POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering



Politecnico di Torino

Master's Degree Thesis

Segmentation-based approach for a heuristic grasping procedure in multi-object scenes

Supervisor

Prof. MARINA INDRI

Supervisors at Comau

Ing. SIMONE PANICUCCI

Ing. ENRICO CIVITELLI, PhD

Ing. LUCA DI RUSCIO

Candidates

DAVIDE CESCHINI

RICCARDO DE CESARE

October 2023

Abstract

The latest progresses in AI and robotics allow to automate many repetitive and tiring tasks. As a result, the focus of many activities can be transferred to more proactive and stimulating aims.
The work is focused on automating piece picking, that is one of the most common tasks in the majority of industrial/logistic environments.
As a matter of fact, depending on the available robot gripper (vacuum), an algorithm should be able to correctly identify suitable contact points from an RGB+D image. AI applied to image segmentation is a convenient way to achieve such a goal.
At first, a solid model in charge of highlighting each class of objects in a cluttered scene is adopted, using a predefined dataset suiting this specific task. However, this kind of approach, namely multi-class semantic segmentation, is strictly related to the class labels of the chosen dataset.
To increase the flexibility of this method, a change of perspective leads to a model able to segment unclassified objects, which is supposed to be more interesting from a practical point of view. This approach would not depend on the current dataset objects classes; hence it can be applied to detect the two generic classes, object and background, neglecting the semantic contents of each object in the scene.
The first part of the new pipeline consists in adopting object detection models for bounding boxes prediction. The obtained detections are then fed to the state-of-the-art promptable segmentation model, named SAM, which gives as final output the desired segmentation masks. The results are then compared with the ones of Mask R-CNN, a popular end-to-end learning instance segmentation model. Starting from the masks, proper grasping points are estimated by computing their corresponding centroid-like points.
Such approach results to work well in most of the tested scenarios. KNN and PCA techniques are exploited to complete the objective pose of the suction end-effector. Both segmentation performance and final practical tests are reported to show the effectiveness of the entire work.

Acknowledgements

We would like to express our gratitude to Professor Marina Indri for her unwavering support and availability throughout the entirety of this project. We would also like to extend our appreciation to our mentors, Enrico Civitelli, Luca Di Ruscio, and Simone Panicucci, for fostering a stimulating working environment, offering continuous feedback on our work and providing valuable resources during our thesis at Comau S.p.A.

Table of Contents

List of Tables v

List of Figures vi

Acronyms x

1 Introduction and background 1
- 1.1 Grasping robots . . . 1
- 1.2 Computer vision basement with DL . . . 2
  - 1.2.1 Key features of ML . . . 2
  - 1.2.2 From ML to DL . . . 3
  - 1.2.3 Supervised vision tasks . . . 6
  - 1.2.4 Binary vs multi-class approach . . . 10
  - 1.2.5 ML framework . . . 11
- 1.3 Problem statement and thesis structure . . . 12
  - 1.3.1 Thesis outline . . . 12

2 Benchmarks and state-of-the-art 13
- 2.1 Semantic segmentation networks . . . 13
  - 2.1.1 U-Net model . . . 13
  - 2.1.2 PSPNet model . . . 16
- 2.2 Object detection networks . . . 17
  - 2.2.1 SSD model . . . 17
  - 2.2.2 Faster R-CNN model . . . 18
- 2.3 Instance segmentation networks . . . 20
  - 2.3.1 Mask R-CNN model . . . 20
  - 2.3.2 SF-Mask R-CNN . . . 21
  - 2.3.3 SAM . . . 23

3 Datasets 25
- 3.1 Dataset role and its general structure . . . 25

3.2 GraspNet Dataset . . . 26
3.3 ARMBench Dataset . . . 28
3.4 Train, validation and test split . . . 29
3.5 Adopted choice for the problem . . . 30

4 Hardware description 31
4.1 GPU for ML training . . . 31
4.2 RealSense D435i for image acquisition . . . 31
4.3 Racer-3 industrial robotic arm . . . 32

5 Segmentation method 34
5.1 Multiclass semantic segmentation . . . 34
5.1.1 Setting and hyper-parameters configuration . . . 34
5.1.2 Training loss and validation accuracy . . . 35
5.1.3 Accuracy on test dataset . . . 38
5.2 Binary object detection . . . 40
5.2.1 Setting and hyperparameters configuration . . . 40
5.2.2 Training loss and validation accuracy . . . 41
5.3 Binary instance segmentation . . . 45
5.3.1 Training Loss and Validation Accuracy . . . 46
5.3.2 Accuracy on Test Datasets . . . 48
5.3.3 Pro and cons of Mask R-CNN and SAM . . . 49
5.3.4 U-Net vs Mask R-CNN . . . 50
5.3.5 Comparison with ARMBench-trained model . . . 51
5.3.6 Refinement using both datasets . . . 53

6 Grasping method 57
6.1 Robot reference system . . . 57
6.2 From segmentation masks to grasping points . . . 60
6.3 Approaching direction to the grasping point . . . 63
6.4 Total pipeline description . . . 64

7 Experimental tests 67
7.1 Custom-metrics based analysis . . . 67
7.1.1 Segmentation quality criteria . . . 67
7.1.2 Grasping quality criteria . . . 69
7.2 Experiment's outcomes . . . 70
7.2.1 Category 1 : Easy . . . 71
7.2.2 Category 2: Medium . . . 75
7.2.3 Category 3: Hard . . . 79
7.2.4 Category 4: Transparent and Translucent . . . 83
7.2.5 Experimental comparison Mask R-CNN and SAM . . . 85

8 Conclusions and future works 89

Bibliography 91

List of Tables

5.1 Adopted configuration for the training phase of the multi-class semantic segmentation models 35
5.2 Results comparison 39
5.3 Adopted configuration for the training phase of the binary object-detection models 41
5.4 Model weights and inference time on Intel Core i7-1165G7s 44
5.5 Test seen accuracy object detectors 44
5.6 Test similar accuracy object detectors 44
5.7 Test novel accuracy object detectors 45
5.8 Model weights and inference time on Intel Core i7-1165G7s 48
5.9 Test seen accuracy instance segmentation models 48
5.10 Test similar accuracy instance segmentation models 49
5.11 Test novel accuracy instance segmentation models 49
5.12 UNet vs Mask R-CNN accuracy in AP metrics 51
5.13 Comparison between GraspNet and ARMBench models 53
5.14 Accuracy on ARMBench Test 55
5.15 Accuracy on GraspNet Test Novel 56
5.16 Accuracy on combination of GraspNet Test Novel and Test ARMBench 56

7.1 Quality table for Detection (Easy) task 73
7.2 Quality table for Segmentation (Easy) task 74
7.3 Quality table for Picking (Easy) task 74
7.4 Quality table for Detection (Medium) task 77
7.5 Quality table for Segmentation (Medium) task 78
7.6 Quality table for Picking (Medium) task 78
7.7 Quality table for Detection (Hard) task 81
7.8 Quality table for Segmentation (Hard) task 82
7.9 Quality table for Picking (Hard) task 82
7.10 Segmentation times on Intel Core i7-1165G7 87
7.11 Segmentation times on NVIDIA RTX A5000 88
7.12 Picking criteria computational times 88

List of Figures

1.1 Simple MLP with forward and back propagation [2] 4
1.2 Functioning of convolutional layer [3] 5

2.1 U-Net architecture [5] (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. . . 14
2.2 PSPNet architecture [19] 16
2.3 SSD architecture [21] (example for 300x300 image resolution) . . . 17
2.4 Faster R-CNN architecture [12] 20
2.5 SF-Mask R-CNN architecture [24] 22
2.6 Fundamental components of SAM: a) Kind of prompts to be provided; b) Essential structure of SAM; c) Data loop to generate the overall Dataset SA-1B [20] 23
2.7 SAM overview [20]. From left to right: the image encoder embeds the input image that will be connected to an embedding prompt (only mask prompts are embedded using convolutions); the final mask decoder will then output the valid masks, each with its score . 24

3.1 Test Seen example 27
3.2 Test Similar example 27
3.3 Test Novel example 28
3.4 Object Segmentation Dataset of ARMBench [32] 29

4.1 Realsense camera visualization 32
4.2 Racer-3 anthropomorphic arm [36] 32

5.1 Grid search resuming table 36
5.2 Training losses plot of U-Net model 36
5.3 Validation accuracy plot 37
5.4 Training losses plot of PSPNet model 38

5.5 Validation accuracy plot . . . 38
5.6 Target vs predicted masks with U-Net on a test image . . . 39
5.7 Target vs predicted masks with PSPNet on a test image . . . 40
5.8 Training loss plot of Faster R-CNN model . . . 42
5.9 Validation accuracy plot of Faster R-CNN model . . . 42
5.10 Training loss plot of SSD model . . . 43
5.11 Validation accuracy plot of SSD model . . . 43
5.12 Target (on the left) vs Predicted (on the right) bounding boxes with Faster R-CNN model on test novel dataset . . . 45
5.13 Training loss plot of Mask R-CNN model . . . 46
5.14 Validation accuracy plot of Mask R-CNN model . . . 46
5.15 Training loss plot of SF-Mask R-CNN model . . . 47
5.16 Validation accuracy plot of SF-Mask R-CNN model . . . 47
5.17 Training loss plot of Mask R-CNN model trained on ARMBench dataset . . . 51
5.18 Validation accuracy plot of Mask R-CNN model trained on ARM-Bench dataset . . . 52
5.19 Validation accuracy on GraspNet . . . 54
5.20 Validation accuracy on ArmBench . . . 54
5.21 Validation accuracy on weighted combination of GraspNet and Arm-Bench . . . 55

6.1 Robot main reference systems. At the top the local Shaft frame and at the bottom the fixed Base frame [39] . . . 58
6.2 Hand to eye calibration scheme [40] . . . 59
6.3 Camera radial distortion with calibration chessboard [41] . . . 60
6.4 Example 1 of Skeletonize and grasping point choice . . . 62
6.5 Example 2 of Skeletonize and grasping point choice . . . 62
6.6 Example 3 of Skeletonize and grasping point choice . . . 62
6.7 Point cloud with local frame - banana . . . 63
6.8 Point cloud with local frame - Nescaffe . . . 63

7.1 Example of vision errors . . . 68
7.2 Example of picking errors . . . 69
7.3 Point cloud with local frame - ball . . . 70
7.4 Easy picking - single . . . 71
7.5 Easy picking - cluttered . . . 72
7.6 Medium picking - single . . . 75
7.7 Medium picking - cluttered . . . 76
7.8 Hard picking - single . . . 79
7.9 Hard picking - cluttered . . . 80

7.10 Mask R-CNN - Transparent single - scenario 1 83
7.11 Mask R-CNN - Transparent single - scenario 2 83
7.12 Mask R-CNN - Translucent single - scenario 1 84
7.13 Mask R-CNN - Translucent single - scenario 2 84
7.14 Normal vectors for transparent and translucent objects 84
7.15 Mask R-CNN examples 85
7.16 SAM examples 86
7.17 Segmentation mask - Mask R-CNN (right) vs SAM (left) 87

List of Algorithms

1 Algorithm for grasping pipeline 65

Acronyms

AI

Artificial Intelligence

AP

Average Precision

AUC

Area Under Curve

CME

Confidence Map Estimator

CNN

Convolutional neural network

DCNN

Deep Convolutional Neural Network

DL

Deep Learning

FCN

Fully Convolutional Network

FM

Fusion Module

FN

False Negative

FP

False Positive

FPN

Feature Pyramid Network

GAP

Global Average Pooling

IoU

Intersection Over Union

KNN

K Nearest Neighbours

mAP

Mean Average Precision

mIoU

Mean Intersection Over Union

ML

Machine Learning

MLP

Multi Layer Perceptron

NMS

Non Maximum Suppression

NMT

Neural Machine Translation

NN

Neural network

PCA

Principal component analysis

PPM

Pyramid Pooling Module

PRC

Precision Recall Curve

R-CNN

Region Based CNN

RoI

Region Of interest

RPN

Region Proposal Network

SACE

Self Attention Confidence Estimator

SAM

Segment Anything Model

SF

Synthetic Fusion

SGD

Stochastic Gradient Descent

SPP

Spatial Pyramid Pooling

SSD

Single Shot Detector

TP

True Positive

VGG

Visual Geometry Group

Chapter 1

Introduction and background

1.1 Grasping robots

Robotic grasping is an innovative field of study whose aim is to equip robots, and in general machines, with the ability to interact with the environment, similarly to the dexterity of humans. The ability of robots to easily manipulate objects is extremely important across many domains. This capability allows the machines to work alongside humans in a cooperative way, or it may allow the substitution of workers in repetitive and physically demanding tasks. However, robotic grasping extends the boundaries of the mere convenience, in fact it also guarantees efficiency, consistency and security in many fields. A classic example comes from the industrial automation field, where grasping robots are used in assembling and disassembling production's lines. Furthermore, they are crucial in the logistics and warehousing to efficiently fulfil orders and optimize the inventory. Recent development saw also grasping robots applied in surgical procedures, reducing human errors and providing higher precision. The field of interest of the thesis regards industrial automation, in particular the bin picking task.
The bin picking collaborative machines showed different ways of functioning. Initially, model-based approaches were considered to evaluate just the quality of the pose grasp, as in the general case of analytic grasp synthesis mostly was based on mechanics. So, the strong point is to describe, in some realistic fashion, the contact models that can be encountered. These are often divided in friction-less contact, frictional contact and soft-finger contact as reported in [1]. The strong requirement of having a model that is fully-observable leads to a non-flexible way of generalizing the grasping task, given that changing the overall environment settings will cause the failure of the experiment.

In order to have a more robust system a data-driven approach can be a valid solution for such an issue. The development of Machine Learning (ML) in the field of Artificial Intelligence (AI) has already a wide usage among researchers, with particular interest in Deep Learning (DL) techniques. Training a model based on classical inputs like images can make the robot agnostic with respect to the scene that it is observing. To get this point, a deep research has been investigated in the main DL cases of study concerning the computer vision area. This is because, from an industrial point of view, acquiring data through a camera as a sensor is a powerful and convenient working procedure.
With the following sections the aim is to explain the key aspects of ML, focusing more on the specific tasks used throughout the entire pipeline.

1.2 Computer vision basement with DL

1.2.1 Key features of ML

ML is a sub-field of AI that focuses on developing algorithms and models that enable computers to learn from data, and make predictions or decisions without being explicitly programmed for each task. For a general overview it is possible to distinguish several fundamental concepts characterizing ML, and at the same time DL.

- **Data-driven approach :** Data are crucial for machine learning, and the quality and quantity of data significantly impact the model's performance. Data are typically split into training data (used to train the model), validation data (used to evaluate the model's performance on seen examples, allowing to modify the initial training configuration) and test data (used as final check on unseen examples generally).

- **Algorithm Selection :** ML offers a wide array of algorithms, including decision trees, support vector machines, K-Nearest Neighbours (KNN), Neural Networks (NN) and many more. Choosing the right algorithm depends on factors such as the nature of the data (continuous, categorical, etc.), the task (classification, regression, clustering, etc.), and the available resources (computational power, memory, etc.). Given the complexity of the task and the availability of a large amount of data, NNs were the right choice for facing the problem. However also simple shallow learning techniques like Principal Component Analysis (PCA) were adopted for solving secondary points.

- **Type of Learning:** Computer vision with DL is mainly a supervised task, so called since the model is provided with a labelled dataset, where each data point has input features and corresponding target labels. The model learns

to map inputs to outputs by optimizing a loss function that measures the difference between its predictions and the true labels.
For sake of completeness a brief description of the other two types of learning is provided. Unsupervised learning deals with unlabelled data, and the goal is to find patterns or structures in the data without any specific target variable. Reinforcement learning involves an agent that interacts with an environment where the agent takes actions, receives feedback in the form of rewards or penalties, and learns to maximize cumulative rewards over time.

- **Model Evaluation and Hyperparameters Tuning:** Model evaluation involves assessing the model's performance on validation data, using appropriate metrics describing the reached accuracy. This aspect is more consistent with the DL framework, as we will see next.

1.2.2 From ML to DL

Deep learning approach has been widely investigated in the last decades given its capability to lead to an end-to-end learning where raw input data can be mapped to the desired output, in contrast with the classic machine learning algorithms where models cannot learn automatically complex and abstract feature representations from raw data, needing manual feature engineering. The NN keyword better highlights the difference: starting from the basic concept of perceptron, that it is used for simple classification tasks, the combination of multiple layers of this type is responsible for the core architecture of all the well-known deep learning models.

MLP, LOSS & BACKPROPAGATION

Multi Layer Perceptron (MLP) takes into account the complexity of the non-linear phenomena by concatenating a series of linear and non-linear functions.
At each layer there are some 'neurons' in charge of weighting and biasing the input coming from the previous neuron of the previous layer. Such structure will produce a final output which will not be the desired one at first. The basic idea in supervised learning is to introduce a cost function (generally known as loss): a quantity to be minimized with respect to all the parameters of each neuron.
This particular function is chosen in relation to the specific problem and it tells how much the predicted output is far from the assigned target. Thus, minimizing the loss means making the predictions as close as possible to the targets.
The fundamental approach to update the parameters is the gradient descent algorithm in back-propagation mode. This means that all the functions inside the NN are computed in a forward fashion until the computation of the final loss. Then, starting from the gradients of the loss itself, the back-propagated gradients

of the loss, with respect to the internal parameters, are computed in a reverse mode. From a computationally point of view it is a tremendous advantage given that the gradient does not need to be solved n-times, where n is the number of parameters (millions in general), but once for having just the loss as function to be optimized. A quite explicit description is presented in Fig. 1.1. For the loss/error is used the typical mean squared error between the 'Observations' (targets) ad 'Model' (predictions).

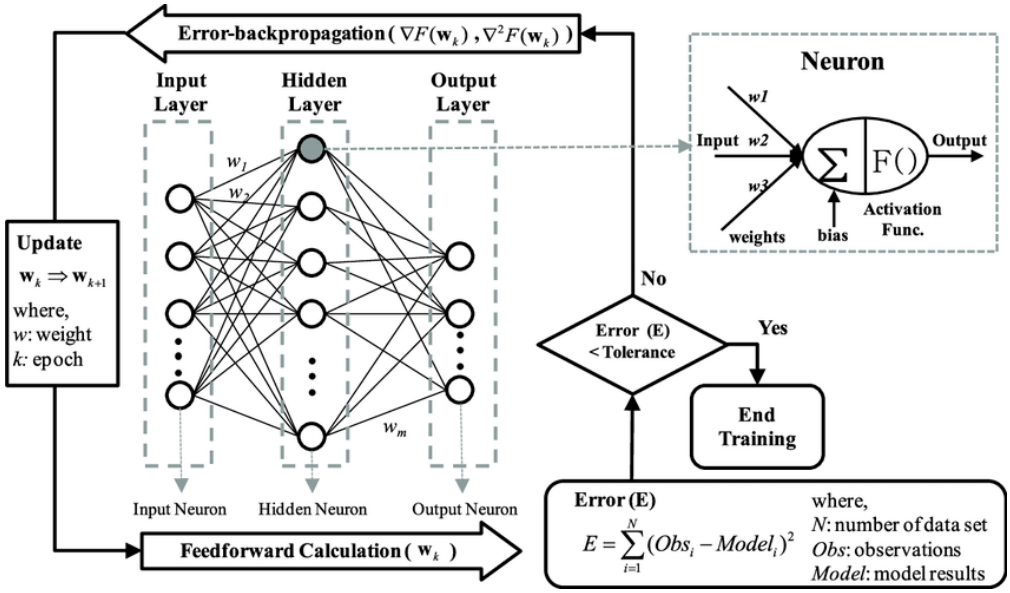


Figure 1.1: Simple MLP with forward and back propagation [2]

LEARNING RATE, SGD & EPOCHS

As reported in Eq. 1.1, where L is the loss function and ω is the generic parameters vector, the gradient descent algorithm tells that for sufficiently small learning steps, given by the term α, the models moves along the steepest descent direction of the considered function, by updating its parameters after each time step t.

$$\omega^{(t+1)} = \omega^{(t)} - \alpha \nabla L_{\omega^{(t)}} \tag{1.1}$$

The coefficient α, which is known as learning rate, plays a key role in the learning phase. Proceeding with the description, it will be shown how the tuning of such a parameter can seriously modify the behaviour of the training. Generally the amount of data is large and computing the gradient with subsequent optimization on the entire dataset would require an enormous computational cost. The idea is thus to perform this step in different batches, each of them containing a portion

of the dataset. Although the final result will not reflect the true one, it can still be considered a good approximation. After a batch iteration the optimization will continue with the next ones until the entire dataset is covered. At the end of this process it is said that one epoch is completed.
This is called Stochastic Gradient Descent (SGD) given the randomic picking of the batches at each epoch.

CNNs

A Convolutional Neural Network (CNN) is a specialized type of NNs that is designed to process and recognize visual data, such as images and videos. The key concept that gives CNNs their name and makes them powerful for image-related tasks is "convolution".
Convolution is a mathematical operation that involves combining two functions to produce a third function. In the context of CNNs, convolution is used to process and analyze images by applying a set of learnable filters, also known as "kernels" or "feature detectors", to the input image. These filters are small matrices, typically 3x3 or 5x5, learned during the training process.
The convolution operation involves sliding the filter over the input image, multiplying element-wise the filter's values with the corresponding input pixel, and then summing up the results to produce a single output. This process is repeated across the entire image, producing a new "feature map" that highlights the presence of certain visual patterns or features in the input image.
This basic idea is reported in Fig. 1.2.

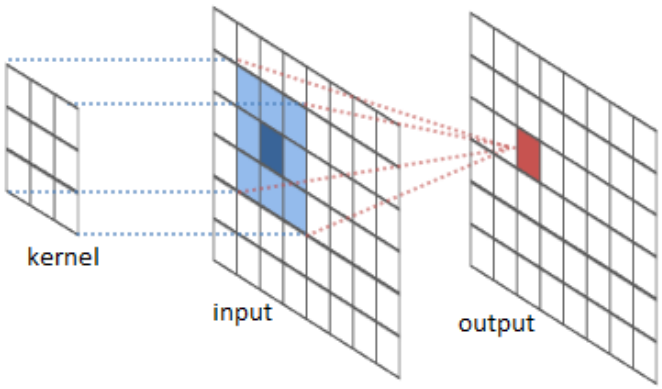


Figure 1.2: Functioning of convolutional layer [3]

Using CNNs on behalf of MLP has several more advantages, some of them listed below:

- **Local connectivity**: Convolutional layers only look at small local regions

of the input image at a time. This allows CNNs to capture local patterns and features without considering the entire image at once, making them more computationally efficient.

- **Parameter sharing**: It reduces the number of learnable parameters as the same set of filters is applied to different regions of the input image. This enables the model to generalize better and work well with various input sizes.
- **Translation invariance**: A particular feature learned in one part of the image can be recognized in a different part of it, making the network robust to changes in position.

1.2.3 Supervised vision tasks

Semantic segmentation

Semantic segmentation is the task of assigning pixel-wise labels to the input images, where each label represents an object category. Such problem has become progressively more important in the fields of robotics, medical surgery, self driving transportation and military applications. The introduction of Deep Convolution Neural Networks (DCNN), has tremendously boosted the task in producing accurate results. The Fully Convolution Network (FCN) [4] was one of the first architectures proposed, which exploited the Visual Geometry Group (VGG) classifier fine-tuned on current task, exploiting transfer learning. The model showed innovative elements, common in many architecture. Due to its nature of fine-grained localization, segmentation task is negatively affected by:

- **Pooling layers**: The hierarchical features created by pooling can loose localization. In addition, depending on the input image size, the output feature will have different dimension.
- **Convolution layers**: The nature of the CNNs themselves does not allow them to capture global context information but only local structures.

