, which causes duplicate detections.

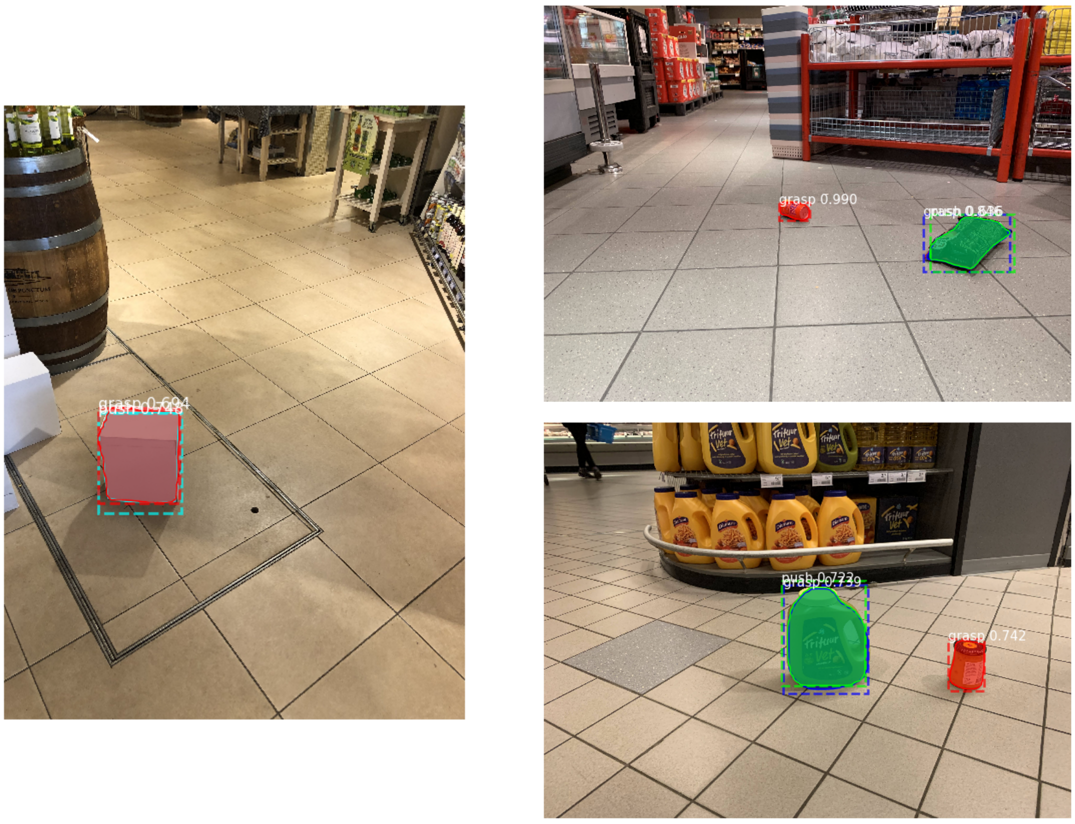


Figure 4.3: Three examples of duplicate detections that have overlap with the ground truth object mask.

Experiment
In contrast to object type classification, in which a detection generally clearly fits a certain class, affordance classification does not always have this clear distinction between classes. In this work, the line between an object being pushable or graspable lies in the human interpretation of that action being possible.
To tackle this hypothesis, the problem and its influence on the performance evaluation will be enlightened by counting the resulting amount of duplicate detections in the validation sets. Duplicate detections are defined as two detections that have significant mask overlap ($IoU > 0.5$) and have been classified as different

affordance classes. Next to the total number of duplicates, the number of detections that significantly overlap with a ground truth object will be counted as well. This latter number of duplicates is directly related to the amount of *True Positive* and *False Positive* detections. One of the detections must be correct while the other is incorrect. On the other hand, duplicates that have no overlap with ground truth masks account for two *False Positives*. These numbers for duplicate detections and their fractions of the total amount of *True Positives* and *False Positives* will show the impact of duplicates on the model's overall performance. To get a better understanding of these duplicate detections, several examples are shown in Figure 4.3.

