Visual Affordance Prediction of Hand-Occluded Objects

Università di Genova

Queen Mary University of London

Tommaso Apicella

Supervisors: Prof. Paolo Gastaldo
Prof. Riccardo Berta
Prof. Andrea Cavallaro

Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN)
University of Genoa

Centre for Intelligent Sensing (CIS)
Queen Mary University of London

This dissertation is submitted for the degree of
Doctor of Philosophy

Joint Doctorate in Interactive and Cognitive Environments - Cycle 36

March 2024

Visual Affordance Prediction of Hand-Occluded Objects

Tommaso Apicella

Joint Doctorate in Interactive and Cognitive Environments
JD-ICE

XXXVI cycle

Acknowledgements

This PhD Thesis has been developed in the framework of, and according to, the rules of the Joint Doctorate in Interactive and Cognitive Environments JD-ICE with the cooperation of the following Universities:

Università degli Studi di Genova (UNIGE)

DITEN - Dept. of Electrical, Electronic, Telecommunications Engineering and Naval Architecture

SEALab - Smart Embedded Applications Laboratory

Elios Lab - Electronics for the Information Society Laboratory

Primary Supervisor: Prof. Paolo Gastaldo
Secondary Supervisor: Prof. Riccardo Berta

Queen Mary University of London (QMUL)

EECS - School of Electronic Engineering and Computer Science

CIS - Centre for Intelligent Sensing

Primary Supervisor: Prof. Andrea Cavallaro

Acknowledgements

I would like to thank my supervisors Prof. Paolo Gastaldo, Prof. Riccardo Berta, and Prof. Andrea Cavallaro, that supported me and gave me the unique opportunity of the joint supervision. During these years they guided me pushing me outside my comfort zone, making me see things from different perspectives, reminding me the objective of having an impact on the community. I want to thank also the collaborators, Giulia S., Alessio, and Edoardo, I think I learned a lot from their experience and from the discussions about research.

My year in London was one of the most formative experiences I ever had. It was a pleasure to spend time in CS440 and outside with Alessio, Changjae, Faxian, Long, Yik, Chaoran, Dmitrii, Vandana, Alina, Helen, Xavier. I am really grateful for the moments spent together, also with Simone, Giulia, Dayana, Nino, Richard, Alfie. All these people accepted me and helped me enjoying my stay in London.

A special thanks to long-time friends Giacomo R., Luca T., Pietro, Sara, Vincenzo, Davide, Monica, Nicola, Luisa, Auri, Daniele, Valerio, Barbison, Marco, Fra, Loci, Edo, Bob, Carola, Chiara, for the joyful moments and to the even longer-time friends Emanuele, Luca B., and Giacomo I., because they make my life beautiful.

Last but not least, I would like to thank my family, my mom and dad, my brother, my cousins Gianmarco, Veronica, Alessandro, Furio, that always believed in me and supported me.

Abstract

The prediction of affordances i.e., the potential actions an agent can perform on objects in the scene, is fundamental for human-robot collaboration and wearable robotics scenarios in which objects may be on a tabletop or held by a person. Perceiving affordances from an image is challenging due to the variety of object geometric and physical properties, as well as occlusions caused by clutter or by a person's hand holding the object. In this thesis, we propose a framework for visual affordance prediction that estimates object properties such as position and mass, and identifies graspable regions of objects, supporting the agent to perform the intended actions. As previous methods focused on predicting the filling mass of a container manipulated by a human, the complementary estimation of container mass regardless of the content was underexplored. Moreover, during a human manipulation more than one object could be in the scene, so a selection phase is necessary to focus only on the object of interest. We propose a strategy to select the object manipulated by a human from a fixed frontal RGB-D camera and we design a model to predict its mass. The model learns how to combine color and geometric information to predict the (empty) container mass. The integration of our pipeline with already existing filling mass predictors allows to obtain the complete container mass (object plus content). Object detection methods identify objects in a scene, however in wearable robotic applications the human knows objects location and category. We investigate a transfer learning procedure to locate objects in the scene regardless of their category ('objectness'). We target lightweight object detection models that could be used in a wearable application, where the trade-off between accuracy and computational cost is relevant and was previously not investigated. In case of human manipulations, the identification of the object regions an agent can interact with is more challenging due to occlusions and the poses object may take. We design an affordance segmentation model that learns affordance features under hand-occlusion by weighting the feature map through arm and object segmentation. Due to a lack of datasets to tackle this scenario, we complement an existing dataset, annotating the visual affordances of mixed-reality images of hand-held containers in third-person view. Experiments show that the strategy to select objects and predict their mass outperforms most baselines on previously unseen manipulated

containers; the transfer learning procedure improves the performance of lightweight object detection methods in a wearable application; and the affordance segmentation model achieves better affordance segmentation and generalisation than existing models.