FCN was affected by the above problems, but modern approaches deal with these issues.

- **Encoder-Decoder Architecture**: This architecture was first used in computer vision in [5], but it was taken from the context of Neural Machine Translation (NMT) in paper [6]. The decoder consists of a series of convolutions appointed to capture hierarchical features while reducing the image dimensions. The decoder, instead, up-samples the reduced feature map to the original image size.

- **Skip connections**: The idea was first introduced in [7] and they are used to extract features maps from previous layers and fuse them with feature maps from following ones. They are strictly related with the concept of Decoder-Encoder, since skip connection from the former to the latter allows to retain fine grained localization information.
- **Dilated Convolution**: They were first presented in [8], the basic idea is that the convolution filter presents gaps which increase more quickly the receptive field, as the feature map moves deeper in the network, with respect to normal convolution, where the rate of increase is linear.
- **Spatial Pyramid Pooling (SPP)**: It is a technique for efficient training when the training set has images of different resolutions as described in [9]. Different resolution means different features size, hence such pooling avoids cropping all the images. The module performs pooling operations of bins of the features extracted; each bin is a window of size proportional to the feature map size. The operation is performed in a pyramid way: it is done for different number of bins.

The inputs are batched images and the outputs are probability maps with the same resolution of the input image. Each pixel has a depth equal to the number of classes: the location indicates the probability that the pixel belongs to the respective class. The most used loss function is:

- **Cross Entropy Loss**: It is mainly used in classification tasks.

$$L = -\sum_{i=1}^{N} \omega_{y_i} log\left(\frac{exp(x_{i,y_i})}{\sum_{j=1}^{C} exp(x_{i,j})}\right)$$

  The term inside the log is called soft-max function and represents the probability that the pixel belongs to class c. y_i is the class ($[1 : C]$) that the pixel belongs to while x_{i,y_i} is the value of the output i for the class label y_i. ω_{y_i} is a parameter that gives different importance to the various classes. The index i spans the number of pixels and the batch size.

During inference, the predictions are used to state how good the results are with respect to the GT. Several evaluation criteria have been proposed.

- **Intersection over Union (IoU)**: It is the ratio between the area of intersection and the area of union.

$$IoU = \frac{\sum_{c=1}^{N} TP_c}{\sum_{c=1}^{N} FP_c + TP_c + FN_c}$$

  N indicates the number of classes and each term represent the number of pixel.

- **Mean Intersection over Union (mIoU)**: It is an extension of IoU.

$$mIoU = \frac{1}{N} \sum_{c=1}^{N} \frac{TP_c}{FP_c + TP_c + FN_c}$$

  So it is the class averaged IoU.

These two definitions however present some drawbacks. The first one is that they only measure how many pixels are predicted correctly, without considering the accuracy of the boundary. Secondly, they do not consider the difference between False Positive (FP) and False Negative (FN), which is an important parameter in certain applications, such as surgery. Other metrics are:

- **Precision Recall Curve (PRC)**: The curves plot the precision vs recall for a given class.

$$\textbf{Precision} = \frac{TP_c}{TP_c + FP_c} \quad , \quad \textbf{Recall} = \frac{TP_c}{TP_c + FN_c}$$

  The calculation of the coefficients is performed considering all the images in the evaluation dataset. It can be seen how this metrics also accounts for the unbalance between FP and FN indexes. From the precision-recall curve it is possible to compute the mean average precision metric.

- **Mean Average Precision (mAP)**: It is defined as:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

  where AP_i is the average precision for class i and N is the number of classes. The Average Precision (AP) is the Area Under the Curve (AUC) of the precision-recall graph. For segmentation the recall and precision are defined based on the IoU between the predicted mask and the ground truth one, e.g., if two masks have an IoU $>$ t (with t being some threshold) they are considered a match and therefore a true positive. Considering different values of t, all the results for the case of study will be reported in terms of:

  1. **mAP 50**: Here the IoU threshold is 50% as the noun suggests;
  2. **mAP 75**: Similar to mAP 50 but with IoU threshold that is 75%.
  3. **mAP**: This is the global mean average precision corresponding to the mAP for IoU ranging from 0.5 to 0.95 with a step size of 0.05 also referred as 'COCO mAP' from the homonymous dataset [10].

In the results reported in Chapter 5, the 'mean' attribute disappears as only the 'object' class will be present.

Object detection

Object detection is the task of predicting the bounding box of the object, the rectangle which contains the object itself, and provides the right label, the object class, for each identified region. With respect to transfer learning, where the objective is to train on one task and move the problem to a different one, the new task belongs to the multitasking category: multiple objectives must be accomplished in parallel.
Although it may seem a more difficult problem, the main idea is that solving parallel issues may help in the solution of others, reaching higher accuracy with respect to approaching the individual problems. Obviously, the considered tasks must be in some sort connected, for instance they may share the same feature space, as a result they will have an effect on each other.
The general framework for object detection is the following.

- **Backbone**: It is used for feature extractions, most of the times pre-trained backbones are used.

- **Region Proposal Algorithm**: This is an algorithm or another model which predicts the Region of Interests (RoI), which are the regions which may contain an object. Tow popular frameworks are the Selective Search algorithm [11] and the Region Proposal Network (RPN) [12].

- **Feature Extractor**: For each bounding box's prediction the corresponding features are extracted. This will be the input of successive blocks for label prediction or segmentation, for multi stage detectors. In single stage detectors, the label and the boxes are predicted simultaneously.

- **Non Maximum Suppression (NMS)**: This is an algorithm, based on the IoU metric, used to eliminate the additional boxes generated as analyzed in [13]. The algorithm is recursive and at each step the box with the highest score from the list of all current boxes is taken aside, then the IoU between this box and all the other boxes is calculated, the boxes whose IoU is greater than a selected threshold are eliminated from the list; the process is repeated until no boxes are left. In recent years learning based NMS approaches showed good results in improving the accuracy in occluded and dense detection contexts.

During training phase it is common to find the following loss functions.

- **Classification loss**: It is usually a cross entropy loss.

- **Localization loss**: It measures the distance between the real bounding box

and the predicted one, it is usually the smooth L1 loss defined below:

$$\text{SmoothL1Loss}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

During inference the following elements are predicted.

- **Bounding boxes**: The rectangles which determine the object localization
- **Classification labels**: The category in which the objects belong to.
- **Objectness score**: This represents the probability that associated with each bounding box that the area contains any object, independently from the class.

As inference metric it is common to use the AP and mAP. In the calculation of AP and mAP in object detection, the threshold is considered among bounding boxes.

Instance segmentation

The instance segmentation problem consists in finding all the instances of the individual objects in an image. This kind of task involves the object detection part and a segmentation one. Thus similar metrics and loss functions of the previous sections are used.

1.2.4 Binary vs multi-class approach

The semantic segmentation framework is a multi-class approach, thus it requires to know a priori the different classes that are present. It may eventually try to classify the unseen classes by including an unknown class category. This procedure is however complex and requires to model such unknown class. The same semantic segmentation network could also be used as binary predictor, thus focusing on two categories: object and background. However, this is going to create problems when the objects are overlapping each other making them indistinguishable, especially in the bin picking context we are interested in. This is of course a limitation since our objective will be to test networks capabilities to segment also unseen objects. Object detectors, however, are not limited by this, in fact they can be trained to detect only the general object class. The same concept is true for instance segmentation: since it is based on detection the binary classification is suited for this kind of task.

1.2.5 ML framework

To develop the algorithm, the following libraries have been exploited:

- **PyTorch**: It is an open-source machine learning framework which provides flexible solutions to model and train deep neural networks [14]. It has been chosen since the majority of deep learning libraries are based on it and it has GPU support. The framework has also implemented a data-type called tensor, that have GPU-capabilities, and a system called Autograd, an automatic differentiation tool to automatically compute the gradient of tensors with respect to some objective.
- **Torchvision**: It is an important package within the pytorch framework, and it contains the main neural network definitions and loss functions for the three tasks analyzed so far. It is also provided with many visualization tools to easily plot bounding boxes, binary segmentation masks and key-points on existing RGB images.
- **OpenCV**: It is an open source library for computer vision tasks, that was written in C and C++ [15] but provides also interfaces in Python and Java. It allows to read and apply various transformations to images like resizing, filtering and colour space transformations. It is also accompanied by machine learning and deep learning approaches for classification and detection. There are also tools for camera calibration, correcting lens distortion and camera pose estimation algorithm. Despite the many modules, it was primarily used for image acquisition and transformation.
- **Albumentations**: This package provides Python functions to apply augmentations to images as explained in [16]. It is flexible and user friendly since the augmentations are applied considering also the bounding boxes, labels and segmentation's masks. It also has a wide range of augmentation that can be tested.
- **Torchmetrics**: The package contains the main metrics used and it is based on the COCO evaluator. Being user friendly was the reason it has been chosen, especially for the object detection and instance segmentation tasks.
- **Scikit-learn**: It is a Python library [17] that supports many classification, clustering and regression algorithms.

1.3 Problem statement and thesis structure

The goal of the thesis is to develop a grasping pipeline to be applied in the industrial context of bin picking. Such algorithm will be divided into two parts: a segmentation and a grasping algorithm. The segmentation network will be responsible for detecting, as precisely as possible, the shapes of possibly all the visible objects in the scenes. The most important requirement is that it should accurately predict all the completely visible objects. The accuracy of the network will be tested on test's datasets to provide the theoretical results for the choice of the model. In addition practical experiments will be carried out. Such experiments will focus on two scenarios: detection of unseen objects and detection in highly occluded environment with similar objects. A simple grasping policy will also be presented, thus the experiment will be correlated with user-defined metrics on the segmentation and grasping part to determine the weakest point of the algorithm.

1.3.1 Thesis outline

In Chapter 2 the models' architectures are discussed. As starting point an extensive structural and functional analysis on the most relevant NN was carried out, in the context described in Section 1.2.4. Besides the models investigation, the datasets selection was a crucial point, too. Chapter 3 is therefore dedicated to the description of the two adopted ones, highlighting the main reasons of such choices. Then, in Chapter 4 a brief hardware overview is reported. In Chapter 5 the segmentation outcomes are depicted: both the multi-class semantic segmentation and the binary instance segmentation achievements are available in terms of plots and resuming tables, which show the training, validation and test phases of the corresponding models. The segmentation results are followed, in Chapter 6, by the description of the pipeline and its heuristic for the grasping point determination. The first part of the chapter illustrates the procedure to get admissible grasping points, whereas the second one describes the picking pose calculation.
The final practical tests were realized at the Comau's robotic cell and they are reported in Chapter 7. The test objects were clustered in 3 categories, according to their capacity of being picked. For each of them two different scenarios were analyzed, depending on the degree of clutterdness. Chapter 8 concludes the thesis, outlining the results and the conclusions derived by the overall work. Tips and future works are then proposed in order to give a larger view of the problem and possible solutions to better exploit this kind of pipeline.

Chapter 2

Benchmarks and state-of-the-art

2.1 Semantic segmentation networks

Starting with the multi-class semantic segmentation approach, it was investigated the most solid architectures that enable to accomplish such a task. In particular the U-Net and the PSPNet models were studied. Obviously, more complex and sophisticated networks can be found (for instance based on Transformers), however this would result in overspending in terms of computational resources that is out of the scope of this thesis.

2.1.1 U-Net model

U-Net is a popular convolutional neural network architecture widely used in image segmentation tasks, particularly in biomedical image analysis. It was proposed by Olaf Ronneberger et. al. in their 2015 paper [5]. The U-Net's architecture is named after its U-shaped structure, which consists of an encoder and a decoder. The encoder captures the spatial features of the input image and reduces its spatial dimensions through a series of convolutional and pooling layers, while the decoder up-samples the reduced feature maps back to the original input image size. The key innovation of U-Net is the introduction of skip connections that help to propagate high-resolution features from the encoder to the decoder. These skip connections enable precise localization using fine-grained details learned in the encoder part to construct an image in the decoder part, making U-Net particularly suitable for tasks like image segmentation. The overall structure is depicted precisely in Fig. 2.1.

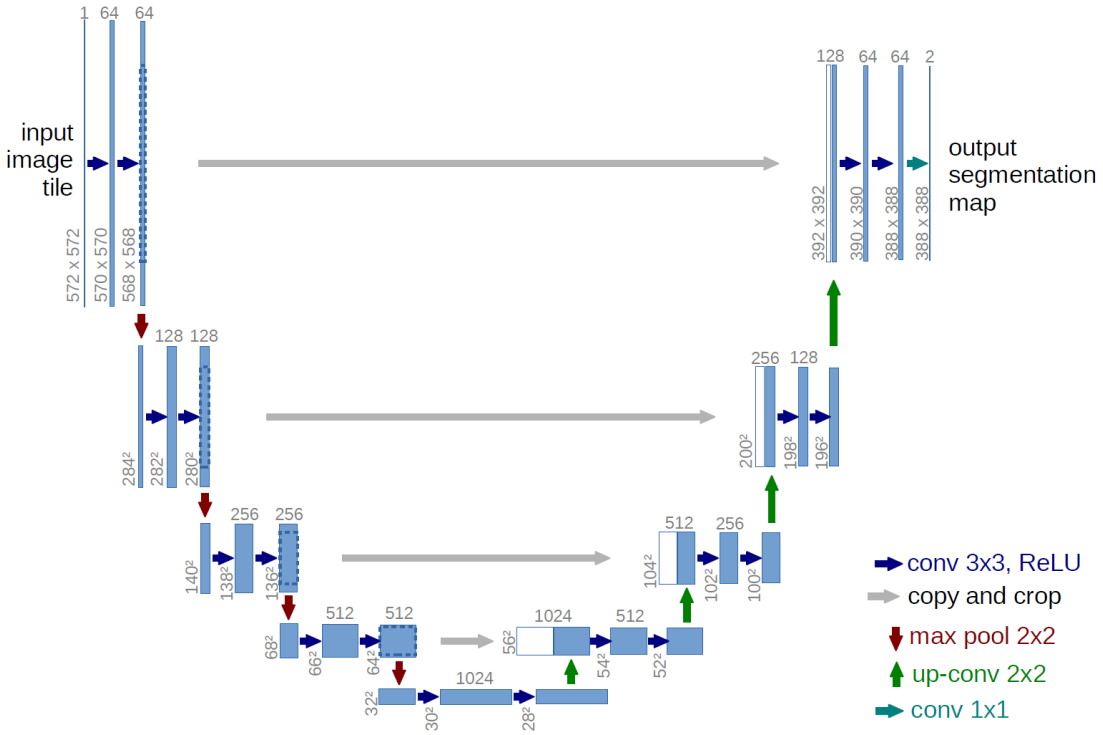


Figure 2.1: U-Net architecture [5] (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Going through the U-Net architecture in detail:

- **Encoder:** The encoder part of U-Net consists of multiple repeated blocks of convolutional layers followed by a max-pooling operation. Each block typically includes two or more convolutional layers, each followed by a batch normalization and a ReLU activation function. These layers are responsible for learning and capturing the hierarchical features of the input image while reducing its spatial dimensions.

- **Decoder:** The decoder part of the U-Net consists of multiple blocks that upsample the feature maps to the original input image size. Each decoder block includes a transposed convolution (also known as deconvolution or upsampling) followed by a concatenation with the corresponding feature map from the encoder. This concatenation forms the skip connection, and it allows the model to retain detailed spatial information from the encoder, which helps

to have an accurate segmentation.

- **Skip Connections:** As previously mentioned, the skip connections are a critical component of the U-Net architecture. They connect the feature maps from the encoder to the corresponding decoder blocks. These connections help the decoder to recover fine-grained details from the early layers of the encoder that might have been lost during the down-sampling process.

- **Output Layer:** At the end of the decoder, a 1x1 convolutional layer with an appropriate activation function (usually sigmoid for binary segmentation or softmax for multi-class segmentation) is used to produce the final segmentation map with the same spatial dimensions as the input image.

- **Loss Function:** The common loss function used with U-Net for image segmentation tasks is the cross-entropy loss. It measures the dissimilarity between the predicted segmentation map and the ground truth mask. The goal during training is to minimize this loss function.

- **Data Augmentation:** To increase the diversity of the training data and improve the model's generalization, data augmentation techniques like rotation, flipping, random cropping and color jitter are often applied.

The U-Net architecture has proven to be effective in various image segmentation tasks, including biomedical image segmentation (e.g., cell nuclei segmentation, tumor detection), road and building segmentation in satellite imagery and more. Its ability to capture detailed spatial information and its efficient use of skip connections make it a popular choice for semantic segmentation problems. For this reason it was thought that also with industrial objects it could have worked effectively.

Backbone structure

Backbone networks are responsible for extracting relevant and informative features from the input data. These features capture various levels of abstraction and patterns present in the data. In image data, for instance, lower layers of a backbone network might detect simple edges and textures, while higher layers might capture more complex shapes and object parts. Moreover pre-trained backbone networks can be used as a starting point for various tasks. By training a backbone network on a large dataset, it learns generic features that can be fine-tuned for specific tasks. The U-Net model was implemented using Segmentation Models library in PyTorch framework. This allowed to use a model, with a standard backbone (ResNet-18 [7]), pre-trained on the large dataset ImageNet [18].

2.1.2 PSPNet model

Pyramid Scene Parsing Network (PSPNet) is a neural network introduced by Hengshuang Zhao et al. [19] in 2017. The network's structure is capable of capturing multi-scale contextual information making it suitable to find structures at different scales, which is essential in computer vision. The network is shown in Fig. 2.2.

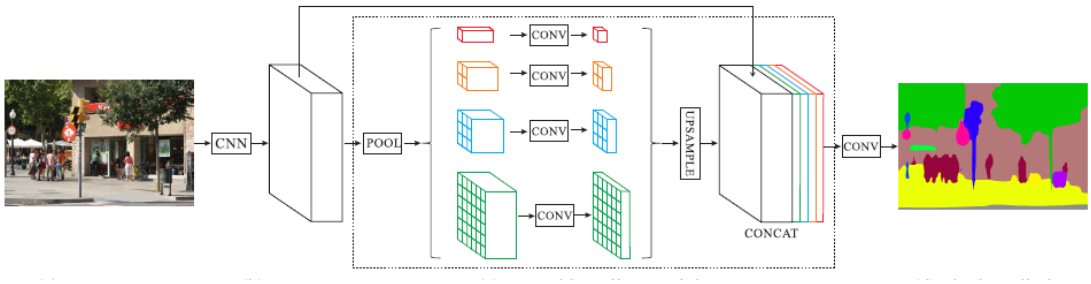


Figure 2.2: PSPNet architecture [19]

It follows the network's architecture:

- **Encoder**: The original paper uses a ResNet-50 as encoder with dilated convolution. Due to the training resources we opted for a ResNet-18 with dilated convolution.
- **Pyramid Pooling Module (PPM)**: Global Average Pooling (GAP) has been long used in image classification due to its ability of capturing global contextual information, and it is also used in segmentation. However, the latter presents highly annotated scenes, hence GAP may loose the spatial relations. The proposed pooling solves the problem by dividing the last feature map in non overlapping bins and performs average pooling on each of them. The features are then reduced in dimension through a 1x1 convolution. The original implementation uses 6 bins of size 1x1,2x2,3x3 and 6x6. The presence of this module can be beneficial in the context of segmentation of cluttered object scenes where capturing contextual relations between the object is of paramount importance.
- **Decoder**: The decoder architecture is a simple up-sampling operation, performed on every feature map of the PPM. These features maps are then concatenated with the last feature map of the backbone. The final convolution generates the prediction map, which is arg-softmaxed to produce the segmentation mask.

The model is based on the same library of U-Net.

2.2 Object detection networks

The introduction of Segment Anything Model (SAM) [20] by meta-AI allowed to focus on object detectors, since the accuracy of the segmentation is now depending on the accuracy of the bounding boxes predictions. Two families of common object detectors have been studied: multistage object detectors and single stage object detectors.

2.2.1 SSD model

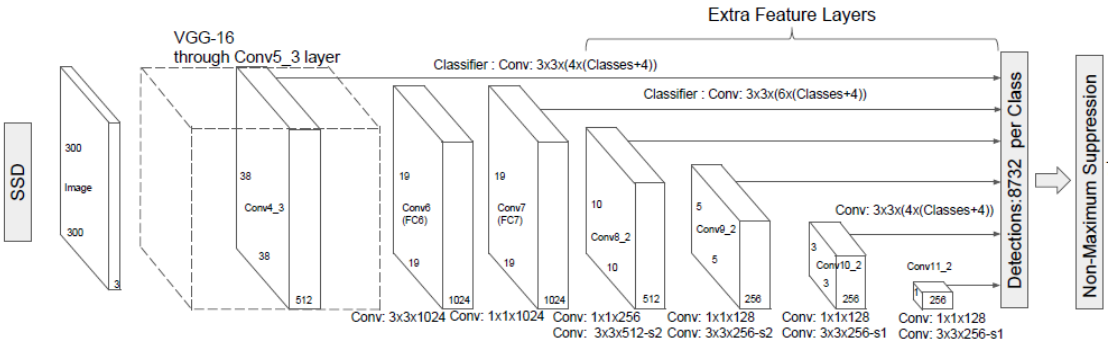


Figure 2.3: SSD architecture [21] (example for 300x300 image resolution)

The SSD network, shown in Fig. 2.3, is composed of five main parts:

- **Feature extractor network**: The original paper [21] uses a VGG16 but this architecture was changed with ResNet-50 [7] defined in torch vision library. The network was truncated at the conv4_3 layer, thus neglecting the conv5 layers and the classification head of the base architecture.
- **Multi-scale features layers**: This block consists of a series of convolutional layers progressively decreasing in size to allow making predictions at multiple scales. From this block 6 feature maps are extracted: the first one comes out the conv4_3 layer and the others come from the convolutional layers of the multiscale feature extractor.
- **Bounding box proposal**: at each feature maps, fixed bounding boxes are proposed: at each pixel of every feature map the shifts with respect to the fixed bounding boxes are predicted. The procedure to choose the fixed bounding boxes is explained in detail in [21].
- **Prediction heads**: There are two heads, a regression head which predicts the bounding boxes shift and a classification head that predicts the class

probability. This is done through convolutional layers, with 3x3 kernel and padding 1 to keep the dimensions, for each feature map extracted before and for each bounding box proposal for the corresponding feature map. Hence a total of (c+4)*k filters are used per feature map, where c is the number of classes (for binary problem 2), 4 is the number to completely describe a bounding box and k is the number of filters. However, only the boxes that have a score above a certain value are kept.

- **NMS**

The network was taken from the Torchvision library and it accepts as input a list of tensors of shape 3xHxW and returns the bounding boxes in pascal-voc format (4 edges (x1,y1,x2,y2)), the objectness score, which is the class specific probability associated to each proposal, and the classification label. In training model also the localization loss (smooth l1 loss) and the classification loss (cross entropy loss) are given.