Results

The total count of duplicate detections for both the synthetic and real validation sets are listed in Table 4.4. The total count is split up into duplicates that have significant overlap with the ground truth mask and duplicates that do not have significant overlap with the ground truth mask. As mentioned, the duplicates directly affect the number of *True Positives* and *False Positives* in the overall model performance. The fraction of the total amount of *True* and *False Positives* caused by these duplicates is listed in Table 4.4 as well.

Confidence level > 0.5	**Synthetic**	**Real**
Total count	53	92
Ground truth overlap	44	89
No ground truth overlap	9	3
Contribution to TPs [%]	11	37
Contribution to FPs [%]	14	33

Table 4.4: The total number of duplicate detections, along with the fractions that do and do not have significant overlap with a ground truth object mask (*IoU* > 0.5). Also, the contribution of these duplicates to the total amount of *True Positives* and *False Positives* is expressed in a percentage.

Hypothesis 2

The lower performance on real images compared to synthetic images is partly due to the more complex scenarios found in real images, i.e. objects that are located against shelves, located close to another or stacked on top of each other.

Experiment

To evaluate the ability to reject or accept this hypothesis, the real image validation dataset was modified. Complex scenarios in which objects that were stacked on top of each other, located close to another or located close to surrounding shelves were removed from the dataset. A comparison in performance between the original real image validation set and the adjusted non-complex version of this validation set will yield a rejection of acceptance of this hypothesis. If the performance ameliorates, then this proves that complex scenarios are partly causing lower performance on the real image validation set.

Results

Below, the results of the comparison between the original real image validation and the adjusted non-complex validation set are listed. In total, 44 out of 301 objects were removed from the validation set by deleting the annotation or the image as a whole. Logically, the total amount of ground truth detections (*True Positives* + *False Negatives*) slightly decreased due to the removed scenarios. A notable difference can be observed in the amount of missed detections (*False Negatives*) that almost decrease by half, especially considering the push class, that dropped with a relatively large amount. The drop in *False Negatives* is higher relative to the drop in *True Positive* detections, which is observed by the 6.5% increase in *Recall*.

Confidence level > 0.5	**Real**			**Real non-complex**		
	push	grasp	**total**	push	grasp	**total**
TP	100	138	**238**	87	133	**220**
FP	147	145	**292**	141	127	**268**
FN	32	31	**63**	13	24	**37**
Precision [%]	40.5	48.8	**44.9**	38.2	51.2	**45.1**
Recall [%]	75.8	81.7	**79.1**	87.0	84.7	**85.6**

Table 4.5: An overview of the amount of *True Positives, False Positives* and *False Negatives* per affordance class, along with the resulting *Precision* and *Recall*. The comparison is made between the original real image validation set and the same validation set without complex scenarios.

To get a better understanding of the overall performance between these two validation sets, the *Precision* vs. *Recall* curves are plotted in Figure 4.4 and the *AP* is compared in Table 4.6. This metric improved for both the push and grasp classes when removing complex scenarios. A comparison of the average of the two classes shows an increase of 4.2% in *mAP*, which results in an overall improvement of performance.