Table of contents

Published work

Conference papers

[C1] T. Apicella, A. Cavallaro, R. Berta, P. Gastaldo, F. Bellotti and E. Ragusa. An Affordance Detection Pipeline for Resource-Constrained Devices. *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2021.

[C2] T. Apicella, G. Slavic, E. Ragusa, P. Gastaldo and L. Marcenaro. Container Localisation and Mass Estimation with an RGB-D Camera. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[C3] T. Apicella, A. Xompero, E. Ragusa, R. Berta, A. Cavallaro, and P. Gastaldo. Affordance segmentation of hand-occluded containers from exocentric images. *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2023.

List of figures

List of tables

Nomenclature

List of abbreviations

IoU	Intersection over Union
J	Jaccard index
mAP	Mean average precision
P	Precision
R	Recall

List of symbols

α	Coefficient in the focal loss
δ	Coefficient in the Dice loss
ε	Relative absolute error
η	Constant in the L1 loss
γ	Exponent in the focal loss
$\hat{m}$	Predicted mass
$\hat{S}$	Predicted segmentation map
λ	Coefficient of a loss
ϕ	Feature maps
ψ	Bounding box
$\tilde{m}$	Predicted binary segmentation map
A	Dependency between foreground and background pixels
a	Width aspect ratio between object crop and original image
B	Number of images in a batch

b	Height aspect ratio between object crop and original image
C	Segmentation classes
c	Index of a class
D	Varying importance of pixels
d	Average distance
E	Error between ground truth and prediction
G	Flattened annotation
I	RGB image
i	Index of images in a batch
I'	Depth image
j	Index of the recording in a dataset
K	Number of candidate patches
l	Index of pixel position
m	Annotated mass
N	Number of images in a dataset
n	Index of image in a dataset
O	Number of objects in an image
o	Index of object in an image
p	Probability of a class
q	Recall Index
S	Annotated segmentation map
s	Mass estimation score
t	Time instant
U	Number of tasks
u	Task of the agent
V	Number of recordings in a dataset
x	Pixel in an image
z	Euclidean distance

Chapter 1

Introduction

1.1 Motivation

Affordances are the potential actions that objects of interest present in the environment offer to an agent (e.g., a person or a robot) [31]. The perception of objects affordances from visual data enables assistive technologies for robotics and prosthetic applications (e.g., grasping, object manipulation [7, 78]), or collaborative human-robot scenarios (e.g., handovers [14, 101, 105, 128]), shown in Fig. 1.1. In these scenarios the affordance prediction allows the robotic hand to interact with objects [81, 24, 14, 129].

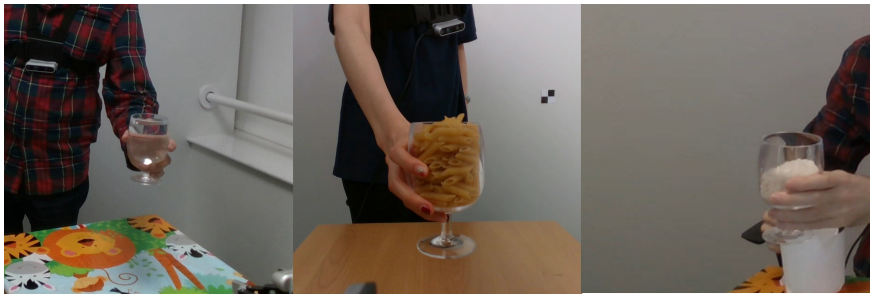
Affordances depend on several entities related to the object and the agent such as the task (or purpose) of the agent, the geometric, kinematic, and physical properties of the end effector (the part of the agent that interacts with objects). The geometric and physical properties of objects affect how the interaction between end effector and object will be carried out. The perception of affordances from an image is a complex problem due also to the changing appearance of objects and environment settings e.g., occlusions, lighting conditions. The variability in object shape and in setting pushed toward the adoption of machine learning techniques to exploit the learning and generalisation capabilities of data-driven methods. In particular, we refer to the supervised learning approach that is one of the most common in the field, but requires data to be annotated. Annotating affordances is a time consuming and expensive activity and it requires the definition of conventions due to the ambiguities.