2.2.2 Faster R-CNN model

Faster R-CNN (Region Convolutional Neural Network) is a foundation object detection architecture that was introduced by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun in their 2015 paper [12]. Faster R-CNN builds upon the R-CNN and Fast R-CNN frameworks and improves both the accuracy and efficiency of object detection.
The Faster R-CNN architecture consists of several key components that work together to achieve accuracy and efficiency in object detection:

- **Backbone Network:** Faster R-CNN starts with a backbone convolutional neural network (CNN) that extracts hierarchical features from the input image. Common choices for the backbone include architectures like VGG16, ResNet, or ResNeXt. The backbone network processes the entire image and generates a feature map. The off-the-shelf architecture provided by PyTorch was used, it relies on ResNet-50 [7] as pre-trained backbone given its great popularity in DL applications. In addition, a Feature Pyramid Network (FPN) [22] is present: it is used as the head of the backbone network because it addresses the challenge of handling objects at different scales, improves object localization, integrates multi-scale and semantic information, enhances performance in object detection.

- **RPN:** The Region Proposal Network (RPN) is a crucial innovation of Faster R-CNN. It is a fully convolutional network that operates on the feature map generated by the backbone. The RPN's primary function is to propose regions of interest that are likely to contain objects. It does this by sliding a small

anchor window of different sizes and aspect ratios across the feature map and predicting whether an object is present inside each anchor window. The RPN predicts objectness scores and adjusts anchor box coordinates to generate region proposals.

- **RoI Pooling (or RoI Align):** The region proposals generated by the RPN are then used to extract fixed-size feature maps from the backbone feature map. This process is known as Region of Interest (RoI) pooling. In Faster R-CNN, RoI pooling converts variable-sized regions into a fixed-size feature representation, which is then fed into subsequent layers.

- **Classifier and Bounding Box Regressor:** The extracted RoI features are used to predict class probabilities and refine the bounding box coordinates of the detected objects. These predictions are performed using fully connected layers. The classifier predicts the class label of the object present in each RoI, while the bounding box regressor refines the coordinates of the object's bounding box.

The Faster R-CNN model works entirely in two stages as in Fig. 2.4.

1. **Backbone and RPN:** The RPN is trained as a binary classification task to determine whether each anchor contains an object or not. It is also trained to adjust the anchor box coordinates to better fit the ground truth object bounding boxes.

2. **Detection network:** After training the RPN, the object detection network concerning the RoI Pooling is trained (the two stages occur sequentially in real-time). This involves training the classifier and bounding box regressor on the extracted RoI features using multi-task loss functions. The loss functions include terms for class prediction and bounding box regression, encouraging accurate localization and classification.

Faster R-CNN shows in this way several key advantages making it one of the most reliable and efficient object detector.
Faster R-CNN enables end-to-end training, which means that the entire model, including the backbone network, the RPN and the object classification/regression components, is trained jointly. This leads to more efficient optimization and improved overall performance.
The use of an end-to-end trainable architecture and the integration of the RPN contribute to improve accuracy in object detection. The model's ability to generate high-quality region proposals and its multi-scale feature fusion lead to more precise object's localization and classification.

Last but not least, Faster R-CNN is relatively straightforward to understand and implement compared to some other complex object detection architectures. This simplicity makes it accessible to a wide range of researchers and practitioners.

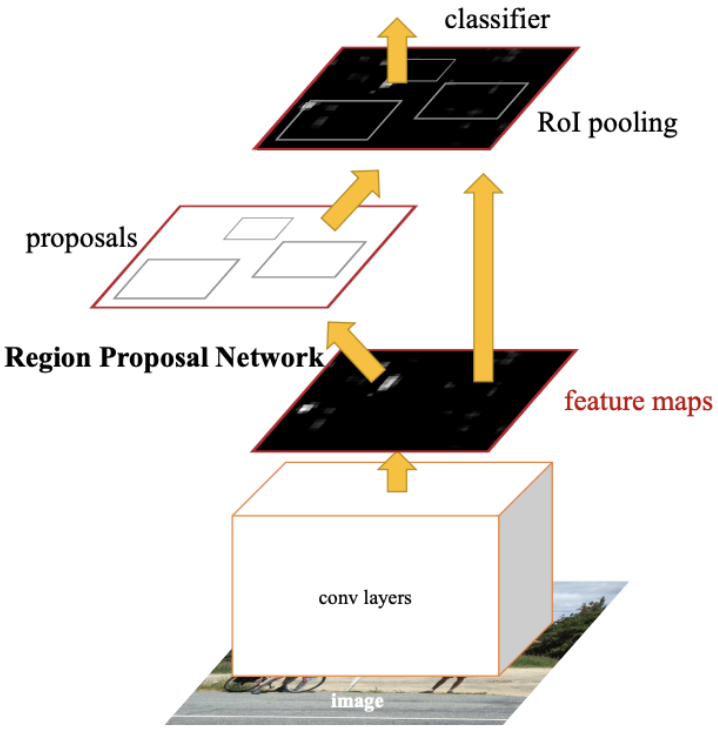


Figure 2.4: Faster R-CNN architecture [12]

The model was taken from the Torchvision library and it employs several loss functions to train the RPN, the classifier, and the regressor. For the RPN, the loss includes terms for objectness classification and bounding box regression. The classifier uses cross-entropy loss for the class predictions, and the regressor uses a smooth L1 loss for the bounding box coordinates.

2.3 Instance segmentation networks

2.3.1 Mask R-CNN model

The instance segmentation task is challenging since it includes two sub-tasks: object detection and segmentation. One of the most common network to tackle the problem is Mask R-CNN developed in 2018 by Kaiming He et al. [23]. The idea is to introduce a mask prediction head, parallel to the ones of Faster R-CNN,

to predict binary segmentation masks for each RoI. The architecture is made of the following blocks:

- **Object detector**: Faster R-CNN
- **Segmentation head**: This is an MLP that predicts for each RoI a mxm segmentation mask (the Torchvision implementation uses the original $m = 14$). The output of the head has dimension Km^2 where K is the number of classes possible, hence it predicts K binary segmentation masks of dimension mxm.

Since it is based on Torchvision implementation of Faster R-CNN it presents the same outputs, with the addition of the binary segmentation masks with the associated label during inference, and the segmentation loss (a binary cross entropy) during training.

2.3.2 SF-Mask R-CNN

Synthetic Fusion Mask R-CNN (SF Mask R-CNN), is an architecture introduced by Seunghyeok Back et al. [24] in 2020, which extends the feature extraction of Mask R-CNN to include the information coming from depth images of the scenes, by introducing a novel fusion module. The fusion techniques are essential when the information of the RGB is not sufficient, especially in the context of industrial bin picking, where high occlusions confuse the boundaries between objects. This concept can actually be extended to any situation where the object boundaries are difficult to distinguish. Thus, the depth information can help segmentation in these contexts; this clearly requires a good quality depth image, which results in a higher cost for depth camera purchase. Two of the most common methods are:

- **Early fusion**: RGB and depth are concatenated and given to the feature extractor.
- **Late fusion**: RGB and depth have their respective feature extractors, these output features are then concatenated or summed (eventually with fpn at multiple scales) and passed through a 1x1 convolution. However there is no Confidence Map Estimator.

The adopted fusion approach, explained in [24] and sketched in Fig. 2.5, proved to be better in terms of AP. The general architecture is based on Mask R-CNN with FPN provided by Torchvision library.

- **Backbone:** Two backbones for RGB and depth feature extraction, the RGB branch uses a ResNet-50 pretrained on ImageNet whilst the depth branch a ResNet-50 with kaiming initialization.

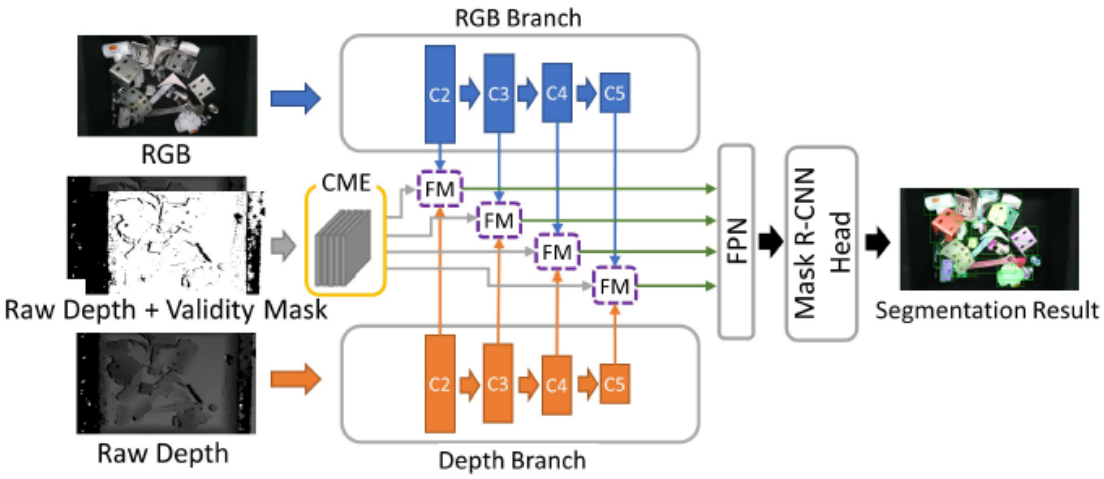


Figure 2.5: SF-Mask R-CNN architecture [24]

- **Confidence Map Estimator (CME):** This is a small convolutional network which receives as input the depth image and a validity mask (a binary mask which is 1 where there is the depth information and 0 elsewhere). Such module is used to assign a pixel-wise reliability of the depth information to avoid errors due to the bad quality of the input depth. A more recent paper [25], of the same authors, proposed a different estimator called SACE (Self Attention Confidence Estimator), which estimates the confidence map from an inpainted depth[1] and RGB images; the CME could only use raw depth. Such confidence map assigns a higher score where the most important features are present. This improved the precision of the segmentation.

- **Fusion Module (FM):** It is the block appointed to the fusion of depth and RGB features. Inside it, the confidence map is resized to the corresponding depth features size and these are multiplied together. The weighted features are concatenated with the RGB ones and passed through a 1x1 convolution.

The network provides the same outputs of MaskRCNN model during both inference and training, since it is based on the Torchvision implementation of MaskRCNN.

[1]depth that underwent inpainting: a process to fill in missing or corrupted parts of an image in a plausible way

2.3.3 SAM

Since instance segmentation, and in general semantic segmentation, are widely explored and studied tasks in computer vision, many improvements have been reached throughout the years. In particular, in March 2023 a new off-the-shelf model for semantic segmentation was developed, named SAM [20].
This is definitely the state-of-the-art for semantic segmentation tasks using DL techniques. Its main objective is to detect the semantic masks of whatever kind of element in a scene (objects, animals, people, ...), making it the first class-agnostic model for semantic segmentation. As reported by the authors, to get such a model they used a data collection loop obtaining the largest segmentation dataset, containing over 1 billion masks and about 11 M licensed images. The entire work was based on 3 main components: the declaration of the desired segmentation via specific prompts, the overall model (SAM) responsible for the data annotation with its final use as promptable segmentation model, and the already mentioned dataset (SA-1B). These three key ingredients are depicted in Fig. 2.6.

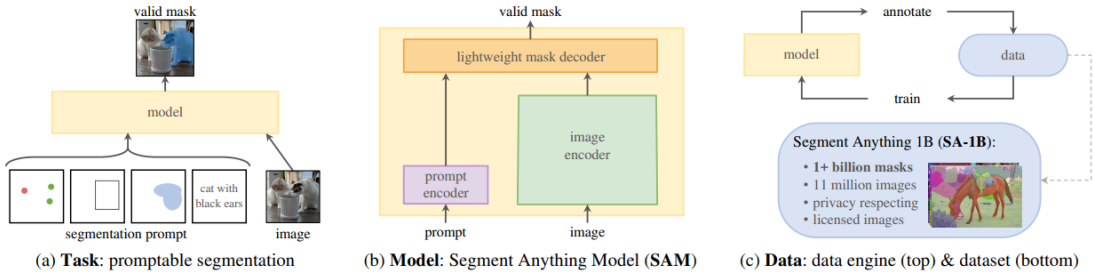


Figure 2.6: Fundamental components of SAM: a) Kind of prompts to be provided; b) Essential structure of SAM; c) Data loop to generate the overall Dataset SA-1B [20]

Segment Anything Task

This model aims to be as generalize as possible in terms of required prompts to be fed in, guaranteeing reliable segmentation outcomes. In particular, four types of prompts are used: points, boxes, raw masks and text information. This fundamental peculiarity enables to shift the segmentation problem to engineering appropriate prompts. This is the key idea that was followed: adopting an object detector to achieve trusted boxes to be fed into SAM for the mask prediction.

Segment Anything Model

The complete model has been realized taking into account the two kinds of input and the final output. Its final structure is composed by: an image encoder, a prompt encoder and a mask decoder. The connected structure is described in Fig. 2.7. For the first module, the authors used a MAE pre-trained Vision Transformer (ViT) [26], to make the model as reliable as possible: this allows to output an image embedding which can be successfully queried by different input prompts to produce segmentation masks. Then a prompt encoder is used to effectively combine prompts and image embedding: for points, boxes and text a sparse prompt is used whereas for masks a dense one. Finally, the ending mask decoder (inspired by Bowen Cheng et al. [27]) is made of a modified Transformer decoder block and a MLP, which maps the output token to a dynamic linear classifier in charge of computing the mask foreground probability at each image location.

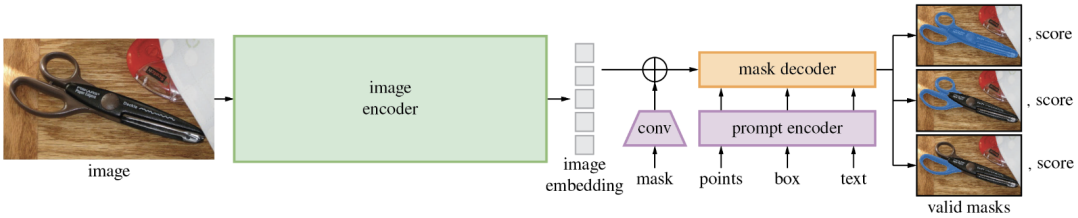


Figure 2.7: SAM overview [20]. From left to right: the image encoder embeds the input image that will be connected to an embedding prompt (only mask prompts are embedded using convolutions); the final mask decoder will then output the valid masks, each with its score

Dataset Creation

The final dataset used to train the overall model was realized in three main steps; the architecture of the model evolved together with the dataset. The first stage consisted in manually labelling the images with a browser-based segmentation tool. The model was initialized on common public datasets, then it was retrained with these new masks. At each step the model was used to help the manual annotators, thus progressively reducing the annotation time. The second phase was a hybrid approach which mixed an automatic generation of the most confident masks with the manual annotation of the remaining ones. At the end, a full automatic process was realized based on the enormous amount of masks previously annotated and on the so called 'ambiguity-aware model', that is the final model that for each object outputs 3 possible masks (from the smallest to the largest).

Chapter 3

Datasets

3.1 Dataset role and its general structure

Facing a supervised learning task like semantic and instance segmentation, the other fundamental element to choose, in addition to the model, is the dataset. This because the model itself tries to learn a proper solution to the desired task relying on information provided by the dataset. It becomes clear how these specific information can affect and address the final solution to a certain output rather than another one.
Considering the assigned task at first the focus was on GraspNet, a quite heterogeneous and reliable dataset that allowed to tackle all the required computations in a proper time. A more recent dataset was then discovered, it is called ARMBench and it mainly deals with generic graspable objects. At the end, to take advantage from their relevant peculiarities, both sets were exploited to get best outcome.
The datasets should cover the two main applications in picking context:

- **Bin picking**: objects within a packaging
- **Plane picking**: objects that lies on a surface

Each scenario should have the following annotations:

- RGB image
- Depth image: this is a nice to have requirements, but it is not necessary for segmenting with many of the models considered.
- Segmentation mask

The bounding boxes are not necessary since they can be derived from the segmentation masks.

3.2 GraspNet Dataset

GraspNet is a large scale dataset built by Hao-Shu Fang et al. in the context of object grasping. The dataset was presented in [28] together with an end-to-end grasp prediction network, working directly on point clouds. The dataset was developed with the intention of enriching the literature of multi-scale-multi-grasp grasping, in fact, up to then the main focus was on isolated objects or single grasp scenarios. This was mainly due to the high effort required to annotate such datasets. GraspNet contains a total of 88 objects: 32 belongs to YCB dataset [29], 13 from DextNet 2.0 [30] and 43 were picked by the authors. The differences in shapes, textures and local geometries makes them suitable for grasping. A total of 190 scenes were collected, each scene was captured under 256 perspectives with 2 cameras: a Realsense 435 and a Kinect 4 Azure. Both the RGB and Depth images were saved leading to a total of 512 images per scene, each of them contains around 10 objects. The picking procedure was automated with a robot arm, to which the cameras were attached, that slid towards the various points of view. Each scene is correlated with several annotations:

- Camera poses with respect to first frame.
- Camera pose of the first frame with respect to world frame.
- Camera intrinsic.
- Segmentation mask: the mask contains the numbers from 0-87 for the 88 objects present;0 refers to background.
- 6D pose annotations of objects in the scene
- Grasping pose annotations. The grasping points have been generated according to a two step procedure. Accurate 3D meshes of the objects were down sampled obtaining a few grasping points, for each of them V views were sampled uniformly on a sphere. The grasp candidates were found on a grid of dimension DxA where D is the number of gripper depth and A the set of inplanes rotations. Each grasp candidate was assigned a confidence score according to a force-closure metric. Using the 6D pose annotations of the object in the scene, this grasps are projected onto the scene itself.

The dataset has been divided as follows, where test cases are reported in Figs. 3.1, 3.2 and 3.3:

- **Train**: scenes 0-99.

- **Test seen**: scenes 100-129. It contains objects that have already been seen during training (40 out of 88 classes), but with different positions and perspectives.
- **Test similar**: scenes 130-159. It contains objects of similar shapes and semantic information with respect to the training set (23 out of 88 classes, different from test seen ones). For example a real object as a drill in train set is represented as a toy-drill in similar test set.
- **Test novel**: scenes 160-189. It contains object of unseen shape.



Figure 3.1: Test Seen example



Figure 3.2: Test Similar example



Figure 3.3: Test Novel example

3.3 ARMBench Dataset

As the main aim of this project is to make the final DL model able to segment the maximum quantity of unseen object becoming a 'class-agnostic' binary segmentation model, new datasets with high object heterogeneity have been explored. Concerning the industrial picking a quite recent dataset was found to be the most appropriate one: the ARMBench dataset by Amazon [31].
The authors wanted to provide to researchers a large-scale dataset of scattered objects within a tote box, highlighting the variety of such objects and the degree of clutter. This is because automation processes in a modern warehouse always require a valid robotic arm able to face an unstructured storage and an ever-changing inventory, like the case of an Amazon stock. To accomplish the more general operations in an automated picking process, they developed three macro-sub-datasets, each of them concerning a specific task:

- **Object Segmentation:** It consists of over 50k rgb images with more than 450k high-quality annotations. It is characterized by a huge variation in the objects and the degree of clutter, as previously mentioned.

- **Object Identification:** It presents over 200k unique items in varying settings, it is used for classification with uncertainty estimation.

- **Defect Detection:** Over 19k images with more than 4k videos of activities with defects and 100k without defects.

This overall view allows the reader to be aware of what kind of dataset the chosen task is based on, related to their own needs. The first dataset was the required one for training a segmentation model, it presents many different objects with respect to the GraspNet dataset and more occluded scenes.

Object Segmentation

Object Segmentation dataset of ARMBench is subdivided into three other categories: Mixed-Object-Tote, Zoomed-Out-Tote-Transfer-Set and Same-Object-Transfer-Set as shown in Fig. 3.4. As reported in the figure, the main part of such a dataset is the Mixed-Object-Tote, where the generic tote box is filled with a variable number of objects of different shape and consistency. The remaining two are showed for the sake of completeness. Due to annoyances and needs like the dimension of the images (which is not always the same) and the number of objects (that can exceeds the 25-30 annotated masks per image), in the final implementation practical tricks have been adopted, enabling a proper training of the model using the ARMBench Dataset.

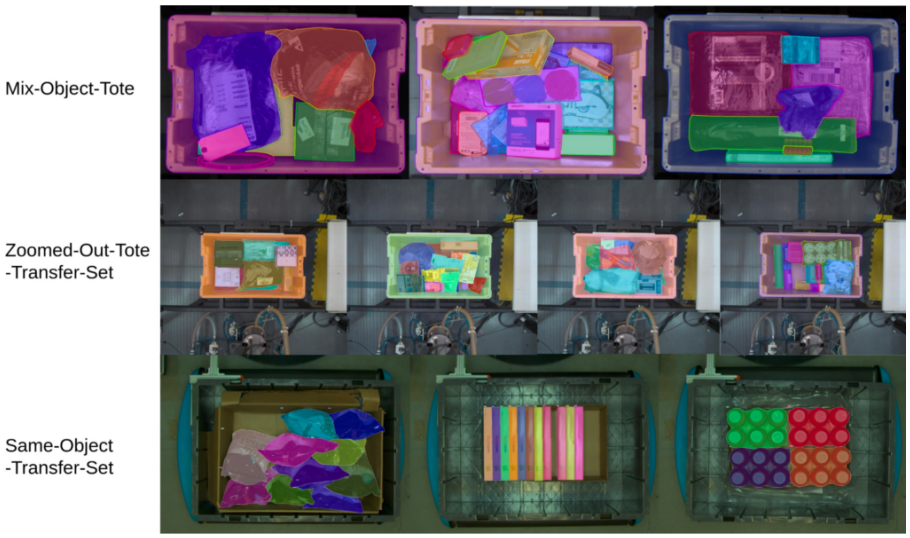


Figure 3.4: Object Segmentation Dataset of ARMBench [32]

3.4 Train, validation and test split

GraspNet

Given that GraspNet dataset does not come with a proper validation set, it was thought that splitting the entire training set in training and validation was a

reasonable choice for tuning the hyperparameters or simply confirming the chosen ones. In particular, referring to most of experiences of the DL community, the split was following: **80%** (20480 images) for the **training** phase of the model and the remaining **20%** (5120 images) for its **validation**. The test set has been already provided by the authors and in particular three different categories are available: seen, similar and unseen each of them containing 7680 images. The results on the test novel dataset are the most relevant ones.

ARMBench

Differently from GraspNet, in ARMBench the parting of the entire dataset is already provided with train, validation and test sets. For the training part 27634 images are available while for the validation 6001 images, thus keeping a similar split adopted for GraspNet ($\approx$ **82%** for **train** and $\approx$ **18%** for **validation**).
The test set instead includes 5944 images so it is similar to the validation one in terms of dimensions. In this case the variety of object categories is so huge that it was not possible to check precisely the seen/unseen categories present in the test with respect to train dataset, on which the model was trained. But the scope of the ARMBench authors was just to make the model able to learn the most in terms of objects heterogeneity so it should not be considered a problem in this case.

3.5 Adopted choice for the problem

Starting from the multi-class instance segmentation approach GraspNet was first considered as the base dataset adopted for the required training algorithms. This is because having also the grasping labels it will allow to determine the accuracy of the entire pipeline, either using the proposed method or more sophisticated ones. In addition the dataset is provided with depth camera's acquisitions, necessary for one of the tested models. Furthermore it represents the table picking task and it is very label-rich. Another advantage in taking GraspNet as reference dataset is its dimension compared with the one of ARMBench that allowed us to get the training results in an affordable way in terms of computational time. The next experiments showed in Chapter 5 are almost entirely obtained with GraspNet using only the Realsense's images, since the physical setting uses the same camera. ARMBech was then exploited to refine the results, given its wide variety of object categories.