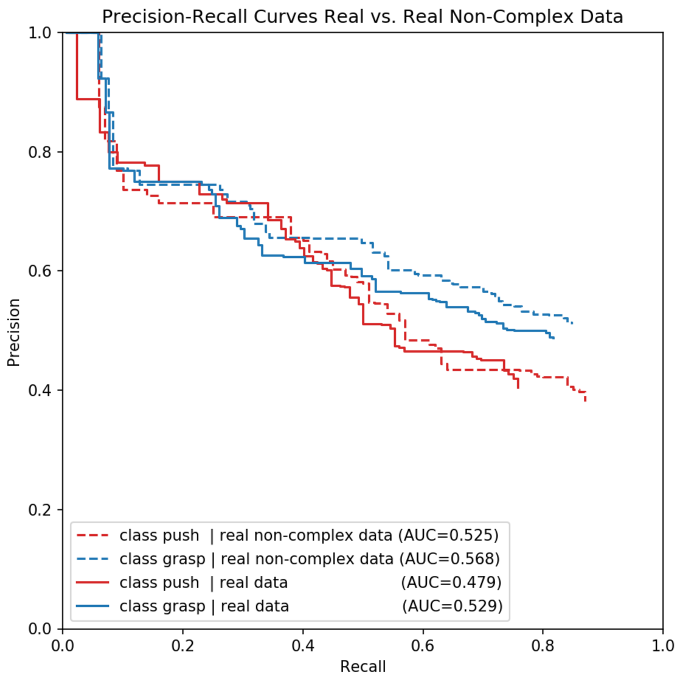


Figure 4.4: A comparison of the *Precision-Recall* curves of the real validation set when removing non-complex scenarios. Each curve is accompanied by its corresponding *Area Under Curve*.

	Real			**Real non-complex**		
	push	grasp	**mean**	push	grasp	**mean**
$AP_{0.5}$ *[%]*	47.9	52.9	**50.4**	52.5	56.8	**54.6**

Table 4.6: A comparison of the model's performance for the original real image validation set and the non-complex validation set in terms of *Average Precision*.

5 Discussion

5.1. Localization & Affordance Classification

As explained in the section covering the architecture of Mask R-CNN (3.1.2), the network consists of two stages. The first stage localizes the object in the image. The results in Table 4.1 show that for this setting, the network can localize most objects from both the synthetic and real validation sets with a *Recall* of up to 100% and 92.1%, respectively. Thus, in terms of object localization, the network has learned well which objects are of interest from the training images. The subsequent stage of Mask R-CNN classifies these objects into their affordance classes. Classification performance can be observed by comparing the localization evaluation with the overall performance results (Table 4.1 & 4.2). Logically, both the *Precision* and *Recall* values drop when including classification into the evaluation. However, this drop is significantly higher for real image data with 23.1 % and 13.0% compared to the synthetic data that drops with 7.9% and 5.5% representing the *Precision* and *Recall*, respectively. The synthetic training and validation images have been split up before the data generation process. This means that the background images and 3D models used to generate these synthetic images were fully split between both sets. Therefore, there cannot be a bias concerning the recognition of backgrounds or products in the synthetic validation data. In short, the network that was trained on synthetic images performs strongly in terms of localizing objects in both synthetic and real images. Regarding the classification performance, the network shows some weaknesses in classifying real image data.

5.2. Synthetic Dataset

One of the main contributions of this work is the novel affordance dataset that was synthetically generated. This dataset is aimed to be used for the training stage of supervised learning-based affordance applications, e.g. the detection and classification of object affordances for mobile manipulation. The ability to generate training images and their masks on the fly is a great advantage over manual human data acquisition and annotation, which is a time-consuming process. The dataset consists of 3237 images accompanied with pixel-wise annotated masks for both *graspable* and *pushable* affordance classes. Another substantial benefit of this method is the possibility to easily enlarge the training dataset using this work's automated data generation script. For this approach of using synthetically generated data for training, it is important to minimize the training data's differences compared to the real-world evaluation data. This is to minimize the difficulties for the network to infer the desired detections in real applications. The results show that, in general, the trained model performs better on synthetic images than on real images; this is due to a better representation of data between training and validation. This observation is expressed in the difference in validation *Loss*, visualized in Figure 3.12. This graph shows that the real validation set converges at a higher *Loss* of 0.55 than the synthetic validation set *Loss* of 0.3. Also, the overall performance comparison between the synthetic and real image validation sets results in a significantly higher *mAP* of 69.8% for the synthetic image validation set, compared to 50.4% for the real image validation set. The difference in performance on real and synthetic data is not purely related to the nature of the synthetic images, as proven in hypothesis 2. This hypothesis is accepted as the results show that the presence of complex scenario images, like stacked objects, have an influence on the model performance evaluation. The corresponding experiment shows that the current content of the synthetic training dataset does not provide enough variation to perform accurate detections on more complex scenarios.

In general, real scenarios in which objects are lying on the floor of retail store environments are accurately represented by the synthetically generated dataset. Still, these results show that this representation can be improved in areas that will be proposed in chapter 7.