(a) (b) (c)

Fig. 1.1 Examples of applications benefiting from visual affordance prediction: (a) object picking [78], (b) human-robot collaboration [105], and (c) wearable robotics [7].

In principle, an agent could perform different actions on the same object region depending on the broader purpose or context e.g., for humans the blade of a knife affords both grasping and cutting based on if they want to use the knife or pass it to another person. Another example is the internal surface of a cup that affords both grasping and containing based on if humans want to pour a liquid or move the cup, and if the cup is filled with content. The annotation can be manually performed on images [80, 82] or can be generated using computer graphics techniques starting from the annotation of a 3D model of the object (CAD) [23]. Moreover, generating a good quality CAD model from a real-life object is a time-consuming and expensive activity and leads to obtain a single instance of an object category. The projection of the object model in a scene assumes perfect knowledge about pose (translation and rotation) of the object in the environment, which is not something necessarily present at testing time, and requires additional processing to handle occlusions. The instances of the same category can vary in shape, hence training a model only on few instances may not be enough to generalise to unseen instances.

To perform actions on objects, an agent needs to estimate their properties e.g. location and mass, and identify the functional regions (affordances) such as the graspable ones [24, 105]. The properties and the functional regions allow the interaction between the agent and the object, they constitute an information that can be refined over time and also during the interaction. Functional regions are related to the object geometry and physical properties of the object e.g., concave shapes afford the holding of a content, sharp regions afford cutting. Estimating the object location, mass,



(a)

(b)

(c)

Fig. 1.2 Challenges in visual affordance prediction: (a) same object instance, different filling and background, (b) same object category, different instances, poses and backgrounds, (c) different object categories in different poses and scenes. Images are from datasets of the literature [40, 42, 107, 125], cropped for visualisation purpose.

and functional regions, from an image is not trivial due to the variations in object appearance that depend on the setting and the application. The object instance could be previously unseen [101, 128], occluded either by other objects in cluttered scenes or by a human hand during a manipulation. Containers are one of the most interesting object set because they can change their properties e.g., the mass due a manipulation (emptying or filling), their appearance in case of the material due to transparencies and/or stiffness [105] (see Fig. 1.2a and Fig. 1.2b).

The mass estimation is closely related to affordance, influencing the interaction between an agent and the objects in both the motion planning and the force regulation, and involving different modalities like vision, audio, tactile. The need of multiple modalities arises to cope with the challenges introduced by using only one input e.g., predicting the mass of container made of an opaque material and filled with unknown material, from an image. The additional audio and the tactile modalities can provide cues about the container material, and the filling material, that may be hard to predict from the visual information. In general, the mass of a container can be formulated as the sum between the mass of the container without considering the content (empty container mass) and the mass of the filling content (filling mass) [123]. Most of the available methods focused on filling mass estimation [15, 54, 67], so the estimation of the container mass regardless of the content is not fully investigated yet. Approaching this subproblem using only the visual modality is challenging because the appearance of objects is influenced by the type of interaction and by the object geometric and physical properties. The object could be occluded by a human hand manipulating it or the colour could be changed by the filling e.g. in case of transparent containers (see Fig. 1.2a). Another challenge consists of locating the object of interest, that may vary based on the setting or the application. Most of the methods that estimate the filling mass uses off-the-shelf detection methods that may lead to mistakes in the mass estimation if more than one object is present in the scene.

Identifying functional regions of objects (affordance segmentation) allows an agent to focus on graspable regions for a potential interaction. Methods in the literature adapt existing models for semantic or instance segmentation to locate and segment the affordances of objects placed on a tabletop [14, 24, 38, 80–82, 97, 129, 130]. The difference between affordance segmentation and the standard semantic and instance segmentation task is that in affordance segmentation most of the pixels of the image belong to the background and not to objects, hence learning the features to segment objects (as in semantic segmentation) is not sufficient, because functional regions are parts of the object mask. This makes the affordance segmentation problem highly imbalanced and challenging. The trade-off between computational cost and accuracy

is rarely investigated, but could be a requirement in the case of wearable robotic applications [90]. The segmentation of functional regions is even more challenging when the object is hand-held by a person due to the occlusions caused by the hand and the different poses that the object may take (see Fig. 1.2b and Fig. 1.2c). Only one work addressed this scenario, but the model did not explicitly consider the presence of the forearm and the hand [53]. This can result in inaccurate affordance segmentation. Additionally, the focus of the method on egocentric images from human perspective could be unsuitable for an assistive application, e.g. human-robot collaboration, or generalise to third person view. Another challenge of this scenario is the lack of datasets to train and evaluate models.