Chapter 4

Hardware description

4.1 GPU for ML training

ML algorithms concerning the DL approach always require a high computational cost due to the update of the model parameters in the training phase with the backpropagation of the gradient.
To avoid such an issue we have the possibility to realize parallelized computations of the total update by means of the batch structure described in Section 1.2.1. Changing the kind of processor from a classic CPU to a proper GPU this problem becomes feasible from a computational point of view. The NN trainings were realized at Comau thanks to the availability of a Quadro M4000. As reported in [33], there are more sophisticated hardware like Tesla K40c, but considering Quardo M4000 a standard GPU, it was quite sufficient for most of the total work that was carried out.

4.2 RealSense D435i for image acquisition

For the real experiments we relied on a quite common but effective sensor, a RealSense D435i camera. Some of the most significant reason motivating this choice were particular advantageous both for the acquisition specific task and for a practical point of view.
The integrated RGB camera captures high-resolution color images, allowing visual data acquisition alongside depth data. This is beneficial for applications that require both depth and color information, such as augmented reality, 3D mapping, and computer vision tasks. It uses stereo vision to capture depth information combining a pair of cameras with the RGB camera to create a 3D depth map of the environment, as shown in Fig. 4.1. Moreover, the D435i features a wide field of view, which makes it suitable for capturing larger scenes and environments, such

as indoor spaces, robotics, and drone applications.



((a)) Camera [34] **((b))** Camera specific [35]

Figure 4.1: Realsense camera visualization

In addition to technical reasons, the camera's compact size and lightweight design make it easy to integrate into various setups and devices like the end-effector of the robotic arm that was adopted and the low price guarantee the possibility to be used by a large computer vision community.

4.3 Racer-3 industrial robotic arm

The grasping setting made use of a Racer-3 robotic arm by COMAU, available in the office, reported in Fig. 4.2.



Figure 4.2: Racer-3 anthropomorphic arm [36]

Given its quite small dimensions, it is the basic robot used for testing and improving new designs. For our experiment it was enough; it needed however the addition of a final shaft to host both the camera and the end-effector: a simple suction cup to pick up the detected objects. More details on the reference frame are given in Section 6.1.

Chapter 5

Segmentation method

The chapter provides quantitative results of the models analyzed in Chapter 2 applied to the datasets of Chapter 3. The settings of the training and validation phases are explained, and the results from testing are studied in terms of both accuracy and computational time. The best trade-off model is picked to continue with the next phase of the pipeline.

5.1 Multiclass semantic segmentation

This section describes the training settings and the results obtained for the semantic segmentation models described in Section 2.1.

5.1.1 Setting and hyper-parameters configuration

The main configuration decisions are reported in Tab. 5.1 As general settings, both for U-Net and PSPNet models, a standard configuration was applied, motivated by the fact that similar choices have been taken in other training examples of multiclass semantic segmentation, and also for practical reasons, concerning the resources available. For the validation phase the adopted transformations are mainly the resize and the normalization of the image with a single image per batch. Even though such configurations are quite relevant design choices, they are not all the possible ones. In order to accomplish this demand a grid search was developed by taking into account the remaining hyper-parameters, as discussed in detail in Section 5.1.2.

SETTINGS	TRAINING CHOICE
BATCH SIZE	16
LEARNING RATE	0.0001
N° EPOCHS	60
IMAGE SIZE	360 x 640
TRANSFORMATIONS	Resize, Random Crop, Rotation, Flip, Color Jitter and Normalization
OPTIMIZER	Adam
N° WORKERS	8

Table 5.1: Adopted configuration for the training phase of the multi-class semantic segmentation models

5.1.2 Training loss and validation accuracy

U-Net performances

Here the results of the U-Net model are explained in terms of training loss and validation accuracy with the addition of representative charts. As above-mentioned, before presenting the final outcome a grid search was performed by changing different hyperparameters, such as the learning rate and its scheduling variables. In this case a linear step schedule was adopted where, after a particular period expressed in terms of number of epochs, the learning rate was decreased by a scalar factor γ. The two main cases that gave the best performance are highlighted:

- **Learning rate $\rightarrow$ 0.0001 , Step size $\rightarrow$ 30 , $\gamma \rightarrow$ 0.5**
- **Learning rate $\rightarrow$ 0.0001 , Step size $\rightarrow$ 20 , $\gamma \rightarrow$ 0.5**

As shown in Fig. 5.1, they reached quite similar results and for a small percentage the second configuration proved to be the optimal one.

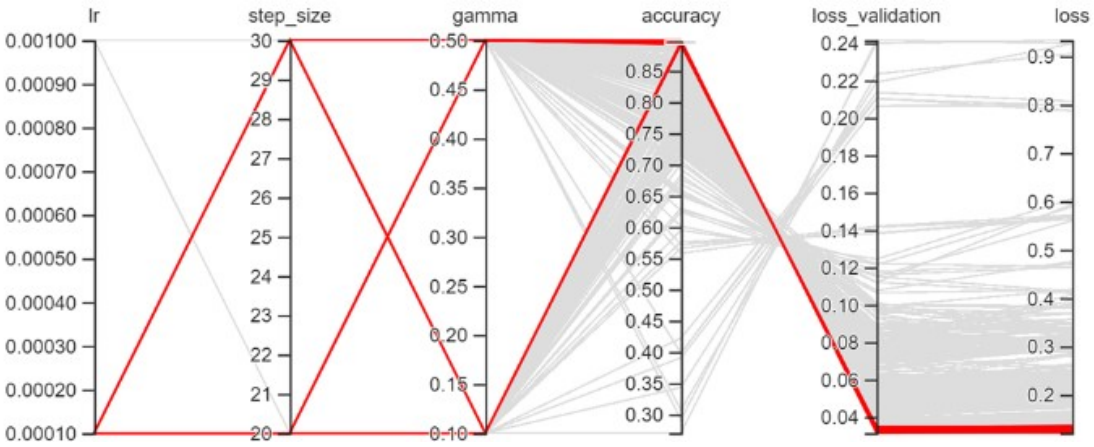


Figure 5.1: Grid search resuming table

The training loss and validation accuracy of each model from the grid search are represented in Figs. 5.2 and 5.3, respectively. As it can be seen in the figures, the effect of the step size and its amplitude, namely γ, have a great influence on the final result. In fact, decreasing the learning rate means to more likely approaching a possible local minimum of the loss function, that not necessarily is a good one. Thus, $\gamma = 0.5$ is a good trade-off to get closer to minimum without being stuck in case it is a 'bad' one. Moreover, stepping twice (step size = 20) in the training phase enables the learning model to get closer to the minimum, slightly better than just with one single step (step size = 30).

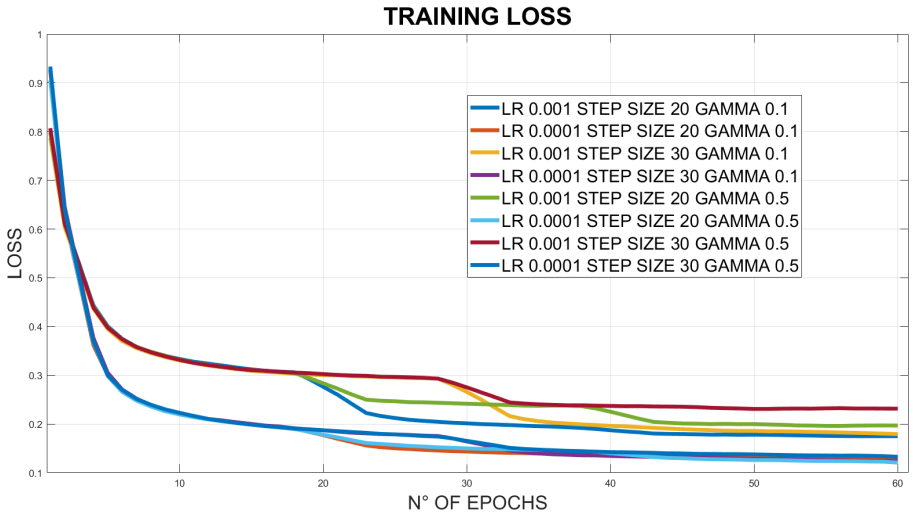


Figure 5.2: Training losses plot of U-Net model

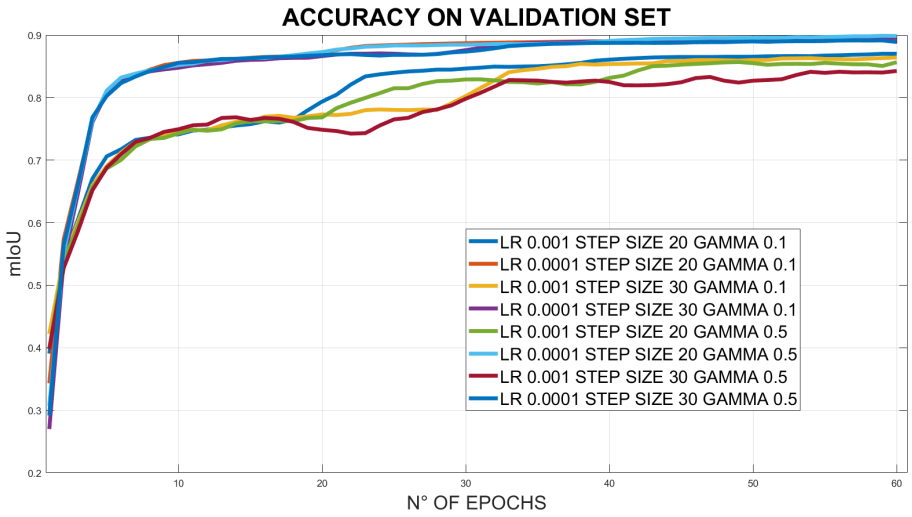


Figure 5.3: Validation accuracy plot

It must be underlined that the validation accuracy was computed taking into account one image out of ten to not spend too much time during this phase. This may lead to a slightly different real outcome, but still reliable.

PSPNet performances

The same grid-search has been performed also on such model. The resulting best models are:

- **Learning rate $\rightarrow$ 0.0001 , Step size $\rightarrow$ 30 , Gamma $\rightarrow$ 0.5**
- **Learning rate $\rightarrow$ 0.0001 , Step size $\rightarrow$ 20 , Gamma $\rightarrow$ 0.5**

Figs. 5.4 and 5.5 plot the training loss and the validation accuracy for the grid search, respectively.

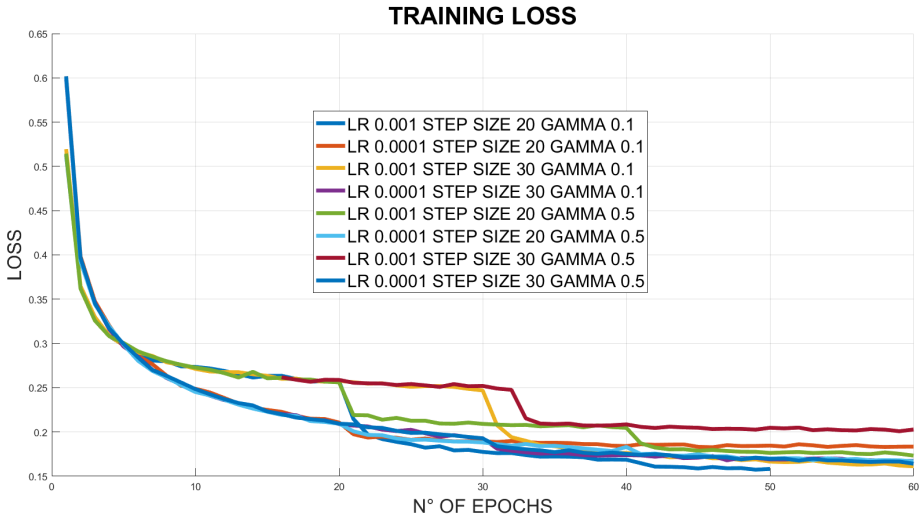


Figure 5.4: Training losses plot of PSPNet model

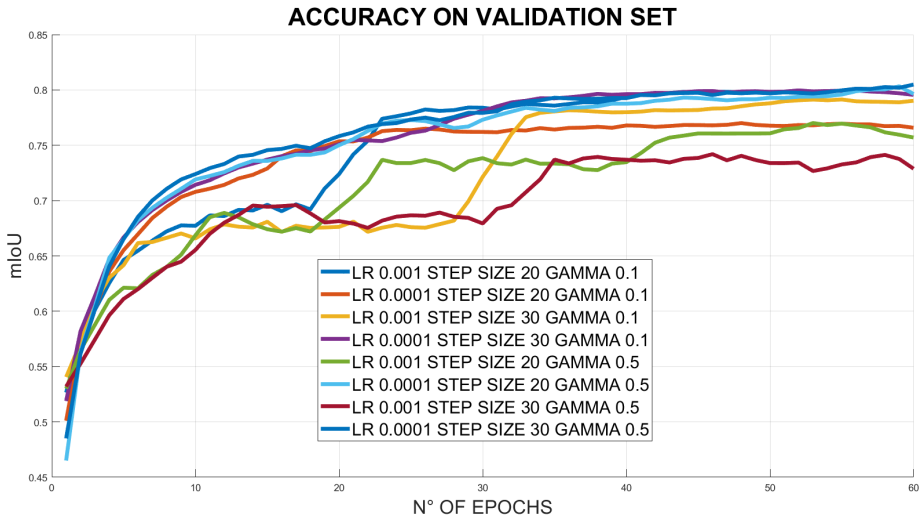


Figure 5.5: Validation accuracy plot

Similar considerations of U-Net can be done for these results as well.

5.1.3 Accuracy on test dataset

For the final performance the mIoU was computed only over the test seen dataset of GraspNet. This is because a multi-class semantic segmentation task will lead to poor results in the case of testing on a partial or total unseen dataset. In Tab. 5.2

the main key aspects are resumed, these are helpful for future comparisons.

	U-Net	PSPNet
MODEL PARAMETERS (M)	$\approx$ **14.3**	$\approx$ **10**
mIoU on TEST (%)	**80.3**	**69.1**
mIoU on VALIDATION (%)	**89.9**	**79.8**
CPU inference time (image/s)	**0.78**	**0.89**

Table 5.2: Results comparison

Examples of the goodness of the semantic segmentation models are reported in Figs. 5.6 and 5.7, with reference to a random test image.

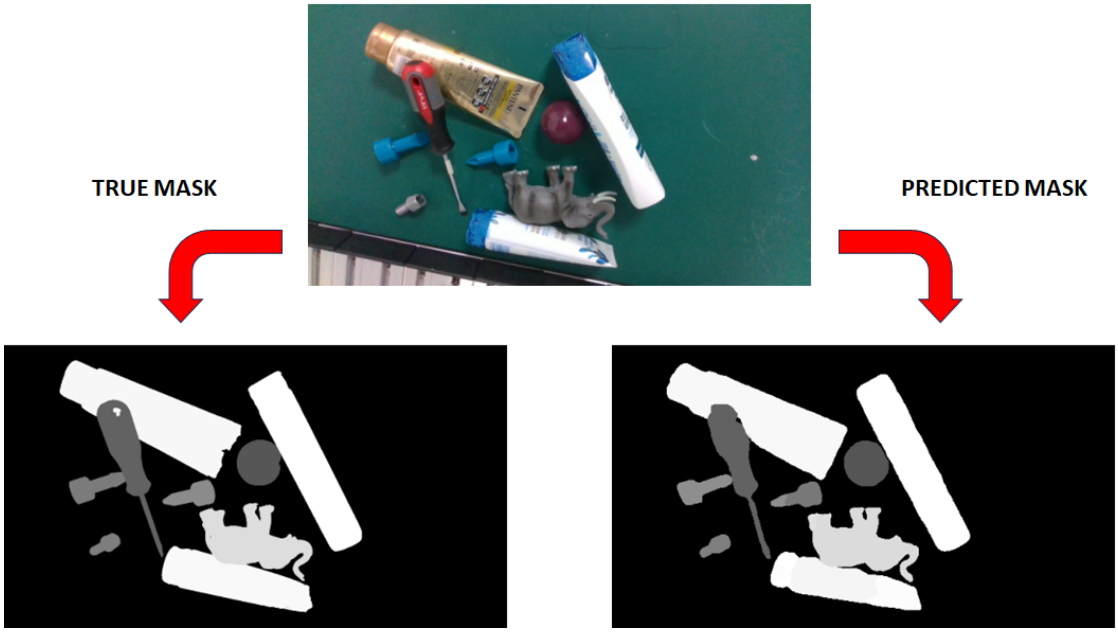


Figure 5.6: Target vs predicted masks with U-Net on a test image

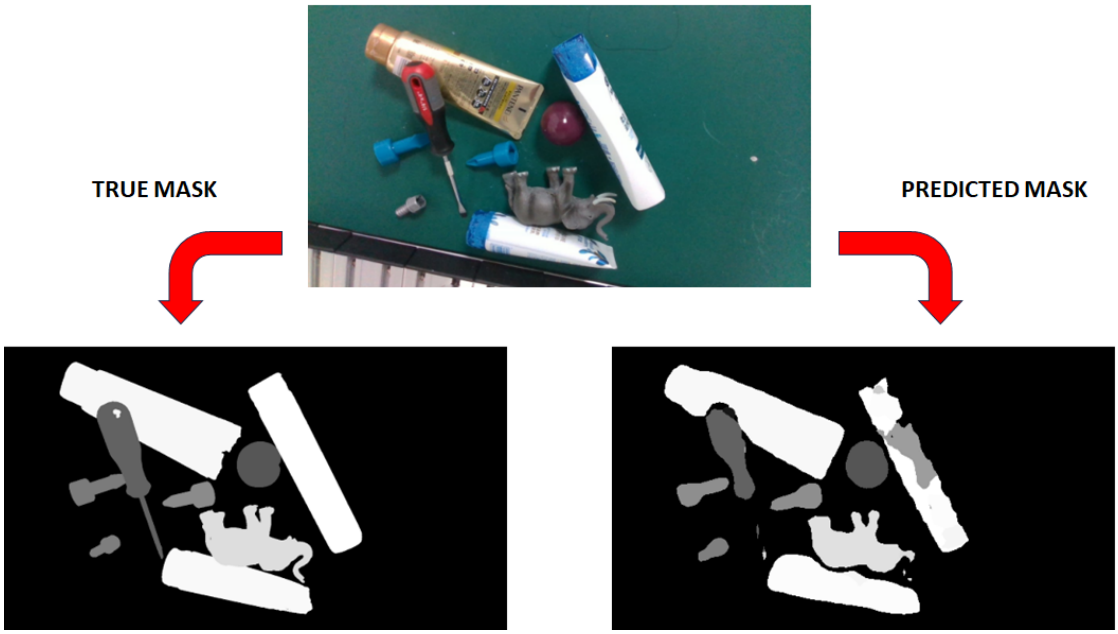


Figure 5.7: Target vs predicted masks with PSPNet on a test image

5.2 Binary object detection

This section describes the settings and the results of the training and testing of the binary object detectors models, presented in Section 2.1.

5.2.1 Setting and hyperparameters configuration

Differently from the multi-class semantic segmentation task, for binary object detection training it was kept the previous best configuration got via grid search, with some small changes in the image transformations part. This because, by comparing other detectors training, it was noticed that most of the hyperparameters were quite similar to ours. For sake of completeness the overall configuration is reported in Tab. 5.3. The main differences with the previous configuration are the batch size and the image transformations. The former has been halved given the higher model weight in terms of network parameters and a larger GPU RAM usage. Image transformations have been chosen suited for this kind of task, like some of the reported ones in [37].

SETTINGS	TRAINING CHOICE
BATCH SIZE	8
LEARNING RATE	0.0001
STEP SIZE	20
GAMMA	0.5
N° EPOCHS	60
IMAGE SIZE	360 x 640
TRANSFORMATIONS	Resize, Random Brightness Contrast, Flip, RGB Shift and Normalization
OPTIMIZER	Adam
N° WORKERS	8

Table 5.3: Adopted configuration for the training phase of the binary object-detection models

5.2.2 Training loss and validation accuracy

Faster R-CNN performances

Figs. 5.8 and 5.9 plot the total loss function and the validation precision for Faster-RCNN model, respectively. From the training point, it can be noticed how the learning rate and step size makes the models able to learn more after each 20 epochs. A possible improvement could be increasing the number of steps (as demonstrated in the previous section) in accordance with a larger number of epochs. This is done to allow the loss to reach the no-descent condition, where the model does not learn anymore. As it can be seen, the AP_{50} and AP_{75} show great results leading to significant accuracy on GraspNet dataset.



Figure 5.8: Training loss plot of Faster R-CNN model

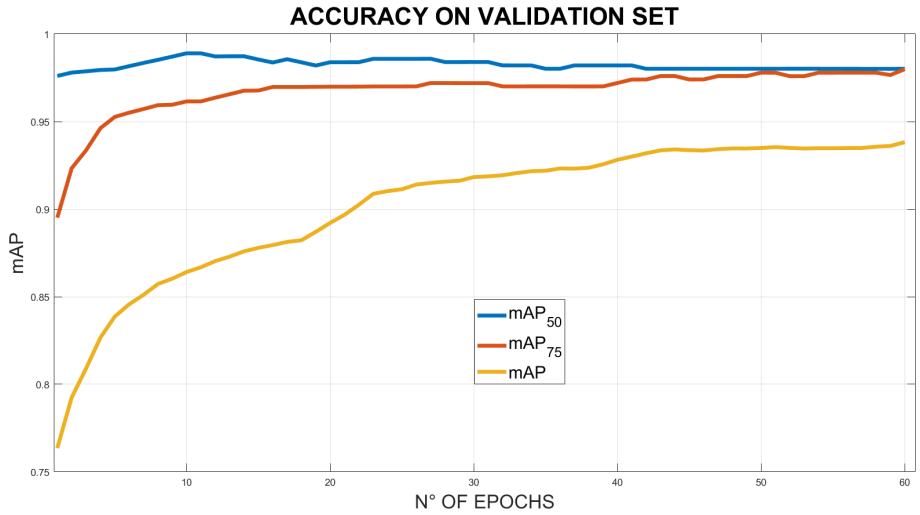


Figure 5.9: Validation accuracy plot of Faster R-CNN model

SSD performances

Figs. 5.10 and 5.11 show the training loss and validation AP of SSD, respectively.