5.3. Subjectivity of Affordances

An important aspect that goes in hand with object affordance classification is the subjectivity of the affordance labels. Classifying an object towards being *graspable* or only *pushable* is arguably more subjective than classifying whether an object is an apple or a banana. When can one tell that an object is graspable? For this work, this boundary was defined as a human interpretation of an object being able to be grasped with a single human hand. If an object does not satisfy this condition, it is pushable. The influence of this subjectivity in the model performance is pointed out by the acceptance of the first hypothesis (4.2.3), regarding the performance analysis of both the synthetic and real validation sets. For this hypothesis, the subjectivity of affordances is expressed in duplicate detections. The results on the real image validation set show that up to 97% of the duplicates have overlap with ground truth object masks. In these cases, two detections of the same object that have similar *Level of Confidences* and are classified as different classes, one of which is correct. This double classification is possible because the network allows masks with different classes to overlap. These duplicates show that the network recognizes visual features that relate to both classes, which affects the results of the model performance evaluation in terms of *True* and *False Positives*. For the real image validation set, the 92 duplicate detections account for 37% of the total amount of *True Positives* and 33% of the *False Positives*. For the synthetic image validation set, the 53 duplicates account for 11% and 14%, respectively. These numbers show the impact of duplicates on the evaluation of the model performance, for the real images in particular. The fluctuations that are found in the evaluation *Losses* plotted in the bottom graph in Figure 4.1 confirm this subjectivity. The network encounters difficulties in classifying certain objects into one particular affordance class. A consequence would be that the network outputs incorrect classes, which in turn results in high errors with respect to the ground truth, i.e. a high *Loss* peak.

These numbers show that duplicates generally have a negative impact on the model performance metrics. Affordances are functional categories, and in this work objects are defined to be classified into a single affordance class. This is due to the assumption that there is a clear visual distinction between the *graspable* and *pushable* classes. However, the duplicate detections show that there are objects that contain more than one functional category, which was not taken into consideration in the evaluation process. This is an interesting finding, which generally is not common in conventional object type classification. Yet, in affordance classification this information may add great value to the decision making process of a mobile manipulator for choosing a *grasping* or *pushing* action.

6 Conclusion

In this chapter, the research question of this master's thesis will be answered by means of the sub-research questions along with the discussed results.

How can the concept of affordances be adopted to classify objects that are commonly found in retail environments into their ability to be interacted with, given the use of a mobile manipulator?

6.1. Intermediate Conclusions

To obtain a concise answer to the main research question, the sub-questions proposed in the introduction will be tackled first.

1. How can a conventional object detection model for 2D images be adapted to classify into affordance labels?

   This work has successfully adopted a conventional object detection model to localize the object of interest in the image and subsequently classify it according to a set of affordance classes. Mask R-CNN was trained on the synthetically generated affordance data, in which objects are located on supermarket floors and are assigned affordance labels instead of object type labels. Therefore, the network will train on visual features that correspond to the objects' functional categories rather than their type categories. The model evaluation shows that the network can localize objects located on floors quite well and does generally not detect objects on shelves. In terms of the evaluation of affordance classification between *graspable* and *pushable* labels, the network's performance is not yet optimal. The subjectivity of affordance is partly to blame for this, and several suggestions to improve the classification are made in the *Future Work* chapter.

2. Can the use of a synthetically generated retail product image dataset solve the issue of the lack of appropriate affordance training data?

   The lack of a dataset that consists of retail store related objects annotated with affordance labels was the reason for this question to be asked. A solution to this data gap was found by the design of a data generation script, which was used to generate a synthetic dataset consisting of 3237 semi-labelled images. The term semi-labelled refers to the semantic masks automatically returned by the script. A minimal human effort is needed to link affordance labels to these masks. The use of this synthetic data in the training process of Mask R-CNN has yielded results showing that the network hands in 15.2% mAP when performing inference on real image data compared to synthetic image data. Meaning that the network has learned some characteristics specific to synthetic data, which are not appearing in real data. However, the benefit of not having to create such a dataset manually may outweigh the performance deficit that comes along with this approach for some applications. Also, several aspects of improving the synthetic data generation approach with the aim of more resembling real image data are suggested in chapter 7.