In this thesis, we tackle the challenging problem of affordance prediction, introducing an overall framework (see Fig. 1.3) that considers the connections between the affordances and the object properties. The modular structure allows the replacement of each component based on the needs e.g., lightweight models to increase the throughput, or more accurate models. Vision models are first trained offline on the single tasks to learn the parameters that are kept constant during the inference phase. Object detection methods restrict the input of subsequent models to some regions of interest in the image, ignoring the ones that are not informative. The selection of the object of interest in the scene depends on the specific purpose of the interaction e.g., taking the object that is held by a human. Based on the objects present in the scene, other models predict the mass, and the graspable regions to perform the task. The predictions of the visual system can be fused and used as prior information to support the movement planning, and the adjustment of the robotic hand pose and force during the interaction (*Control*). The fusion of the estimated mass and segmented affordances is strictly related to the control phase and the experimental validation consists in analysing the performance with a robotic hand, hence we keep it as part of the future work. In this thesis, we focus on the computer vision aspect of the framework, in particular the detection, mass estimation, and affordance segmentation subproblems from visual data. We tackle the mass estimation and the segmentation separately due to the absence of a single unifying dataset with the necessary annotation to train and test models. However, as

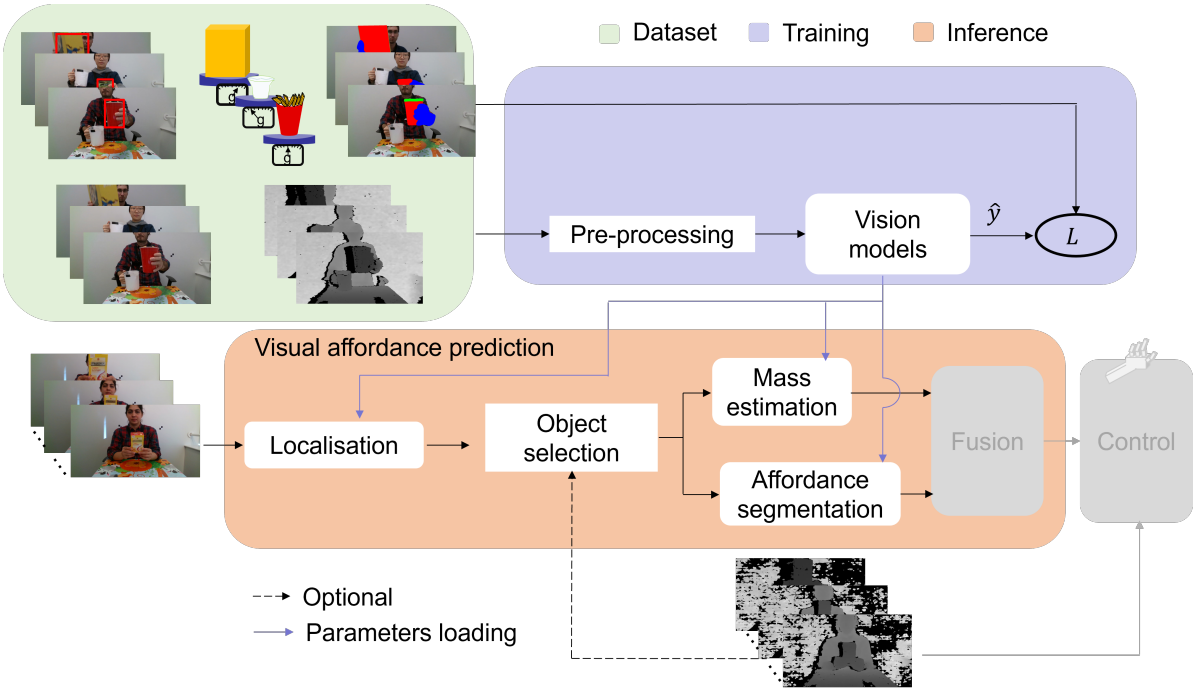


Fig. 1.3 Visual affordance prediction framework. The vision system predicts the location, the mass and the functional regions of objects in the field of view; based on the predicted information, the control system guides the robotic hand to interact with objects.

mentioned before, these subproblems are strictly related to the way the agent will interact with objects, from the planning of the object reaching to the actual interaction.