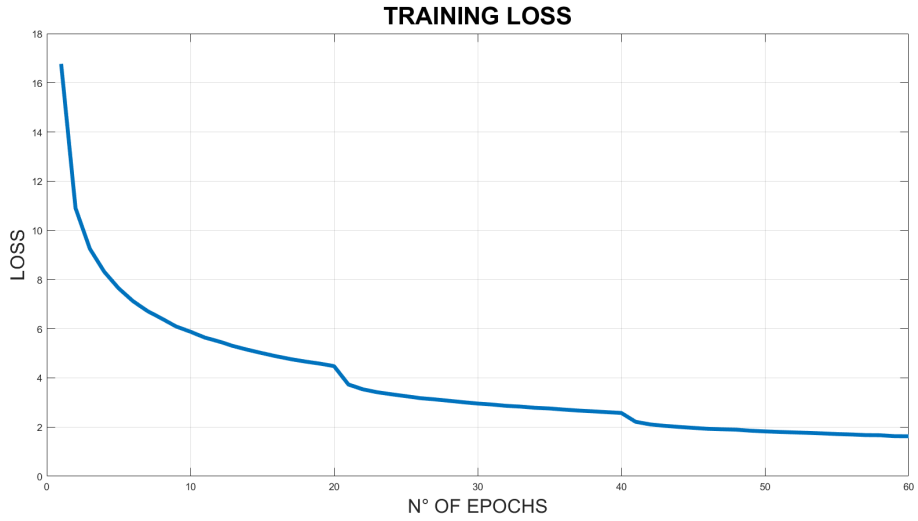


Figure 5.10: Training loss plot of SSD model

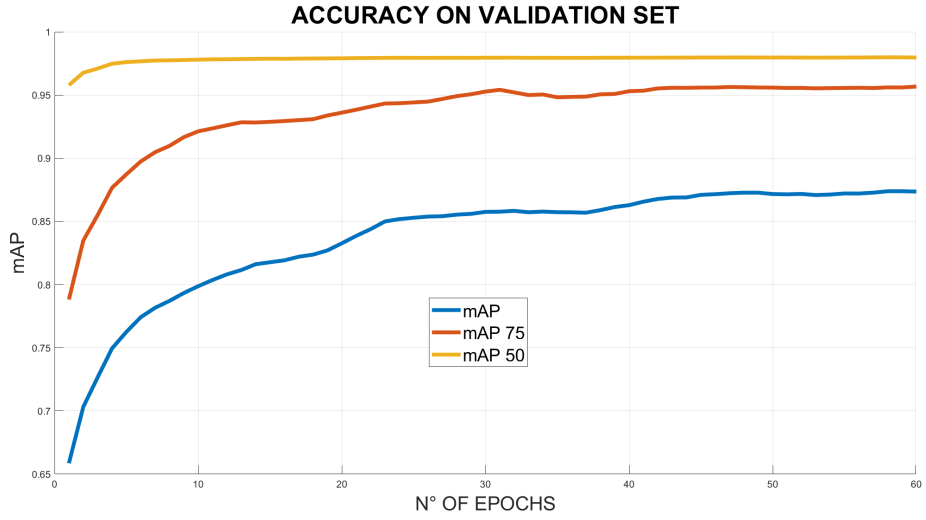


Figure 5.11: Validation accuracy plot of SSD model

Accuracy on test dataset

Differently from multi-class semantic segmentation, where the model is only able to deal with the same categories of object present in the training set, for binary object detection also the accuracy results in the 'test similar' and 'test novel' datasets of GraspNet were taken into account. The main reasons are two:

- binary mode allows a better generalization, having to face just object-background

categorization.

- object detection's task is simpler with respect to semantic segmentation, because predicting a box containing a target is less difficult than predicting each pixel of the same target.

In the following tables all the performances in terms of AP and processing time are reported. The AP metrics are evaluated for the three test datasets of GraspNet. Tab. 5.4 shows how SSD is in general faster and lighter with respect to Faster-RCNN, this is a characteristic of other single shot detectors.

MODEL	MODEL PARAMETERS (M)	CPU inference time (s/image)
Faster R-CNN	≈ 41	≈ 5
SSD	≈ 30	≈ 2

Table 5.4: Model weights and inference time on Intel Core i7-1165G7s

The results in Tab. 5.5 show how the detected boxes are almost perfect when the model has to predict them with already seen objects. In fact, just from the AP_{50}, that is a quite reliable accuracy indicator, the difference between the validation case and the test one is less than 1%.

MODEL	**AP_{50} (%)**	**AP_{75} (%)**	**AP (%)**
Faster R-CNN	97.9	93.6	82.0
SSD	96.7	91.2	80.9

Table 5.5: Test seen accuracy object detectors

Changing the test dataset by passing from seen to similar objects, the performance starts decreasing, as highlighted in Tab. 5.6.

MODEL	**AP_{50} (%)**	**AP_{75} (%)**	**AP (%)**
Faster R-CNN	92.0	82.4	72.8
SSD	78.2	68.6	60.3

Table 5.6: Test similar accuracy object detectors

In the final case (Tab. 5.7) it can be seen how the accuracy is significantly decreased, but it can still be considered acceptable, as showed in Fig. 5.12, where in the model output most of the bounding boxes are properly predicted; here only the Faster R-CNN boxes are reported given its greater predictions compared with the SSD ones.

MODEL	**AP_{50} (%)**	**AP_{75} (%)**	**AP (%)**
Faster R-CNN	85.2	65.4	59.6
SSD	73.5	50.3	48.2

Table 5.7: Test novel accuracy object detectors

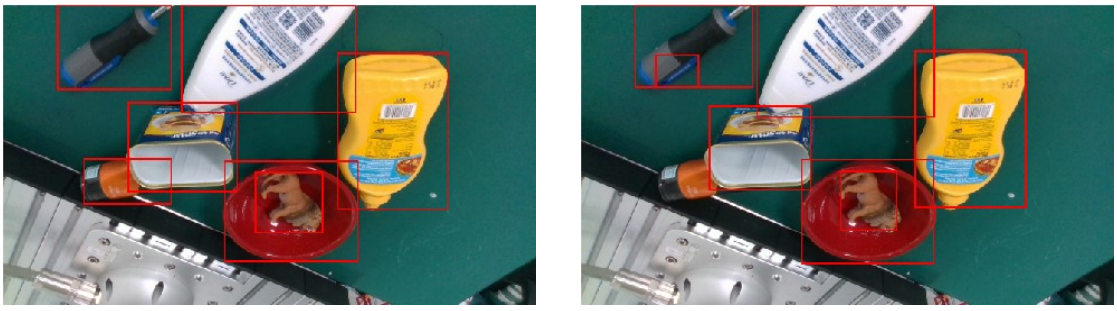


Figure 5.12: Target (on the left) vs Predicted (on the right) bounding boxes with Faster R-CNN model on test novel dataset

When predicting seen objects, the test accuracy is really high for all the models. Thus for such application they can all be used. SSD model is anyway the worst one, and such differences are highlighted especially in the test similar (Tab 5.6) and test novel (Tab 5.7) datasets. Thus, for such applications Faster-RCNN is suggested.

5.3 Binary instance segmentation

This section describes the training and testing results of the instance segmentation models, presented in Section 2.1.

5.3.1 Training Loss and Validation Accuracy

Mask R-CNN on GraspNet

Figs. 5.13 and 5.14 plot the total training loss and validation AP of MaskRCNN, respectively. Also in this case the use of learning rate scheduling helps to get better results in terms of loss minimization and accuracy improvement.

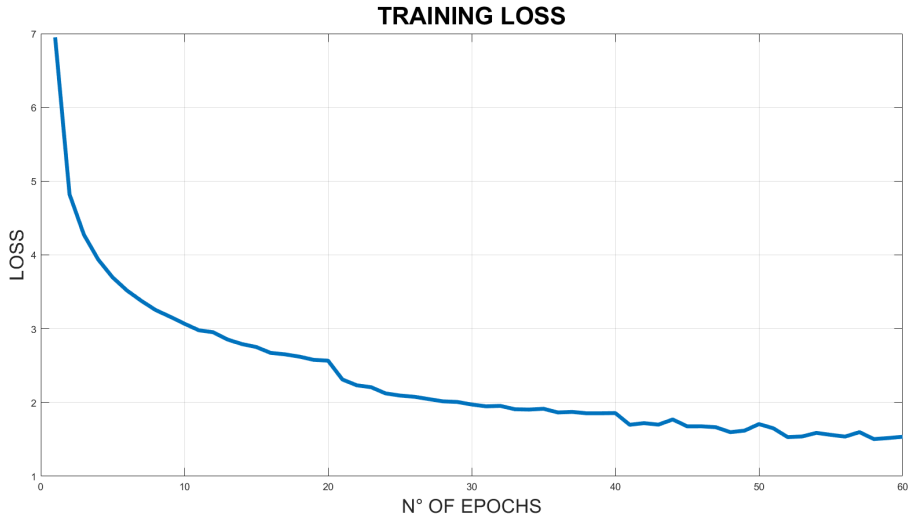


Figure 5.13: Training loss plot of Mask R-CNN model

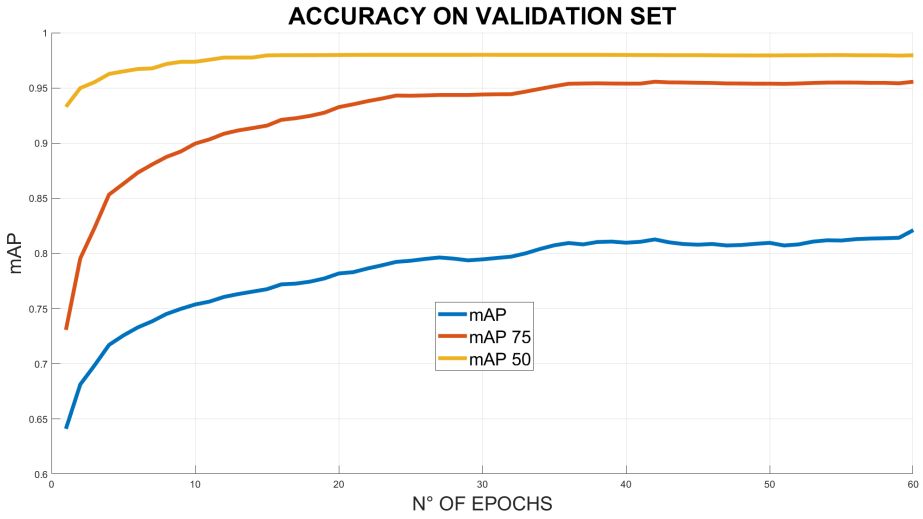


Figure 5.14: Validation accuracy plot of Mask R-CNN model

SF-Mask R-CNN

Figs. 5.15 and 5.16 show the total loss and AP of SF MaskRCNN, respectively. The obtained results are quite similar to the previous ones as it can be expected given the same core structure.

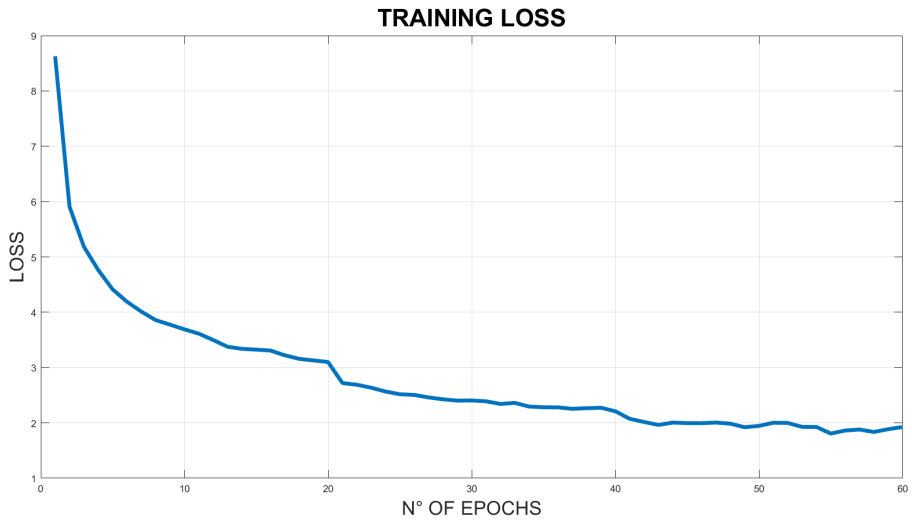


Figure 5.15: Training loss plot of SF-Mask R-CNN model

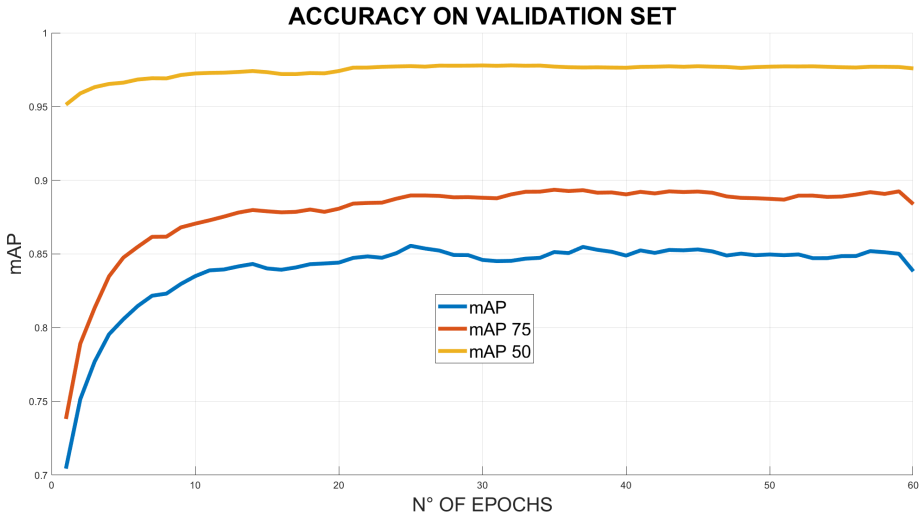


Figure 5.16: Validation accuracy plot of SF-Mask R-CNN model

5.3.2 Accuracy on Test Datasets

Tab. 5.8 reports the total segmentation times of the models and the weights. It can be seen how the usage of SAM notably increases the segmentation time, with respect to the base detector or segmentation network.

MODEL	Model Parameters (M)	CPU inference time (s/image)
Mask R-CNN	≈ 44	≈ 5
SF-Mask R-CNN	≈ 168	≈ 6.1
SAM	≈ 641	≈ 42

Table 5.8: Model weights and inference time on Intel Core i7-1165G7s

Tabs. 5.9, 5.10 and 5.11 present the results of the models on the Seen, Similar and Novel datasets, respectively. It must be marked that the introduction of SAM shifts the attention towards finding a good object detector. In fact the accuracy of the SAM's segmentation masks are close to the ones of the respective bounding boxes. Thus, the better the detection the better the segmentation results will be. This comes at the cost of a higher computational time, which is not justifiable if the model will only be applied with seen or similar objects. However, it will bring great improvements with unseen objects. It is also interesting to notice that the use of depth information increases the AP 50 and the AP75 in both the test similar and test novel.

MODEL	**AP_{50} (%)**	**AP_{75} (%)**	**AP (%)**
Faster R-CNN + SAM	96.9	87.9	73.1
Mask R-CNN	**97.6**	**89.4**	**76.1**
Mask R-CNN + SAM	97.4	88.9	75.1
SF-Mask R-CNN	97.0	88.5	73.5
SF-Mask R-CNN + SAM	96.8	87.9	73.5
SSD + SAM	96.5	86.1	72.3

Table 5.9: Test seen accuracy instance segmentation models

MODEL	AP_{50} (%)	AP_{75} (%)	AP (%)
Faster R-CNN + SAM	92.3	83.8	72.5
Mask R-CNN	89.2	77.4	67.7
Mask R-CNN + SAM	90.4	82.6	73.2
SF-Mask R-CNN	92.3	78.2	69.0
SF-Mask R-CNN + SAM	**93.3**	**84.3**	**74.0**
SSD + SAM	78.9	71.5	63.8

Table 5.10: Test similar accuracy instance segmentation models

MODEL	AP_{50} (%)	AP_{75} (%)	AP (%)
Faster R-CNN + SAM	83.8	64.0	57.5
Mask R-CNN	75.9	50.5	48.6
Mask R-CNN + SAM	83.8	**64.7**	**59.9**
SF-Mask R-CNN	77.7	52.4	48.5
SF-Mask R-CNN + SAM	**84.6**	61.7	56.8
SSD + SAM	73.1	51.5	49.2

Table 5.11: Test novel accuracy instance segmentation models

5.3.3 Pro and cons of Mask R-CNN and SAM

This section resumes the final positive and the negative aspects regarding the utilization of either Mask R-CNN or SAM model.

Mask R-CNN

Regarding the instance segmentation model, it must be taken into consideration that it is a multitasking model, tackling both segmentation and object detection. Thus, the overall model will be larger and the training phase may require more time

and a more powerful hardware. The results will not be always the desired one but still acceptable (at least for our experiments). The main advantage of not being so precise is that by using a model like Mask R-CNN gives a total end-to-end learning. Moreover, the much lower computational time with respect to SAM allows the network to be used when a good trade-off between performances and inference time is necessary.

SAM

The SAM approach is based instead only on the training of an object detector: this allows just to train a lighter model and it gives also the possibility to better explore the state-of-the-art networks responsible for this specific task, in order to predict more reliable bounding boxes. Beyond this first block, there is also the advantage of having SAM as segmentation model: as previously mentioned, it gives the user the possibility to prompt the precise box and obtain the corresponding segmentation mask given its class-agnostic property. In reality, this works only if the user can manually assign the box at each iteration selecting the proper grade of segmentation; in an automatic fashion instead, we can only rely on the predicted box, hoping that this one will completely segment the item, which is not always guaranteed in practice. The negative aspect that led us to realize most of the test cases with Mask R-CNN was the inference time of SAM: the model takes about 42 seconds with CPU. This is an absolute 'waste' of time, if we think that just with a single image acquisition it is not guaranteed the complete segmentation of all the objects in a highly cluttered scene.

5.3.4 U-Net vs Mask R-CNN

Another comparison that justifies the transition from the multi-class semantic segmentation approach to the binary instance one is that it guarantees a higher accuracy. Once the performance of the Mask R-CNN on the GraspNet test seen set were derived the U-Net results were reported in the same framework (instance segmentation), given the possibility to identify each mask by its own category. In this way the multi-class output was converted in a binary one, allowing the accuracy measurement in terms of AP. In Tab. 5.12 the outcome is reported.
As it can be noticed, there is almost a 30 % difference in all the three AP cases, thus confirming the previous hypothesis. Furthermore, the binary masks of U-Net are directly derived from the semantic segmentation context, where the predictions may contain small wrong objects (corresponding pixels may represent a particular area of the background). This increases the number of FP in the new binary instance segmentation environment.

MODEL	AP_{50}	AP_{75}	AP
UNet	66.6 %	51.4 %	44.7 %
Mask R-CNN	97.6 %	89.4 %	76.1 %

Table 5.12: UNet vs Mask R-CNN accuracy in AP metrics

5.3.5 Comparison with ARMBench-trained model

At this point relevant results have been reached, especially in the object detection-instance segmentation part. It was seen how Mask R-CNN gave a quite satisfactory solution in order to develop the final segmentation-based pipeline. Since it can be used both as an end-to-end model and in a two step model, it was chosen to continue with this architecture. To improve its performance, by realizing a robust model being more class-agnostic, the ARMBench dataset was adopted as training set. This because it contains specific images of bin-picking scenes with an enormous quantity and variety of objects, as reported in Section 3.3, hoping that such an advantageous characteristic would make the final model more powerful. Despite the base SF-MaskRCNN gave in general better AP, the absence of depth information in the dataset prevented its further refinement. First of all a full training with ARMBench is required in order to have a benchmark to start from. In Figs. 5.17 and 5.18, the training and validation performances are shown, using as hyper-parameters configuration the same adopted in Section 5.2.1.

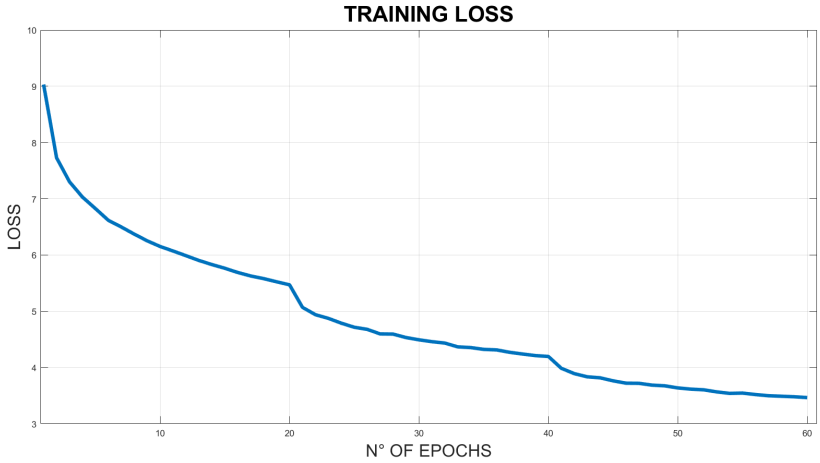


Figure 5.17: Training loss plot of Mask R-CNN model trained on ARMBench dataset

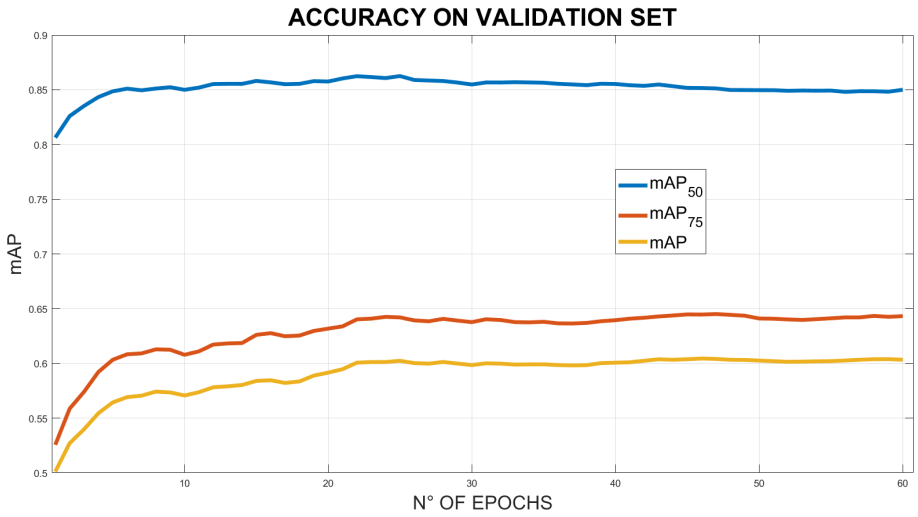


Figure 5.18: Validation accuracy plot of Mask R-CNN model trained on ARM-Bench dataset

It is evident how the overall performance is lower with respect to the model trained on just GraspNet: this is simply because ARMBench is a more 'difficult' set in the sense that the huge variety in object categories and the large number of objects per image do not allow to immediately get a good outcome. Hence, in order to be independent from the related validation set, an impartial comparison was made by testing the best Mask R-CNN model, got with GraspNet, on the ARMBench validation set and the inverse operation , that is the best model trained on ARMBench and tested on GraspNet. In this way one can define which of the two models is able to better generalize the required task. Tab. 5.13 compares the behaviour of the two models. Given that a model trained on ARMBench is used to detect a wide range of different kind of objects, it has no big issues in validating the GraspNet set, even though this is not its training set. Obviously, the accuracy outcome is not so high, but it is a great starting point to achieve a class-agnostic binary instance segmentation model. On the contrary, as GraspNet presents a limited number of specific objects, its corresponding model has no possibility to effectively detect the new items. The reported AP for simplicity is only referred to boxes detection.

ACCURACY	$\mathbf{AP_{50}}$ (%)	$\mathbf{AP_{75}}$ (%)	AP (%)
GraspNet Model on ARMBench	8.3	3.3	3.9
ARMBench Model on GraspNet	**45.9**	**24.7**	**25.5**

Table 5.13: Comparison between GraspNet and ARMBench models

In the following section the two models are mixed up, in terms of training, in order to take full advantage of both dataset peculiarities.