3. Is the resulting affordance model able to generalize objects into their level of interactability, e.g. being *graspable* or *pushable*?

   In the introduction of this work, the assumption of the presence of a clear visual distinction between *graspable* and *pushable* objects was stated. This work shows that in terms of object localization within an image, the trained network performs quite well by capturing most objects. The subsequent object classification stage showed a significant drop in performance which was discussed to be accountable to the subjectivity of affordances. The subjectivity lies in objects that, even for a human, are difficult to classify as either *graspable* or *pushable*. In other words, objects that are clearly distinguishable by humans are handled quite well by the network. However, objects that can arguably have both labels are generally inferred by the network as duplicate detections, meaning that for these objects generalizing affordance classes poses some difficulties. A solution is proposed by introducing joint classes, which will be elaborated on in the next chapter.

6.2. Final Conclusion

This work's main contribution lies in the generation of a novel affordance dataset, which was applied to an instance segmentation network. This allowed localization of objects on retail store floors and subsequently classification into *graspable* or *pushable* affordance labels. This novel dataset was obtained by the design of a data generation script that filled the data gap by generating a retail store related object affordance dataset consisting of 3237 pixel-level annotated images.

A concise answer to the main research question can be formulated, given the answers to the sub-questions. In short, an affordance dataset consisting of retail store related object is essential. This may be obtained using a synthetic data generation approach to avoid intensive manual labour. However, it has been shown that this may compromise the network's performance. In this dataset, objects are labelled towards affordance labels that relate to functional capabilities of mobile manipulators, like *graspable* and *pushable*. A state-of-the-art instance segmentation network can be adopted by training the network on the dataset, after which objects of interest can be accurately localized and classified by a general definition of distinguishing these object affordance classes.

6.3. Limitations

Using the concept of affordances in object detection comes along with its limitations. Primarily, in terms of classifying objects, one must bear in mind the subjectivity of affordance classes. A clear definition to distinct objects of different affordance classes is required. Further, the current approach is designed for mobile manipulators with grasping capabilities that are similar to human hands, meaning that refinement may be needed for other manipulator grippers. Finally, the lack of complex scenarios, e.g. stacked objects, limits the detection capabilities of singular objects. These limitations will be further elaborated upon along with several suggestions for future work in the subsequent chapter.

7 Future Work

7.1. Synthetic Dataset Improvements

7.1.1. Variety in Blender Environment

The first area in which the synthetic dataset can be improved focuses on the variety of the content. In the Blender data generation script, many more conditions can be added to create more variety in the dataset. Lighting conditions were set to a specific setup throughout all images in terms of the location of the lights as well as their intensity and colour. Next to this, 3D object models were generally set to a location within a small range of the image centre. This was required due to the nature of the background images, that not always allowed the object to be placed at another part of the image due to obstructions. Also, more variety in background images may contribute to a better representation of real-world scenarios, especially due to the variety in floor pattern that is accompanied by it. The background images used in this work were limited to two different retail stores. This diversity in floor patterns is important, as these patterns generally are the part of the background image that is directly surrounding the object aimed to be detected.

7.1.2. Complex Scenarios

Several complex scenarios in which the network showed some limitations were touched upon in the second hypothesis under the *Results* chapter (4.2.3). These complex scenarios, like stacked products and products that are lying close to the shelves, were not included in the current synthetic training dataset. Therefore, the model has difficulties in detecting the individual objects as desired under these complex scenarios. Extending the dataset by including these complex scenarios in the data generation phase would enable the network to cope with these detections.

7.1.3. Dataset Size

The third possible amelioration in the synthetic dataset is the training dataset size. A general rule of thumb for supervised learning-based approaches is that more training data results in better generalization capabilities between classes and thus better performance. This is especially important for affordance detection, in which generalization is more difficult to obtain compared to conventional object detection. An important difference is that the classification of object types deals with clear visual features. To catch the variety in features for a specific affordance class, a lot of training examples are required to get a good generalization. The current synthetic dataset size is around 3200 images, from which 2800 images are available for training. This is enough to enable the network to learn features needed to generalize between the classes. However, the model performance evaluation shows that this generalization is not perfect. The amount of *False Positive and Negative* detections listed in section 4.2.2, show that this generalization can be improved.