5.3.6 Refinement using both datasets

A final try to get the best model for binary instance segmentation consisted in implementing a training based on a mixed dataset, which shared both GraspNet and ARMBench information. This procedure is quite common in DL training algorithms due to the fact that in this way the final model will be able to face significant issues like class imbalances and different semantic level of detail, as also experienced by P. Meletis and G. Dubbelman in [38] for semantic segmentation purposes.
In particular, starting from the best model trained on ARMBench (given its higher performance with respect to the GraspNet one), a batch of images was taken by the two datasets, always being consistent with the default setting (with batch size = 8). In this case, due to the different length of the two dataset, it was not possible to have a perfect balance of the data and so the shorter training set was taken (the GraspNet one) and it was added an equal portion of the other. Since the difference is not so huge, the discarded images do not play a key role on the overall result.
Figs. 5.19, 5.20 and 5.21 plot the AP for GraspNet, ArmBench and the weighted sum. As highlighted in the plots the final values, related to the different AP metrics, are quite similar to the previous ones, where the model was trained on a single dataset. In particular, the model accuracy on ARMBench dataset is basically the same ($\approx 61\%$ for the highest global AP). For the GraspNet part instead it is interesting to see how in Section 5.3.5 the final AP_{50} was $\approx 45\%$, while at the first epoch of the new training it reaches $\approx 75\%$: this is due to the fact that the model has already seen the set one time (when the first epoch is completed), hence it is able to better distinguish the GraspNet objects. The rapid increase in terms of accuracy is given also to the non-heavy heterogeneity characterizing this dataset, allowing the model to easily recognize GraspNet items after few epochs.

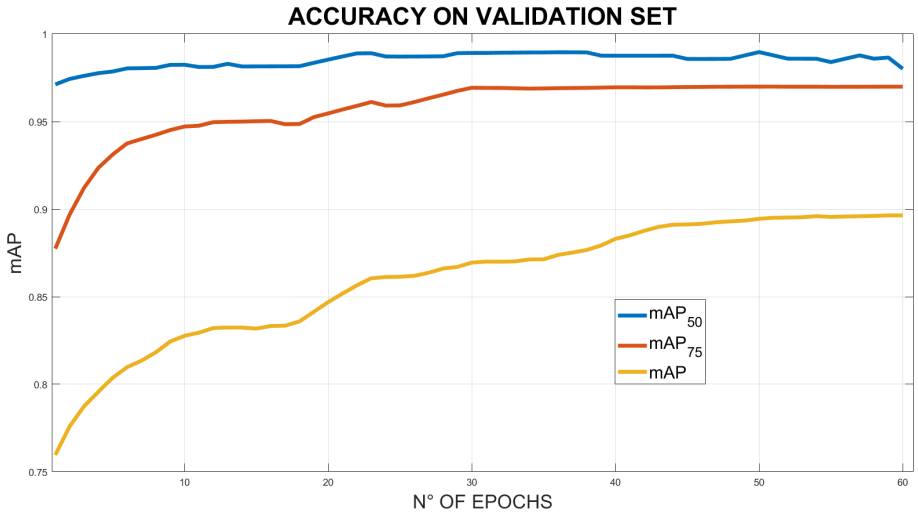


Figure 5.19: Validation accuracy on GraspNet

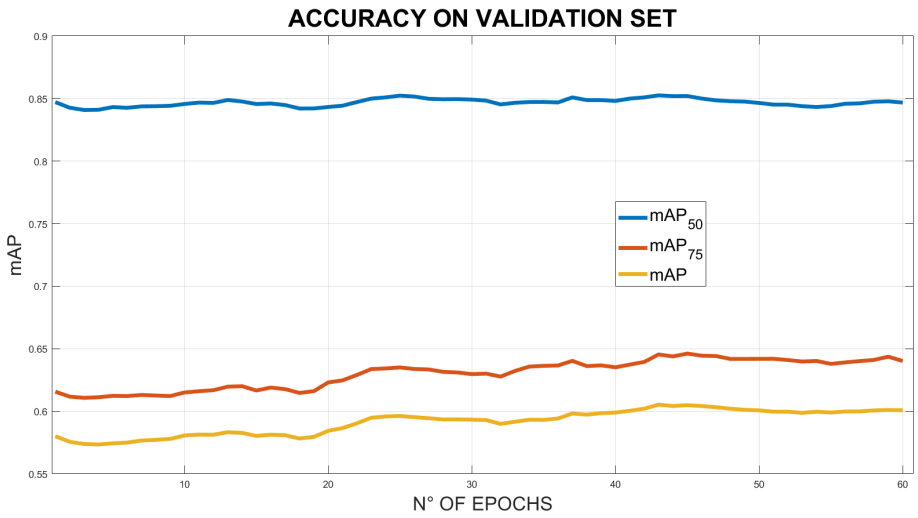


Figure 5.20: Validation accuracy on ArmBench

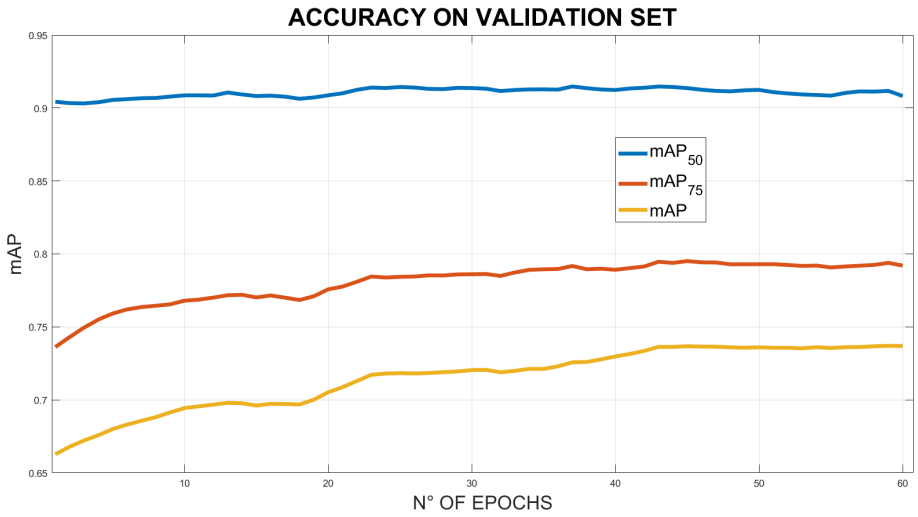


Figure 5.21: Validation accuracy on weighted combination of GraspNet and ArmBench

Tabs. 5.16, 5.15 and 5.14 provide the APs for the Mask R-CNN architecture trained on GraspNet, ARMBench and the combination of the two, applied to ARMBench Test, GraspNet Test Novel and their combination. As it can be seen, for two of the three tests the model trained on both datasets is the optimal one. In addition, from Tab. 5.15 the mixed model also increases the performances of Mask R-CNN trained on GraspNet. Based on this considerations, the mixed model has been chosen to continue with the next steps of the pipeline.

MODEL	**AP_{50}**	**AP_{75}**	**AP**
Mask R-CNN - GraspNet	5.8 %	2.7 %	3.1 %
Mask R-CNN - ARMBench	**80.0** %	**62.8** %	**57.1** %
Mask R-CNN - Mixed	79.1 %	61.8 %	56.3 %

Table 5.14: Accuracy on ARMBench Test

MODEL	**AP$_{50}$**	**AP$_{75}$**	**AP**
Mask R-CNN - GraspNet	75.9 %	50.5 %	48.6 %
Mask R-CNN - ARMBench	19.9 %	10.0 %	10.5 %
Mask R-CNN - Mixed	**82.2** %	**60.1** %	**54.0** %

Table 5.15: Accuracy on GraspNet Test Novel

MODEL	**AP$_{50}$**	**AP$_{75}$**	**AP**
Mask R-CNN - GraspNet	44.4 %	29.1 %	27.4 %
Mask R-CNN - ARMBench	50.0 %	36.4 %	33.7 %
Mask R-CNN - Mixed	**79.5** %	**59.2** %	**53.3** %

Table 5.16: Accuracy on combination of GraspNet Test Novel and Test ARMBench

Chapter 6

Grasping method

This chapter describes how to move from the segmentation masks, derived from the best model of the previous chapter, to the definition of the grasping pose. This includes the choice of the grasping point and the approaching attitude of the end effector. For simplicity, and coherence with the physical setting available, the following discussions will be based on the assumption of a suction gripper with a unique sucker.

6.1 Robot reference system

For the sake of clarity in Fig. 6.1 just the two reference frames, the Base and the Shaft ones, are reported. In fact just from the Shaft frame is quite easy to derive the other two systems (Camera and End Effector) with the addition of constant matrices in charge of translating the former to the desired ones. There are four main reference systems considered.

- End Effector frame
- Shaft frame
- Base frame
- Camera frame

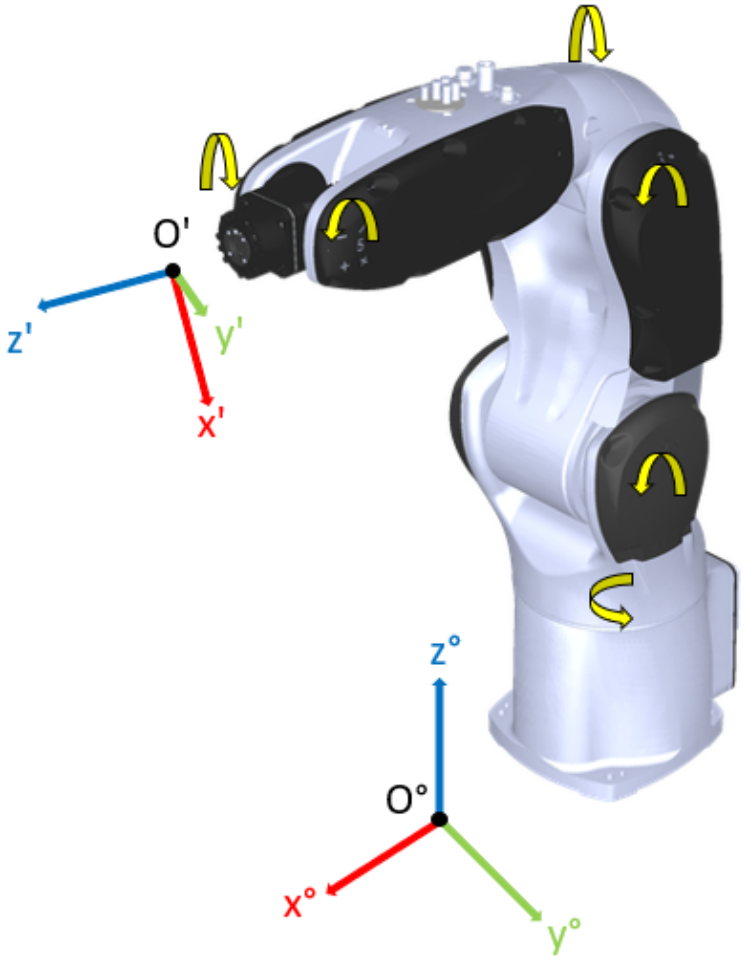


Figure 6.1: Robot main reference systems. At the top the local Shaft frame and at the bottom the fixed Base frame [39]

The RGB and depth images are taken with respect to the camera frame. The depth information is necessary to extract the point cloud for the computation of the approaching direction. The point cloud must be converted into the Base frame coordinates. The rototranslational matrices between camera and shaft and shaft and base are necessary for the conversion. The latter can be derived from the readings of the sensors on the robot. The former depends on the shape of the shaft, which is a rigid structure connecting the camera to the robot. Such matrix is obtained through a camera calibration procedure, which is denoted as hand-eye calibration, and performed via the OpenCV library. The idea is depicted in Fig. 6.2, where all transformation matrices are pointed out. A pattern is placed on a table and it is used to estimate the transformation between the target and the

camera, thus the matrix T_t^c. The camera is moved in different locations, each of them however must see the entire target for correct estimation of the matrix; the end effector poses with respect to the base frame T_g^b are saved. This procedure is repeated for various configurations.

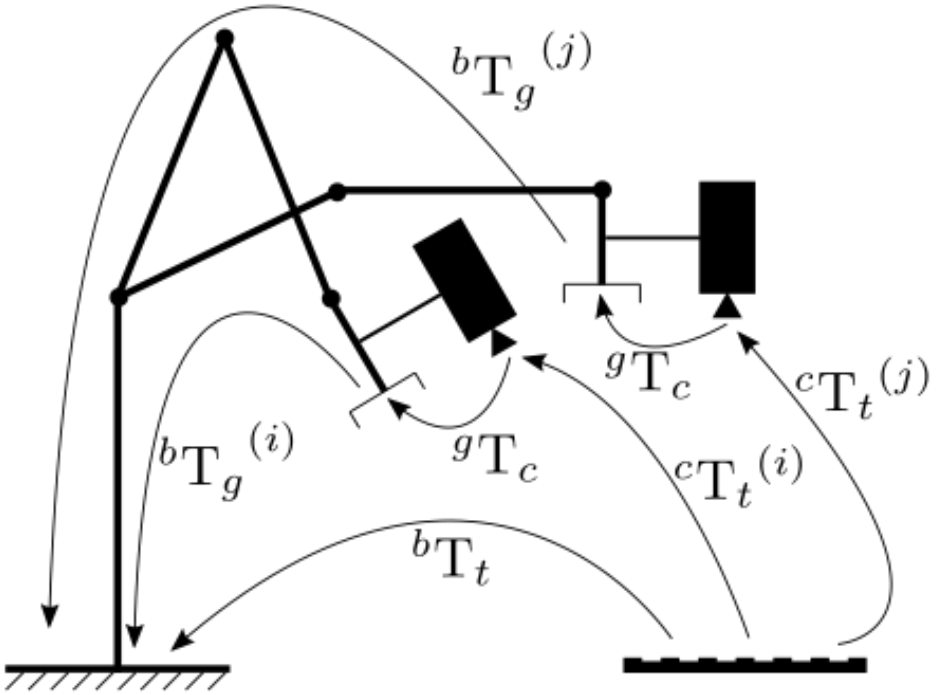


Figure 6.2: Hand to eye calibration scheme [40]

Since the target is fixed and the matrix between the end effector and the camera is constant, for any pair of distinct observation it is true that

$$T_g^{b,i} T_c^g T_t^{c,i} = T_g^{b,j} T_c^g T_t^{c,j}$$

which can be rewritten in the form of $AX = XB$ for any possible pair. The transformation matrices are indicated in Fig. 6.2. The target is a chessboard calibration pattern. This board is used to evaluate the camera intrinsic matrix (T_t^c) and also the camera distortion's parameters. Two errors are in fact possible:

- **Radial:** Straight lines appear curved, and the greater the distance from the centre of the camera the greater the distortion.

$$x_{dis} = x_{real}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{dis} = y_{real}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

where $r^2 = x^2 + y^2$. Fig. 6.3 gives an example of the error.

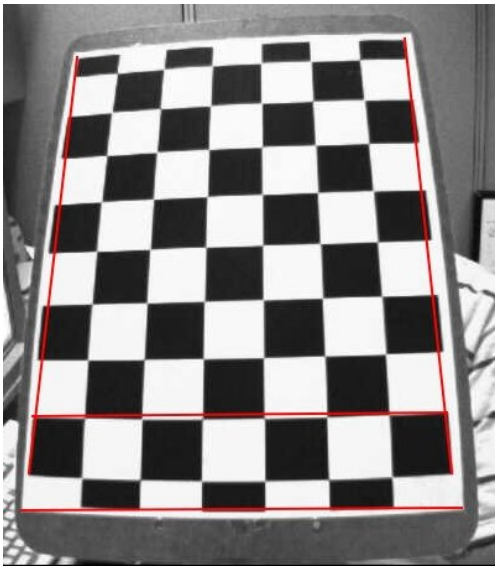


Figure 6.3: Camera radial distortion with calibration chessboard [41]

- **Tangential:** It occurs when the image place and the lenses of the camera are not parallel.

$$x_{dis} = x_{real} + (2p_1 x_{real} y_{real} + p_2(r^2 + 2x_{real}^2))$$

$$y_{dis} = y_{real} + (2p_2 x_{real} y_{real} + p_1(r^2 + 2y_{real}^2))$$

Thus the five distortion parameters $(k_1, k_2, k_3, p_1, p_2)$ are estimated, and accounting for the global distortion the poses with respect to the camera can be determined, hence the transformation is known.

6.2 From segmentation masks to grasping points

This module takes the segmented mask as input and returns the grasping point as output. Different approaches, also based on a neural network training such as the one proposed by A. Alliegro et al. [42], can be implemented. For simplicity an heuristic criteria developed by us was chosen, which it will be seen to work fine on most suction graspable objects, in the next chapter. The procedure consists in two steps:

- Find an equivalent centroid on the 2D segmented image for each binary segmentation mask.
- Among all the centroids found, the first picked will be the one closest to the camera.

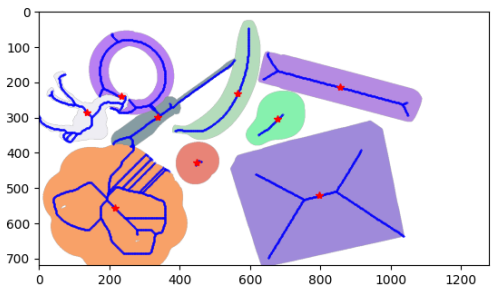
Thus, a point is chosen on a 2D image and is projected on the point cloud. Among all the instances the one with the grasping point having the minimum z coordinate is picked; in fact, as it will be seen later, the point cloud is expressed in the camera reference frame which has the positive z looking towards the scene. The choice of the point on the 2D mask is made as follows.

- Find the centre of mass of the segmentation mask, considering the holes of the depth image.
- Project the centre of mass on a sort of skeleton of the mask.

Some considerations are important. It is necessary to include also the depth information when choosing the 2D point, otherwise the selected grasping may fall in an area where the point cloud is not defined, due to some error in the depth acquisition. The idea of using the centre of mass as a possible point is interesting, since this is the point where there is the largest density of pixels, thus it is more likely to be graspable. This is true for convex shapes. However for concave ones the centre of mass may fall outside the figure. The skeleton is a sort of mid line abstracting the object shape, and should be placed almost at the middle of it. Hence, in the latter case the projection of the centre of mass on the skeleton will still lead to an internal and probable graspable point. This considerations are true supposing a perfect segmentation mask. There are several algorithms concerning the 'skeletonization' of a binary image like the one proposed by W. Abu-Ain et al. in [43]: it removes pixels starting from the border of the object until only a 1 pixel representation remains.. To reduce the procedure complexity the Sklearn library was used. In the examples of Figs. 6.4(b), 6.5(b) and 6.6(b) the blue lines are the skeletons while the red dots are the chosen points. The results display interesting advantages and drawbacks. At first, if the object is well positioned, i.e., the camera mainly sees only one of its faces, and the object has regular shape then the grasping point will probably fall in a graspable area. However the presence of inclined objects may lead to a grasping point choice that may coincide or be too closed to edges of the object. The usage of the centre of mass helps in the choice, as it can be seen on the screwdriver of Fig. 6.4(b). Lastly, the presence of unwanted branches in the skeleton may lead to choose a point too close to the boundary of the object. Possible future developments may be including some policy to account for the sucker area, or training a network specifically for grasping.



((a)) RGB image

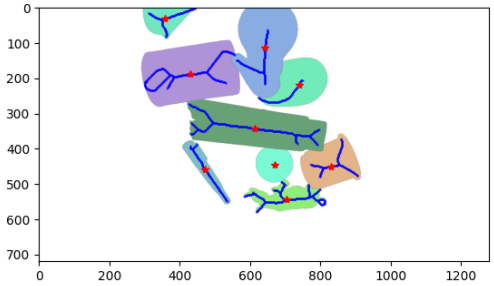


((b)) Skeleton and grasping point

Figure 6.4: Example 1 of Skeletonize and grasping point choice

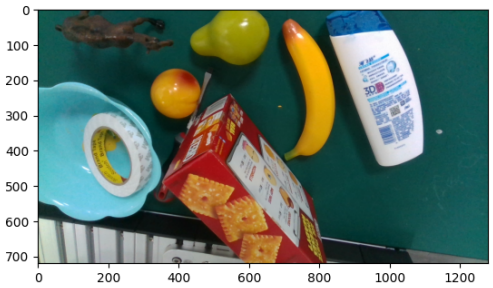


((a)) RGB image

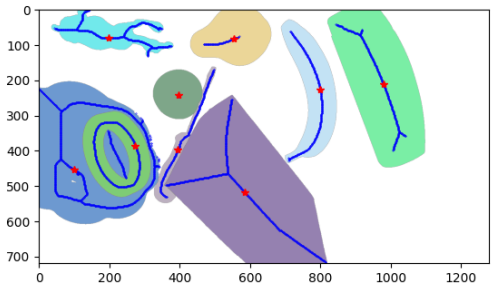


((b)) Skeleton and grasping point

Figure 6.5: Example 2 of Skeletonize and grasping point choice



((a)) RGB image



((b)) Skeleton and grasping point

Figure 6.6: Example 3 of Skeletonize and grasping point choice

6.3 Approaching direction to the grasping point

The approach attitude is based on the normal calculation, in fact a suction gripper requires the gripper to be parallel to the picking surface to allow the suction effect. To calculate the normal first a voxel-down-sampling of the point cloud has been performed. It consists in dividing the space in voxel of fix dimensions and considering for each a unique point given by the centroid of the points inside the voxel. Via this procedure the noise of the point cloud is reduced and the normal calculation is better. Subsequently the K closest points are found from the downsampled point cloud. Finally a plane has been fitted by applying the PCA [44], a quite reliable ML algorithm; to improve the accuracy an outlier robust estimator may be used. The normal direction was chosen entering the object, since the end effector has a frame with the exiting normal. Supposing a perfect depth, the lower the K the better is the normal approximation especially for curved surfaces. However, in case of noise in the depth estimate the value can be changed manually: since a too low value may capture local noise, while a too high value may lead to inaccurate normals.

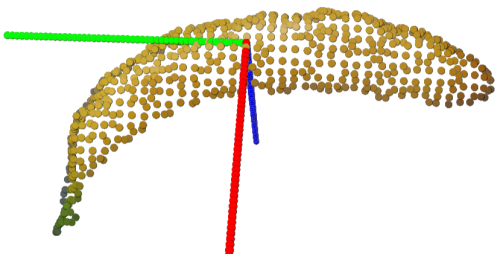


Figure 6.7: Point cloud with local frame - banana

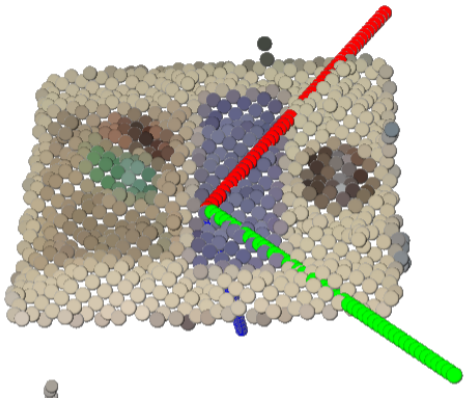


Figure 6.8: Point cloud with local frame - Nescaffe

The choice of the local x axis has been done according to the direction of the end effector x axis. In fact, since the control system tends to match the end effector frame with the local frame, a bad orientation choice would cause unwanted end effector rotations and possibly out of pose errors. To avoid this, the local x axis has been chosen as the end effector x axes projected onto the normal plane; the y axis has been derived to have a dextrose orthonormal frame. With such choice the local x axis is closer to the end effector one, avoiding unwanted rotations. Figs. 6.8 and 6.7 present some examples. In Fig. 6.8 the point cloud is good and the flatness of the surface is hence well captured, as predicted by the normal. In general the more precise the results should be the better the depth image should be, at the cost of employing a more expensive camera. However it will be proved in the following section, that with the current settings the results are anyway acceptable.

6.4 Total pipeline description

Algorithm 1 presents the general structure of the pipeline, including some further complexities to account for the real working setup. Some filters are in fact necessary:

- Line 10: the predictions are filtered with a selected threshold on the objectness score, if such threshold leads to an empty detection it must be reduced until some detection appear.
- Line 26: a grasp point too low probably means that a point on the bin's bottom has been picked. The code moves to the next object.
- Line 28: if the grasp pose is out of admissible range for the robot, the code moves to the next object.

Such filters prevent the stopping of the program, allowing to continue picking, eventually changing the scenario. Such changes may allow to empty the scene.

Algorithm 1: Algorithm for grasping pipeline

Data: acquisition position, acquisition attitude, place position, place attitude,threshold,flag SAM,T_{cam}^{shaft}

Initialize camera and robot objects;
Load networks;
while *True* **do**
    Move to acquisition pose;
    $T_{cam}^{base} \leftarrow T_{shaft}^{base} * T_{cam}^{shaft}$;
    Acquire images;
    Perform first prediction;
    if *prediction* $\neq$ *None* **then**
        filter boxes;
        while *filtered boxes is None* **do**
            Reduce threshold;
            if *threshold* $\leq$ *0.1* **then**
                System exit;
            end
            filter boxes;
        end
        if *SAM refinement* **then**
            Apply SAM on predicted boxes;
        end
        Calculate grasping points;
        Reorder grasping points according to selection criteria;
        index mask $\leftarrow$ 0;
        while *index mask* $\leq$ *number of masks-1* **do**
            Extract depth instance from current segmentation mask;
            Calculate T_{loc}^{base};
            if *grasp z too small* **then**
                index mask $\leftarrow$ index mask +1;
            else if *Pose out of range* **then**
                index mask $\leftarrow$ index mask +1;
            else
                Move robot to place pose;
            end
        end
    else
        System Exit;
    end
end

Future works may focus on the following optimizations:

- **Fixed camera:** The experiments where carried out with the camera mounted on the robot. If the camera is fixed, while the robot is moving the algorithm can acquire a new image and perform the prediction, thus saving time.
- **Suction feedback:** Noisy depth may lead to error on the grasping point z coordinate, thus affecting the grasping. This can be alleviated by providing a feedback to the suction when the contact happened.
- **Memory mechanism:** In case of unsuccessful picking with the object that remains in the same position, a mechanism that records and avoids these cases is necessary.
- **Obstacle avoidance:** If the gripper may collide with the bin a system to predict the collision is needed.

Chapter 7

Experimental tests

This chapter illustrates the metrics used to classify the scenes, and discusses the overall accuracy acquired. The chapter is organized as follows: at first a brief introduction on the metric is given for both the segmentation and grasping parts, then a set of examples is provided.
The division of the metrics will help in understanding whether the cause of unsuccessful picking is due to error in the segmentation part or in the picking phase.

7.1 Custom-metrics based analysis

7.1.1 Segmentation quality criteria

The segmentation quality criteria have been divided into two parts.

- Object detection metrics
- Instance segmentation metrics

With such division the differences, introduced especially via the use of SAM, can be better highlighted. Regarding the first category, it was decided to collect the errors according to the following metrics.

- **Wrong dimensions**: if some of the boxes, even the eventual additional ones, detected for a single object are smaller or bigger than the object dimension.
- **Missed object**.
- **Blobs detection**: when there are boxes that encapsulate more than 1 object.
- **Over detection**: when there are more than 1 boxes that encapsulate an object.

- **Parts detection**: detection of parts of objects (also smaller or bigger than the real object part).
- **Background's detection**.

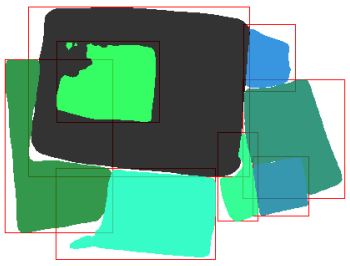
Criteria 1 (smaller boxes) and 5 (smaller part) are different: the former refers to when the boxes almost capture the entire object, while the latter when some boxes specifically detect parts of an object, thus the difference is in the size of the prediction. The same difference is used to distinguish criteria 1 (bigger boxes) and 4: the former is used when the boxes are slightly larger than the object, while the latter when some boxes are so large that encapsulate other instances. For what concerns the segmentation criteria:

- **Under-segmentation**: unique bounding box, but the segmentation is smaller than the ground truth.
- **Over-segmentation**: unique bounding box, but the segmentation is larger than the ground truth.
- **Blobs segmentation**.
- **Parts segmentation**: same concept described for detection anomaly.
- **Background segmentation**.

Criterion 2 refers to when the segmentation includes also parts of adjacent objects or part of the background. This is however different from criterion 4, which is considered in conjunction with criterion 5 for object detection. Fig 7.1 provides some examples to better understand the above discussions.



((a)) Vision errors RGB image



((b)) Vision errors segmentation mask

Figure 7.1: Example of vision errors

As it can be seen, two errors of detection appear: the white box is segmented in two parts and one of them is bigger than the part itself, secondly the box located

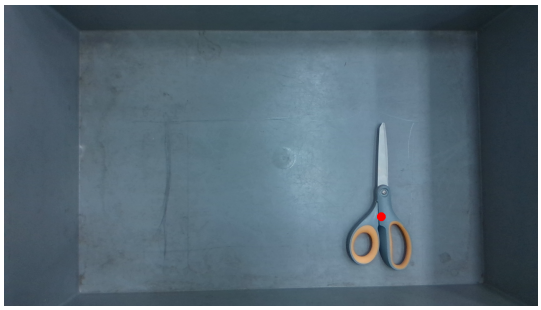
on top of it has a smaller detection. This is going to lead to two errors in the segmentation: a part's segmentation error and an under-segmentation one, which are related to the corresponding errors in the bounding boxes. Such correlation is not however necessary; in fact, for the two red boxes on top of each other, despite a correct detection there is an over-segmentation error. The examples provide useful insights in the criteria description and point out that the dependency of the detection and segmentation accuracy is not necessary.

7.1.2 Grasping quality criteria

The grasping criteria are more subjective to the user opinion and are strictly related to the quality of the segmentation. They are classified as follows.

- **Correct grasping point choice**.
- **Correct normal**.

The correct choice of grasping is trickier, since it strictly depends on the quality of the segmentation. For simplicity, this criterion has been considered only when the segmentation was good enough. The metric was subjective to the user, and it was based on the shape of the object. Future development may include the calculation of a score to assess the quality. Fig. 7.2 provides a couple of examples for incorrect choice of grasping point.



((a)) Picking error scissors



((b)) Picking error mug

Figure 7.2: Example of picking errors

In Fig. 7.2(a) the grasping point is too close to the hole, thus the suction effect does not work and the picking was in fact not successful. A point choice slightly on the right would have probably lead to a successful picking. In Fig. 7.2(b) the view of the mug leads to a point too close to the boundary of the surface; despite the completed picking, a different view could have caused the picking point to be outside the wanted flat region. The second criterion is not related to the

picking logic, but it depends on the used camera. Hence, the better the camera, the better the normal calculation. The criterion was however considered for the sake of completeness, since the used camera is not the optimal one. For instance, in the case of Fig. 7.3 the depth is absent close to the grasping point location, this distorts the normal giving a wrong picking pose.

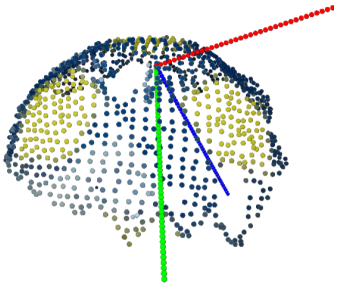


Figure 7.3: Point cloud with local frame - ball

7.2 Experiment's outcomes

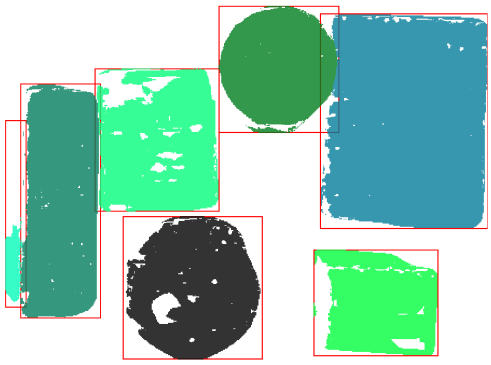
The experiment have been carried out by clustering the objects into 4 bins, depending on capability of the gripper to perform the picking. This division is thus strongly influenced by the physical properties of the objects. The four bins are easy, medium, hard and transparent/translucent. For each category two scenarios have been investigated: single objects scenes and cluttered object scenes. The former included objects separated from each other, it is used to test the capability of the network to detect the single object. The latter scenes were relative to the same objects but in a cluttered context, which is the typical working condition. For this type a sequence of picking actions is shown, to see how the segmentation improves as the scene becomes emptier. Each scene has been evaluated also with SAM; for a matter of repeatability, the SAM's scenes were made as close as possible to the Mask R-CNN ones, however small differences, that almost did not impact the segmentation, are present. After each complex scene, tables will be provided summarizing the observations only on Mask R-CNN model. Some of the presented categories were impossible to be picked due to shape, weight and the category itself. In these cases the objects were removed manually, hence the picking metrics are not recorded, this was done to anyway evaluate the improving of the segmentation accuracy as the scene was emptied.
For detection and segmentation results just the first 9 picking actions were reported, as it becomes generally easy to detect and segment when the scene starts to empty, whereas the picking outcome was entirely described, given that the related errors can occur anytime.

7.2.1 Category 1 : Easy

The category contains graspable objects with regular shape and large dimensions (e.g. boxes, large spheres).

Single objects scenario

The first scenario of the easy test case allows to see how goodness of the overall detection (Fig. 7.4). To work in this kind of conditions, Mask R-CNN should be considered the best trade-off to get the desired masks. All the objects have been detected and segmented.



((a)) Mask R-CNN - mask with depth

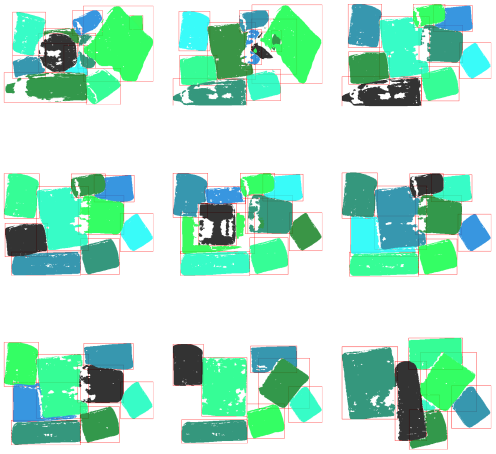


((b)) Mask R-CNN - grasp

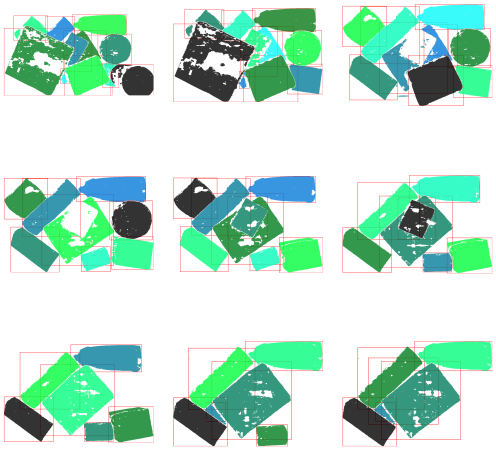
Figure 7.4: Easy picking - single

Cluttered objects scenario

In Fig. 7.5 the cluttered situation is displayed. Differently from the previous case, also the SAM outcome is analyzed in order to visualize small differences in the output masks.



((a)) Mask R-CNN - mask with depth



((b)) SAM - mask with depth



((c)) Mask R-CNN - grasp



((d)) SAM - grasp

Figure 7.5: Easy picking - cluttered

Analysis and comments

The easy category is not problematic to detect and pick since all the objects (not entirely shown) were successfully picked. From the detection and segmentation results, resumed in Tabs. 7.1 and 7.2, no critical aspects arose. There are some over detection and parts' detection and a background segmentation, probably caused by the light contrast present in the tote box.
Regarding the picking quality outcomes (Tab. 7.3), the grasping points were often valid, except for the fourth and seventh scenes where the inclination of the corresponding item was responsible for the wrong choices. These issues were solved by repositioning the considered objects in proper configurations, in order to complete effectively the picking phase.

DETECTION			
N°	*WRONG DIMENSIONS*	*MISSED OBJS*	*BLOBS OF OBJS*
1	0	1	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
N°	*OVER DETECTION*	*PARTS OBJ BOX*	*BACKGROUND BOXES*
1	0	2	1
2	2	0	0
3	0	0	0
4	0	0	0
5	0	2	1
6	0	0	1
7	0	0	1

Table 7.1: Quality table for Detection (Easy) task

SEGMENTATION		
N°	*UNDER/OVER SEGM*	*SEGM BLOBS OF OBJS*
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
N°	*PARTS SEGM*	*BACKGROUND SEGM*
1	2	1
2	2	0
3	0	1
4	0	0
5	2	1
6	0	1
7	0	1

Table 7.2: Quality table for Segmentation (Easy) task

PICKING			
N°	*WRONG CHOICES*	*CORRECT NORMAL*	*FAILURE CASE*
1	0	✓	None
2	1	✓	None
3	0	✓	None
4	1	✓	Grasping point
5	1	✓	None
6	0	✓	None
7	0	✗	Normal pose
8	0	✓	None
9	0	✓	None

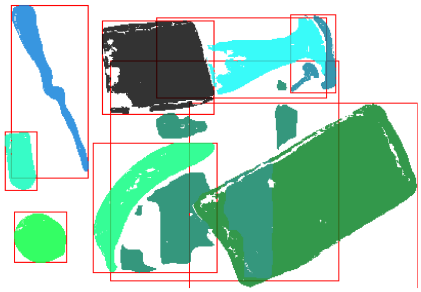
Table 7.3: Quality table for Picking (Easy) task

7.2.2 Category 2: Medium

This category contains object that are regular but small, thus they are not easily graspable. It also includes transparent objects with labels and small items inside plastic bags.

Single objects scenario

Also for this category only the Mask R-CNN outcome is depicted, due its quite reliable performance. As shown in Fig. 7.6, some issues occur in the predicted masks due to over detection: the detection provides more boxes than the ones required, so evident super-positions are present. For objects with a great color contrast (e.g., the plastic container vs its lid) over detection is a problem.



((a)) Mask R-CNN - mask with depth

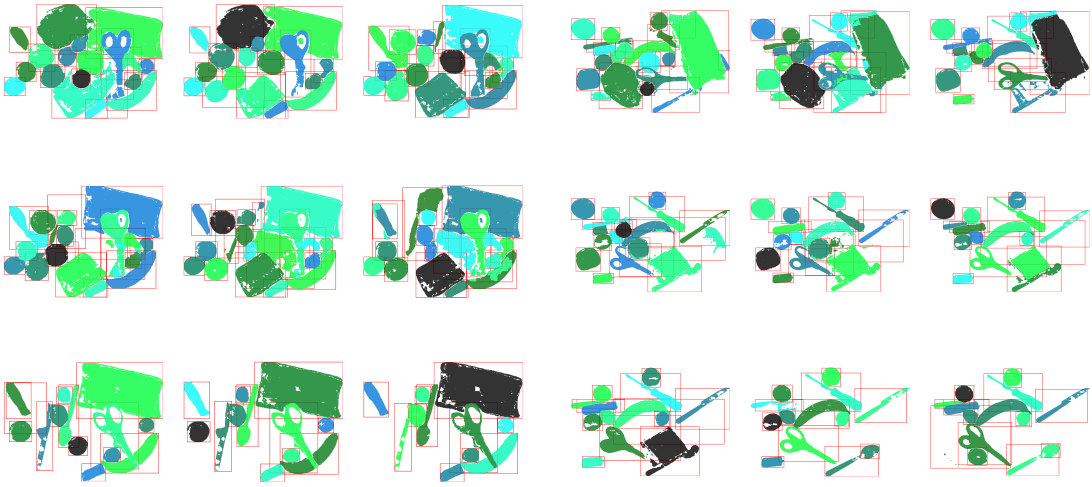


((b)) Mask R-CNN - grasp

Figure 7.6: Medium picking - single

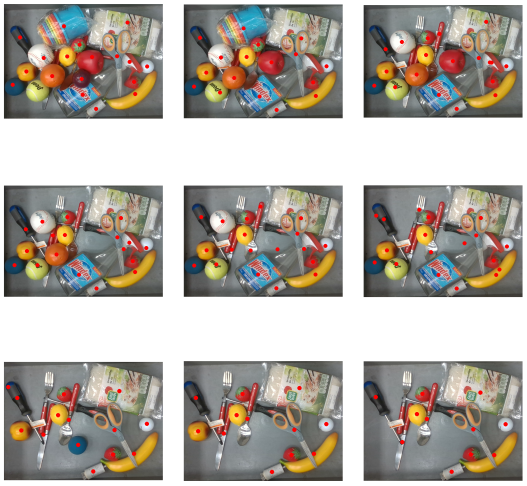
Cluttered objects scenario

As demonstrated in Fig. 7.7, the segmentation accuracy starts to decrease given the number of objects in the scenes, however, as the scene gets empty, the segmentation also improves.

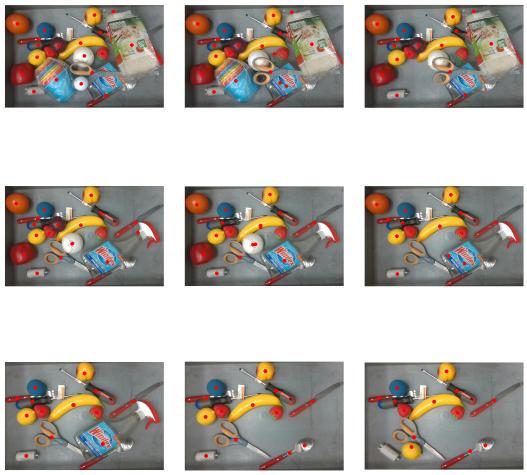


((a)) Mask R-CNN - mask with depth

((b)) SAM - mask with depth



((c)) Mask R-CNN - grasp



((d)) SAM - grasp

Figure 7.7: Medium picking - cluttered

Analysis and comments

This scenario is more complicated, with respect to the previous one, from the picking point of view. In fact, despite a good scene segmentation some of the objects had to be removed by the user. This was due to either a bad choice of the grasping point (scissors) or to the suction effect that did not work (plastic bags. tennis ball, baseball). In Tabs. 7.4 and 7.5 several inaccuracies are reported. In addition, Tab. 7.6 highlights the picking phase, showing no big criticality, except for some grasping points not properly chosen. But as previously said, such problems are out of the scope of the thesis.

DETECTION			
N°	*WRONG DIMENSIONS*	*MISSED OBJS*	*BLOBS OF OBJS*
1	1	1	1
2	1	1	0
3	0	2	2
4	0	3	0
5	0	2	3
6	2	2	0
7	0	2	0
N°	*OVER DETECTION*	*PARTS OBJ BOX*	*BACKGROUND BOXES*
1	0	2	0
2	0	1	0
3	1	2	0
4	0	3	0
5	2	2	1
6	2	3	1
7	1	1	0

Table 7.4: Quality table for Detection (Medium) task

SEGMENTATION		
N°	*UNDER/OVER SEGM*	*SEGM BLOBS OF OBJS*
1	1	1
2	1	0
3	0	2
4	1	0
5	2	3
6	2	0
7	2	0
N°	*PARTS SEGM*	*BACKGROUND SEGM*
1	2	0
2	1	0
3	2	0
4	3	0
5	2	1
6	3	1
7	1	0

Table 7.5: Quality table for Segmentation (Medium) task

PICKING			
N°	*WRONG CHOICES*	*CORRECT NORMAL*	*FAILURE CASE*
1	0	✓	None
2	1	✓	Incorrect grasping point
3	0	✓	None
4	0	✓	None
5	1	✓	Incorrect grasping point
6	0	✓	None
7	0	✓	None
8	0	✓	None
9	0	✓	None

Table 7.6: Quality table for Picking (Medium) task

7.2.3 Category 3: Hard

This categorization contains objects that are difficult to grasp depending on their orientation and shape.

Single objects scenario

From the simple scenario it can be noticed how, despite the reasonable detection, the final masks may not be the ideal ones as shown in Fig. 7.8. Adopting SAM in this case could turn out to be a good choice, increasing the quality of the segmentation masks. Furthermore, the lack of a total depth information could lead to slightly different grasping points. This can be a critical aspect if we consider the complex shape of these objects and the precision required in the grasping point choice.



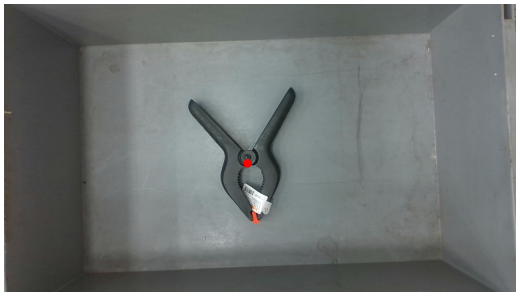
((a)) Mask R-CNN - mask with depth



((b)) Mask R-CNN - mask with depth



((c)) Mask R-CNN - grasp

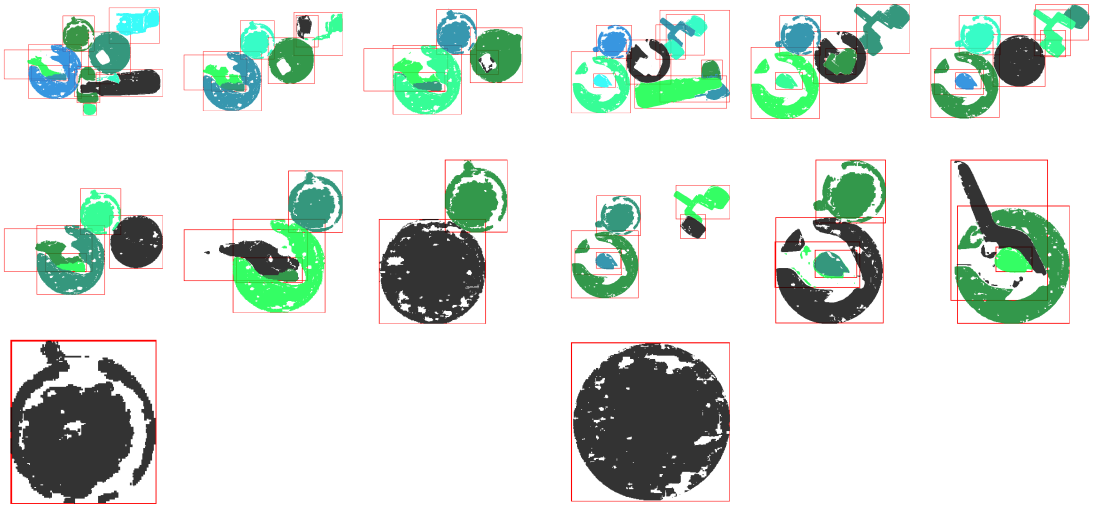


((d)) Mask R-CNN - grasp

Figure 7.8: Hard picking - single

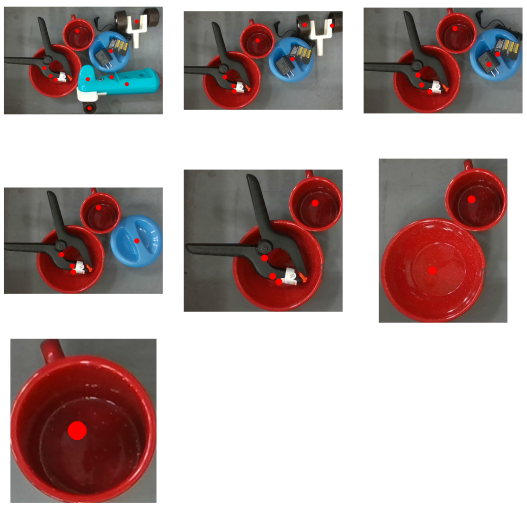
Cluttered objects scenario

In Fig. 7.9, the worst case scenario is presented: hard graspable objects in a complex setting.