7.2. Joint Classes

For any future work regarding affordance detection, one must bear in mind the subjectivity of these classes. A focus may be set on the creation of a more objective definition to distinguish between affordance classes with the aim of minimizing duplicate detections. However, this work shows that some objects fall into both affordance classes. This contradicts the assumption introduced in the introduction, which states that there is a clear visual distinction between *graspable* and *pushable* classes. Therefore, further research in the use of joint classes, in which an object may have multiple affordance classes, is proposed. Along with this proposal, an extension to this work may be conducted by incorporating more affordance classes in terms of interacting with objects.

7.3. Model Parameter Optimization

In this work, most of the Mask R-CNN model parameters were set to the default values that were found in its repository. These parameters met the requirements for comparison in performance between different datasets. However, extensive optimization of the model parameters may be conducted as a continuation of this work to improve performance.

7.4. Robot Specific Learning

The implementation of the affordance model on a mobile manipulator was out of reach for this work. Due to the general distinction that was made between the affordance classes, the model will perform differently across various manipulator grippers. The closer the gripper is to the capabilities of a single human hand, to better the model will perform. As a possible continuation, it is believed that a great leap in performance may be achieved by implementing a reinforcement learning approach, on top of this work's general affordance classification. This action- and reward-based approach was discussed in section 2.4. This work's affordance model can be used to cover all initial learning steps, after which the reinforcement learning approach aims to learn robot specific capabilities and thus specifies affordance detection for a particular robot.

7.5. Summary

In general, the difference in data between the synthetic and real image dataset is suggested to be improved, especially in terms of improving variety in background images, object locations and lighting conditions. Also, the inclusion of complex scenarios to the synthetic training dataset is believed to be important for retail store applications, and a focus on increasing the synthetic dataset used for training is suggested as future work. Next to these dataset improvements, a proposal was made to investigate the use of joint classes, in which multiple affordance categories are allowed for an object. In terms of increasing the model performance, extensive parameters optimization may be conducted. Finally, the implementation of a reinforcement learning approach on top of this work's affordance detection model is suggested to learn robot specific affordances.

8 Acknowledgement

Throughout this master's thesis I have been supported and assisted by a great number of people to get where I am now. Primarily, I am grateful to Ir. Max Spahn who has guided me through this master's thesis as my daily supervisor. During this period, Max has always been available for a good discussion and provided me with strong feedback whenever needed. Further, I would like to thank my supervisor Dr. Javier Alonso-Mora for all the intermediate feedback on my work. Finally, I want to show my appreciation to all people from AIRLab who helped me out in many aspects related to my thesis, but also in terms of social activities.

It was an interesting experience to conduct this thesis from home during the COVID-19 pandemic. It did not facilitate the experience, however I am thankful to have had many people around me including my friends, family and girlfriend, who kept me motivated and offered me some distraction during this final phase of my master's degree.

Bibliography

[1] Free 3D Models for Download | TurboSquid. URL https://www.turbosquid.com/Search/3D-Models/free.

[2] Intersection over Union (IoU) for object detection - PyImageSearch. URL https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[3] mAP (mean Average Precision) for Object Detection | by Jonathan Hui | Medium. URL https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

[4] matterport/Mask_RCNN: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. URL https://github.com/matterport/Mask_RCNN.

[5] Overfitting and Underfitting | Kaggle. URL https://www.kaggle.com/ryanholbrook/overfitting-and-underfitting.

[6] Sketchfab. URL https://sketchfab.com/search?q=groceries&sort_by=-relevance&type=models.

[7] The overall network architecture of Mask R-CNN | Download Scientific Diagram. URL https://www.researchgate.net/figure/The-overall-network-architecture-of-Mask-R-CNN_fig1_336615317.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. pages 248–255. Institute of Electrical and Electronics Engineers (IEEE), 3 2010. doi: 10.1109/cvpr.2009.5206848.