((a)) Mask R-CNN - mask with depth

((b)) SAM - mask with depth



((c)) Mask R-CNN - grasp



((d)) SAM - grasp

Figure 7.9: Hard picking - cluttered

Analysis and comments

Even though this is the hardest grasping case, the final result show no failure cases. However, from Tabs. 7.7 and 7.8 several errors occur; in particular, given the complex shape and colors of some objects, often more than one bounding box is detected, leading to the prediction of many parts segmentation for the same item, an undesired behaviour. Moreover, in some situations some objects are not detected at all, probably for their consistency with respect to the overall scene. In Tab. 7.9 instead, not relevant errors happen: the multiple parts detections lead to multiple grasping choices, but at the end the robot manages to grasp each of the object present in the scene. Some normal vectors are not properly defined for some shadow issues, but still this does not affect the final outcome. However, the experiment can be considered as a lucky case.

DETECTION			
N°	*WRONG DIMENSIONS*	*MISSED OBJS*	*BLOBS OF OBJS*
1	0	2	0
2	0	1	0
3	1	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
N°	*OVER DETECTION*	*PARTS OBJ BOX*	*BACKGROUND BOXES*
1	0	6	1
2	0	4	0
3	0	2	0
4	0	1	0
5	0	2	0
6	0	0	0
7	0	0	0

Table 7.7: Quality table for Detection (Hard) task

SEGMENTATION		
N°	*UNDER/OVER SEGM*	*SEGM BLOBS OF OBJS*
1	2	0
2	2	0
3	2	0
4	1	0
5	1	0
6	0	0
7	0	0
N°	*PARTS SEGM*	*BACKGROUND SEGM*
1	6	1
2	4	0
3	2	0
4	2	0
5	0	0
6	0	0
7	0	0

Table 7.8: Quality table for Segmentation (Hard) task

PICKING			
N°	*WRONG CHOICES*	*CORRECT NORMAL*	*FAILURE CASE*
1	0	✓	None
2	1	✓	None
3	0	✗	None
4	0	✓	None
5	1	✓	None
6	0	✓	None
7	0	✗	None

Table 7.9: Quality table for Picking (Hard) task

7.2.4 Category 4: Transparent and Translucent

This categorization includes transparent and translucent items, and it aims at pointing out how a particular position or light condition can tremendously affect the final detection and segmentation. It is also shown how the missing depth or a noisy one prevent the final picking, despite a correct detection.

Single objects scenario

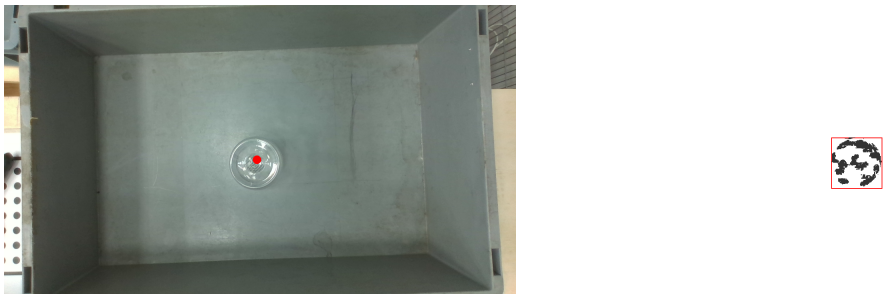
Fig. 7.10 shows how the detection of transparent objects is influenced by the light contrasts.



((a)) MaskRCNN - RGB **((b))** Mask R-CNN - mask

Figure 7.10: Mask R-CNN - Transparent single - scenario 1

The right side of Fig. 7.11 shows how the depth information of the reversed glass is not captured by the camera.



((a)) Mask R-CNN - grasp **((b))** Mask R-CNN - mask with depth

Figure 7.11: Mask R-CNN - Transparent single - scenario 2

Regarding translucent objects the situation is quite similar. In this case, the scene is strongly dependent on the light conditions. Fig. 7.12 shows how translucent objects are detected and the segmentation mask is accurate. However, if the

object's surface is highly reflective big issues can occur in the depth acquisition, as represented in Fig. 7.13.

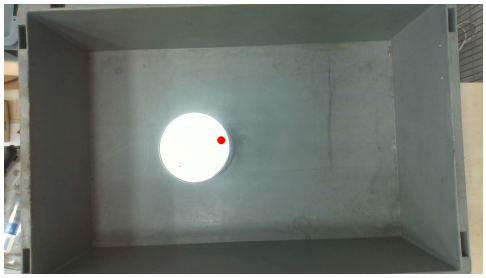


((a)) MaskRCNN - RGB



((b)) Mask R-CNN - mask

Figure 7.12: Mask R-CNN - Translucent single - scenario 1



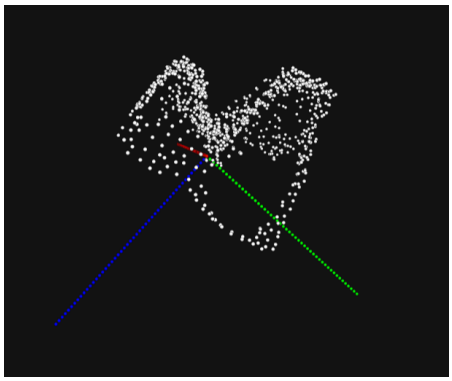
((a)) Mask R-CNN - grasp



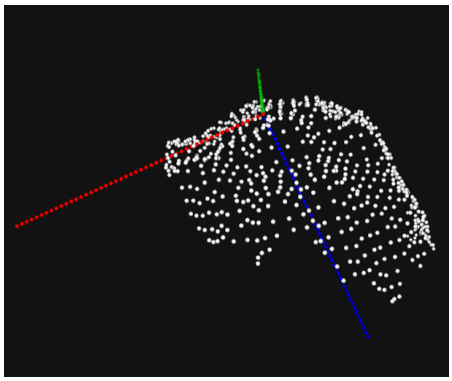
((b)) Mask R-CNN - mask with depth

Figure 7.13: Mask R-CNN - Translucent single - scenario 2

As it can be seen, the depth information is either missing or noisy in both the categories. These factors degrade the final normal vector, as depicted in Fig. 7.14.



((a)) Horizontal glass point cloud



((b)) Translucent ball point cloud

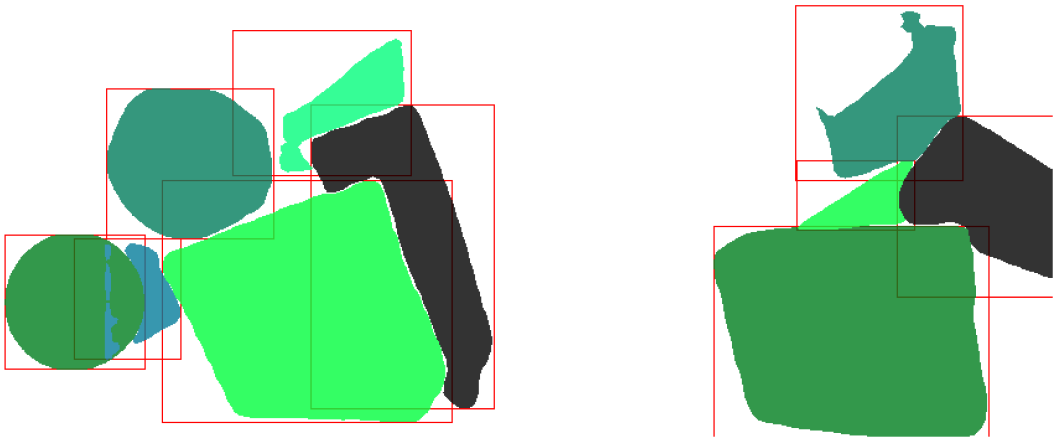
Figure 7.14: Normal vectors for transparent and translucent objects

7.2.5 Experimental comparison Mask R-CNN and SAM

The experiments of Section 7.2 have been presented for both Mask R-CNN and SAM. However, to show that SAM predictions are better than Mask R-CNN ones, the scenarios should be identical for the two models (same detections), which is practically unfeasible. Thus, SAM was run on the same images of the Mask R-CNN's scenario, to see if a better segmentation would have lead to different and perhaps better pickings.



((a)) Mask R-CNN first picked objects

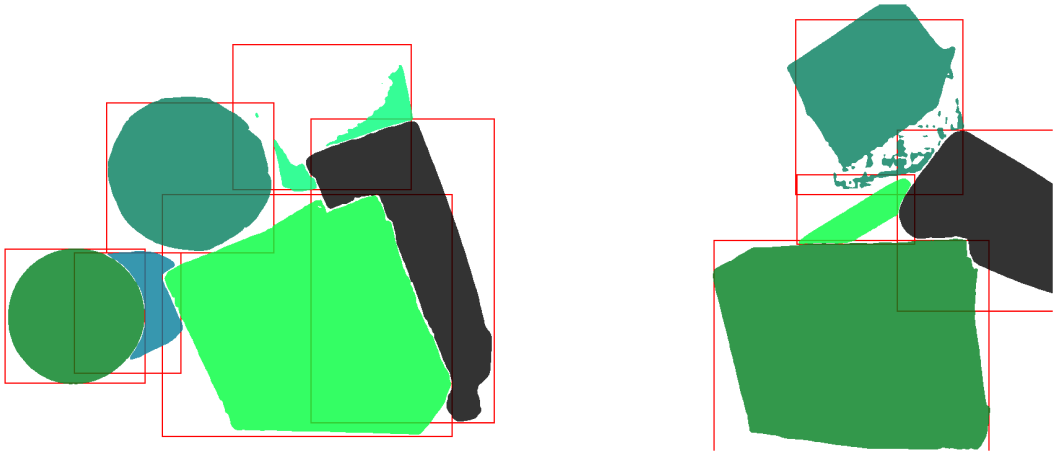


((b)) Mask R-CNN segmentation mask

Figure 7.15: Mask R-CNN examples



((a)) SAM first picked object



((b)) SAM segmentation mask

Figure 7.16: SAM examples

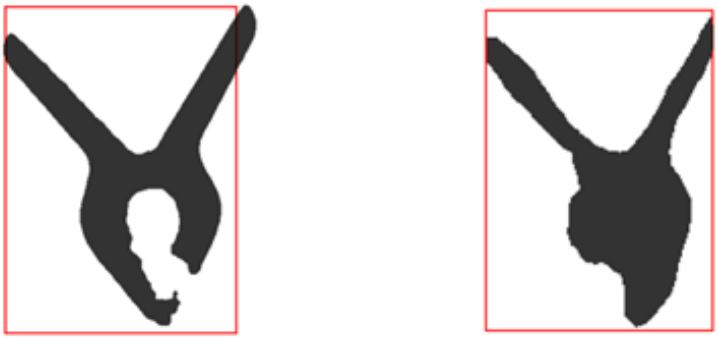


Figure 7.17: Segmentation mask - Mask R-CNN (right) vs SAM (left)

Figs. 7.15 and 7.16 plot the sequence of the first picked objects (green areas) and the segmentation masks for both Mask R-CNN and SAM, applied to the same scenario. As it can be seen, SAM mainly segment only one of the two objects, while Mask R-CNN segments parts of both. This has lead to an invalid grasping point when Mask R-CNN was used. SAM generally provides better segmentation boundaries, which is particularly important for complex objects, such as the one of Fig. 7.17. This is true providing that the detection is accurate. However, the usage of SAM comes at the price of a much higher computational time. Tabs. 7.10 and 7.11 display the computational times of the two models on CPU and GPU respectively. When running on CPU, SAM is too heavy to be used for industrial applications. In these cases a performing GPU is required. Tab. 7.12 provides the computational time for the picking pre-processing phase.

times	**mean (s)**	**std (s)**	**max (s)**	**min (s)**
Segmentation time Mask R-CNN	5.04	0.55	7.05	4.20
Segmentation time SAM	42.8	2.12	50.1	38.2

Table 7.10: Segmentation times on Intel Core i7-1165G7

times	**mean (s)**	**std (s)**	**max (s)**	**min (s)**
Segmentation time Mask R-CNN	0.0256	0.0293	0.0488	0.0188
Segmentation time SAM	0.0657	0.0089	68.1	0.0638

Table 7.11: Segmentation times on NVIDIA RTX A5000

times	**mean (s)**	**std (s)**	**max (s)**	**min (s)**
Grasping point determination	0.284	0.168	0.721	0.0196
Normal calculation	0.150	0.106	0.343	0.0373

Table 7.12: Picking criteria computational times

Chapter 8

Conclusions and future works

Computer vision is a fundamental working sector for industrial automation and the combination with the fast-growing AI field allowed to create a valid and efficient way of tackling a lot of practical demands. In our thesis it was showed how the use of modern DL networks gives the possibility to create a basic pipeline that has, as final aim, the capability of visually separating the objects, contained in a specific scene, in order to predict in which area an industrial manipulator has to steer for the successive picking movement.
In particular, a quite common supervised learning strategy like the binary instance segmentation, supported by a strong dataset to which work on, ends in a satisfactory outcome, as shown in the final experimental tests.
Obviously, in order to realize such a work, a deep study in this area has been made, analyzing both the benchmark models and the state-of-the-art architectures. This mix allowed to obtain two valid ways of reaching the same objective, but an evident difference in terms of available hardware has to be taken into consideration.
A more extensive research and development phase can be carried on in order to improve the process and make it as efficient as possible.
Given some physical and time limits, we tried to gain the best working condition for what DL approach concerns. But certainly having the possibility to train our models for a longer period of time and changing accordingly the training hyper-parameters (like the step size of the learning rate decay for example), could have lead to a small but quite significant enhancement in terms of accuracy.
By taking in consideration also the flexibility and the re-usability of our model, searching for new structured datasets can help out in this sense. This is a shared procedure in the DL community that faces supervised tasks and we confirmed it by just mixing two valid datasets for grasping activity as discussed in Section 5.3.6.

Future works could better explore both the object detector and instance segmentation state-of-the-art networks, or directly get involved in a custom design of a DL model performing these kinds of tasks. Obviously, the second option is more time consuming, because it requires an entire period of research, but just by adopting new detector models like YOLOv7 [45], or by taking inspiration from SAM model can be sufficient to increase the final accuracy by some percentage points.
Even though the scope of this thesis has been focused on the development of a segmentation-based approach, it is worthy to notice that also the grasping choice is a critical aspect to be considered. As shown in Section 6.2, the final point to which the robotic arm has to approach is not always the optimal one. Hence, instead of applying a structured but not so flexible technique based on centroid-like points, another DL model could be investigated. It would be in charge of predicting a possible grasping point based on a solid dataset that comes with the corresponding labels. Thus to follow this path a grasping-like dataset (something similar to GraspNet) and a robust and proper neural network should be investigated to accomplish such a specific task.

Bibliography

[1] Hanbo Zhang, Jian Tang, Shiguang Sun, and Xuguang Lan. *Robotic Grasping from Classical to Modern: A Survey*. 2022. arXiv: 2202.03631 [cs.RO] (cit. on p. 1).

[2] Sung Eun Kim and Il Won Seo. «Artificial Neural Network ensemble modelling with conjunctive data clustering for water quality prediction in rivers». In: *Journal of Hydro-environment Research* 9 (Apr. 2015). DOI: 10.1016/j.jher.2014.09.006 (cit. on p. 4).

[3] *River Trail*. URL: http://intellabs.github.io/RiverTrail/tutorial/ (cit. on p. 5).

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV] (cit. on p. 6).

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV] (cit. on pp. 6, 13, 14).

[6] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. «Sequence to sequence learning with neural networks». In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 6).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV] (cit. on pp. 7, 15, 17, 18).

[8] Fisher Yu and Vladlen Koltun. «Multi-scale context aggregation by dilated convolutions». In: *arXiv preprint arXiv:1511.07122* (2015) (cit. on p. 7).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition». In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. DOI: 10.1007/978-3-319-10578-9_23. URL: https://doi.org/10.1007%2F978-3-319-10578-9_23 (cit. on p. 7).

[10] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV] (cit. on p. 8).

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. «Rich feature hierarchies for accurate object detection and semantic segmentation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587 (cit. on p. 9).

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV] (cit. on pp. 9, 18, 20).

[13] A. Neubeck and L. Van Gool. «Efficient Non-Maximum Suppression». In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 3. 2006, pp. 850–855. DOI: 10.1109/ICPR.2006.479 (cit. on p. 9).

[14] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG] (cit. on p. 11).

[15] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. «A brief introduction to OpenCV». In: *2012 Proceedings of the 35th International Convention MIPRO*. 2012, pp. 1725–1730 (cit. on p. 11).

[16] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. «Albumentations: Fast and Flexible Image Augmentations». In: *Information* 11.2 (Feb. 2020), p. 125. DOI: 10.3390/info11020125. URL: https://doi.org/10.3390%2Finfo11020125 (cit. on p. 11).

[17] Fabian Pedregosa et al. *Scikit-learn: Machine Learning in Python*. 2018. arXiv: 1201.0490 [cs.LG] (cit. on p. 11).

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 15).

[19] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. «Pyramid Scene Parsing Network». In: *CVPR*. 2017 (cit. on p. 16).

[20] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV] (cit. on pp. 17, 23, 24).

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. «SSD: Single Shot MultiBox Detector». In: *ECCV*. 2016 (cit. on p. 17).

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV] (cit. on p. 18).

[23] Ross Girshick, Georgia Gkioxari, Piotr Doll, and Kaiming He. *MaskRCNN*. 2018. arXiv: 1703.06870 [cs.CV] (cit. on p. 20).

[24] Seunghyeok Back, Jongwon Kim, Raeyoung Kang, Seungjun Choi, and Kyoobin Lee. «Segmenting unseen industrial components in a heavy clutter using rgb-d fusion and synthetic data». In: *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020, pp. 828–832 (cit. on pp. 21, 22).

[25] Joosoon Lee, Seunghyeok Back, Taewon Kim, Sungho Shin, Sangjun Noh, Raeyoung Kang, Jongwon Kim, and Kyoobin Lee. «Fusing RGB and depth with Self-attention for Unseen Object Segmentation». In: *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. 2021, pp. 1599–1605. DOI: 10.23919/ICCAS52745.2021.9649991 (cit. on p. 22).

[26] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. «Masked Autoencoders Are Scalable Vision Learners». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 16000–16009 (cit. on p. 24).

[27] Bowen Cheng, Alex Schwing, and Alexander Kirillov. «Per-Pixel Classification is Not All You Need for Semantic Segmentation». In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 17864–17875. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/950a4152c2b4aa3ad78bdd6b366cc179-Paper.pdf (cit. on p. 24).

[28] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. «GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11444–11453 (cit. on p. 26).

[29] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. «Yale-CMU-Berkeley dataset for robotic manipulation research». In: *The International Journal of Robotics Research* 36.3 (2017), pp. 261–268 (cit. on p. 26).

[30] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. «Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics». In: *arXiv preprint arXiv:1703.09312* (2017) (cit. on p. 26).

[31] Chaitanya Mitash, Fan Wang, Shiyang Lu, Vikedo Terhuja, Tyler Garaas, Felipe Polido, and Manikantan Nambi. «ARMBench: An object-centric benchmark dataset for robotic manipulation». In: *arXiv preprint arXiv:2303.16382* (2023) (cit. on p. 28).

[32] *ARMBENCH DATASET*. 2023. URL: http://armbench.s3-website-us-east-1.amazonaws.com/segmentation.html (cit. on p. 29).

[33] Mouna Afif, Riadh Ayachi, and Mohamed Atri. «Indoor objects detection system implementation using multi-graphic processing units». In: *Cluster Computing* 25 (2022), pp. 469–483. DOI: 10.1007/s10586-021-03419-9 (cit. on p. 31).

[34] *Intel Realsense Depth Camera D435i*. URL: https://shop.line.me/@oculusthailand/product/1000180676 (cit. on p. 32).

[35] Hongjun Wang, Yiyan Lin, Xiujin Xu, Zhaoyi Chen, Zihao Wu, and Yunchao Tang. «A Study on Long–Close Distance Coordination Control Strategy for Litchi Picking». In: *Agronomy* 12 (June 2022), p. 1520. DOI: 10.3390/agronomy12071520 (cit. on p. 32).

[36] *Robot Antropomorfo Comau Racer 3-0.63*. URL: https://etneo.com/prodotto/robot-antropomorfo-comau-racer-3-0-63/ (cit. on p. 32).

[37] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. «Learning Data Augmentation Strategies for Object Detection». In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Cham: Springer International Publishing, 2020, pp. 566–583. ISBN: 978-3-030-58583-9 (cit. on p. 40).

[38] Panagiotis Meletis and Gijs Dubbelman. «Training of Convolutional Networks on Multiple Heterogeneous Datasets for Street Scene Semantic Segmentation». In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1045–1050. DOI: 10.1109/IVS.2018.8500398 (cit. on p. 53).

[39] *Comau Racer 3 Specifications*. URL: https://robodk.com/robot/Comau/Racer-3 (cit. on p. 58).

[40] *OpenCV; Camera Calibration and 3D Reconstruction*. URL: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga687a1ab946686f0d85ae0363b5af1d7b (cit. on p. 59).

[41] *OpenCV: Camera Calibration*. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (cit. on p. 60).

[42] Antonio Alliegro, Martin Rudorfer, Fabio Frattin, Aleš Leonardis, and Tatiana Tommasi. *End-to-End Learning to Grasp via Sampling from Object Point Clouds*. 2022. arXiv: 2203.05585 [cs.RO] (cit. on p. 60).

[43] Waleed Abu-Ain, Siti Norul Huda Sheikh Abdullah, Bilal Bataineh, Tarik Abu-Ain, and Khairuddin Omar. «Skeletonization Algorithm for Binary Images». In: *Procedia Technology* 11 (2013). 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013, pp. 704–709. ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2013.12.248. URL: https://www.sciencedirect.com/science/article/pii/S2212017313004027 (cit. on p. 61).

[44] Jonathon Shlens. *A Tutorial on Principal Component Analysis*. 2014. arXiv: 1404.1100 [cs.LG] (cit. on p. 63).

[45] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: 2207.02696 [cs.CV] (cit. on p. 90).