[9] Thanh Toan Do, Anh Nguyen, and Ian Reid. AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5882–5889. Institute of Electrical and Electronics Engineers Inc., 9 2018. ISBN 9781538630815. doi: 10.1109/ICRA.2018.8460902.

[10] Abhishek Dutta and Andrew Zisserman. The VIA annotation software for images, audio and video. In *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, pages 2276–2279, New York, NY, USA, 10 2019. Association for Computing Machinery, Inc. ISBN 9781450368896. doi: 10.1145/3343031.3350535. URL https://dl.acm.org/doi/10.1145/3343031.3350535.

[11] James Jerome Gibson. The Theory of Affordance. In *Perceiving Acting and Knowing*. 1977. ISBN 0898599598.

[12] Arren J. Glover and Gordon F. Wyeth. Toward Lifelong Affordance Learning Using a Distributed Markov Model. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):44–55, 3 2018. ISSN 23798939. doi: 10.1109/TCDS.2016.2612721.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. Technical report, 2016. URL http://image-net.org/challenges/LSVRC/2015/.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2 2020. ISSN 19393539. doi: 10.1109/TPAMI.2018.2844175. URL https://github.com/.

[15] Thomas E. Horton, Arpan Chakraborty, and Robert St. Amant. Affordances for robots: A brief survey. *Avant*, 3(2):70–84, 2012. ISSN 20826710. URL www.avant.edu.pl/en.

[16] Peter Kaiser, Markus Grotz, Eren E. Aksoy, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. Validation of whole-body loco-manipulation affordances for pushability and liftability. In *IEEE-RAS International Conference on Humanoid Robots*, volume 2015-Decem, pages 920–927. IEEE Computer Society, 12 2015. ISBN 9781479968855. doi: 10.1109/HUMANOIDS.2015.7363471.

[17] Mia Kokic, Johannes A. Stork, Joshua A. Haustein, and Danica Kragic. Affordance detection for task-specific grasping using deep learning. In *IEEE-RAS International Conference on Humanoid Robots*, pages 91–98. IEEE Computer Society, 12 2017. ISBN 9781538646786. doi: 10.1109/HUMANOIDS.2017.8239542.

[18] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8693 LNCS, pages 740–755. Springer Verlag, 2014. doi: 10.1007/978-3-319-10602-1{_}48. URL `https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48`.

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. Technical report, 2017.

[20] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Multi-model approach based on 3D functional features for tool affordance learning in robotics. In *IEEE-RAS International Conference on Humanoid Robots*, volume 2015-December, pages 482–489. IEEE Computer Society, 12 2015. ISBN 9781479968855. doi: 10.1109/HUMANOIDS.2015.7363593.

[21] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2015-June, pages 3200–3206. Institute of Electrical and Electronics Engineers Inc., 6 2015. doi: 10.1109/ICRA.2015.7139640.

[22] Huaqing Min, Chang'an Yi, Ronghua Luo, Jinhui Zhu, and Sheng Bi. Affordance Research in Developmental Robotics: A Survey. *IEEE Transactions on Cognitive and Developmental Systems*, 8(4):237–255, 12 2016. ISSN 23798939. doi: 10.1109/TCDS.2016.2614992.

[23] Austin Myers, Ching L. Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance detection of tool parts from geometric features. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2015-June, pages 1374–1381. Institute of Electrical and Electronics Engineers Inc., 6 2015. doi: 10.1109/ICRA.2015.7139369.

[24] Anh Nguyen, Dimitrios Kanoulas, Darwin G. Caldwell, and Nikos G. Tsagarakis. Detecting object affordances with convolutional neural networks. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2016-Novem, pages 2765–2770. Institute of Electrical and Electronics Engineers Inc., 11 2016. ISBN 9781509037629. doi: 10.1109/IROS.2016.7759429.

[25] Anh Nguyen, Dimitrios Kanoulas, Darwin G. Caldwell, and Nikos G. Tsagarakis. Object-based affordances detection with Convolutional Neural Networks and dense Conditional Random Fields. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2017-Septe, pages 5908–5915. Institute of Electrical and Electronics Engineers Inc., 12 2017. ISBN 9781538626825. doi: 10.1109/IROS.2017.8206484.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39 (6):1137–1149, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031. URL `https://github.com/`.

[27] Barry Ridge, Aleš Leonardis, Aleš Ude, Miha Deniša, and Danijel Skočaj. Self-supervised online learning of basic object push affordances. *International Journal of Advanced Robotic Systems*, 12(3):24, 3 2015. ISSN 17298814. doi: 10.5772/59654. URL `http://journals.sagepub.com/doi/10.5772/59654`.

[28] Anirban Roy and Sinisa Todorovic. A multi-scale CNN for affordance segmentation in RGB images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9908 LNCS, pages 186–201. Springer Verlag, 2016. ISBN 9783319464923. doi: 10.1007/978-3-319-46493-0{_}12. URL `https://link.springer.com/chapter/10.1007/978-3-319-46493-0_12`.

[29] Sebastian Ruder. An overview of gradient descent optimization algorithms. 9 2016. URL `http://arxiv.org/abs/1609.04747`.

[30] Shiliang Sun. A survey of multi-view machine learning, 12 2013. ISSN 09410643. URL `https://link.springer.com/article/10.1007/s00521-013-1362-6`.

[31] Andreas ten Pas and Robert Platt. Handle detector ROS package. URL `http://wiki.ros.org/handle_detector`.

[32] Andreas ten Pas and Robert Platt. Localizing handle-like grasp affordances in 3D point clouds. In *Springer Tracts in Advanced Robotics*, volume 109, pages 623–638. Springer Verlag, 2016. doi: 10.1007/978-3-319-23778-7{_}41. URL `https://link.springer.com/chapter/10.1007/978-3-319-23778-7_41`.

[33] Emre Ugur and Justus Piater. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2015-June, pages 2627–2633. Institute of Electrical and Electronics Engineers Inc., 6 2015. doi: 10.1109/ICRA.2015.7139553.

[34] Emre Ugur and Justus Piater. Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection. *IEEE Transactions on Cognitive and Developmental Systems*, 9(4):328–340, 12 2017. ISSN 23798939. doi: 10.1109/TCDS.2016.2581307.

[35] Emre Ugur, Yukie Nagai, Erol Sahin, and Erhan Oztop. Staged development of robot skills: Behavior formation, affordance learning and imitation with motionese. *IEEE Transactions on Autonomous Mental Development*, 7(2):119–139, 6 2015. ISSN 19430604. doi: 10.1109/TAMD.2015.2426192.

[36] Natsuki Yamanobe, Weiwei Wan, Ixchel G. Ramirez-Alpizar, Damien Petit, Tokuo Tsuji, Shuichi Akizuki, Manabu Hashimoto, Kazuyuki Nagata, and Kensuke Harada. A brief review of affordance in robotic manipulation research. *Advanced Robotics*, 31(19-20):1086–1101, 10 2017. ISSN 15685535. doi: 10.1080/01691864.2017.1394912. URL `https://www.tandfonline.com/doi/full/10.1080/01691864.2017.1394912`.

[37] Lap Fai Yu, Noah Duncan, and Sai Kit Yeung. Fill and transfer: A simple physics-based approach for containability reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 711–719, 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.88. URL `http://structure.io`.

[38] Philipp Zech, Simon Haller, Safoura Rezapour Lakani, Barry Ridge, Emre Ugur, and Justus Piater. Computational models of affordance in robotics: a taxonomy and systematic classification, 10 2017. ISSN 17412633. URL `http://journals.sagepub.com/doi/10.1177/1059712317726357`.

[39] Andy Zeng, Shuran Song, Kuan Ting Yu, Elliott Donlon, Francois R. Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, Nima Fazeli, Ferran Alet, Nikhil Chavan Dafle, Rachel Holladay, Isabella Morena, Prem Qu Nair, Druck Green, Ian Taylor, Weber Liu, Thomas Funkhouser, and Alberto Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3750–3757. Institute of Electrical and Electronics Engineers Inc., 9 2018. ISBN 9781538630815. doi: 10.1109/ICRA.2018.8461044.