1.2 Problem formulation

A camera captures RGB-D frames with a human interacting with an object. Our objective is to the detect objects of interest in a frame, estimate their mass, and identify the affordance class *graspable* on the objects surface, separating it from *non-graspable* classes e.g., *contain* (internal surface of a cup), *pound* (top of a hammer), and *arm* (in case the object is hand-occluded).

Let $I_t \in \{0,...,255\}^{W \times H \times 3}$ be the RGB image of one or more objects $o = \{1,...,O\}$ an agent can interact with, where W is the width, H is the height, t is the time instant. The object(s) may be placed on a tabletop or held by one person. Let $I'_t \in \mathbb{R}^{W \times H}$ be the depth map associated to the RGB image, and $u \in \mathbb{N}$ be one of the U tasks of the agent.

We define a function $f_1(.)$ that predicts the location of objects encoded as bounding boxes $\psi_{t,o} \in \mathbb{R}^4$ where the dimensions are the top left corner coordinates in the image plane, width and height. A function $f_2(.)$ predicts the object mass $m_{t,o} \in \mathbb{R}_{>0}$. A function $f_3(.)$ predicts the segmentation mask $S_{t,o} \in \{0,...,C\}^{W_o \times H_o}$, where W_o and H_o are the object width and height in the image plane, each pixel is one of the C classes e.g., *background*, the object affordances *graspable* and *contain*, and the *arm* in case the object is hand-occluded.

Considering our framework, the function f_1, that localises and selects the objects, is fed with the images while the mass estimation and the graspable regions segmentation are performed by f_2 and f_3 respectively, based on f_1 output:

$$\psi_{t,o} = f_1(I_t, I'_t, u) \ , \tag{1.1}$$

$$m_{t,o} = f_2(I_t, I'_t, \psi_{t,o}) \ , \tag{1.2}$$

$$S_{t,o} = f_3(I_t, \psi_{t,o}, u) \ . \tag{1.3}$$

The task u of the agent is considered as a prior and defined based on the potential use of objects e.g., using a hammer to pound a nail, or taking a filled cup from a person's hand. The depth information I'_t is used to select candidate objects and to perform the mass estimation. I'_t is not considered in the affordance segmentation since most of the literature works show that the additional depth information does not improve performance [8, 17, 62, 80]. The overall function describing the visual affordance prediction can be written as combination of f_2, f_3 as $f(f2, f3)$.

1.3 Contributions

Referring to the problem formulation, we provide contributions for the functions f_1 (object detection), f_2 (mass estimation), f_3 (affordance segmentation). Considering a camera framing a human manipulating an object e.g., a container, our aim is to design a method to detect the manipulated object and estimate the mass regardless of the content. Moreover, we tackle the problem of the detection in case of resource-

constrained devices and the affordance segmentation in case of hand occlusion. The contributions of the thesis are the following:

1. A procedure to locate the objects of interest in the scene. In case of a human manipulating a container, we propose to select the manipulated container based on the average distance computed from the depth map of a fixed-frontal camera. The assumption is that the container manipulated by a human will be the nearest with respect to the fixed-frontal view e.g., when preparing a handover [C2]. In case of wearable robotic applications, a human indirectly controls the camera on a robotic hand, moving it towards the object of interest. Since the human knows what object to grasp (object of interest), we propose to learn the location of objects regardless of the category (objectness) using a transfer learning technique called fine-tuning (more details in Appendix A.2). We fine-tune lightweight models to target resource-constrained devices that are common in wearable applications [C1]. The container selection procedure obtains an object localisation success of 90% on a human object manipulation dataset, and the objectness fine-tuning improves the detection performance on two affordance detection benchmarks. We publicly release the code of the container selection procedure[1].

2. A model to estimate the mass of a hand-occluded container regardless of the content, using both color (RGB) and geometric information. The rationale behind the model is that the object mass depends not only on the appearance that provides cues about the material, but also on the dimensions. The model processes RGB crops of the objects, the average distance of the object from the fixed-frontal camera, and aspect ratio of the object crop compared to the image resolution [C2]. Results show that the model is able to learn from images having similar containers, yet in different configurations, e.g., lighting conditions or filling. The combination of the mass estimation model with the mentioned container selection results in a pipeline for container localisation and mass estimation. The pipeline outperforms most of the baselines on two test sets

[1] https://github.com/CORSMAL/Visual

containing never seen object instances. We publicly release the code and the trained model[1].

3. A pipeline composed by the sequence of objectness detection and affordance segmentation models to restrict the input of affordance segmentation models to objects, instead of the whole scene [C1]. Moreover, we design a model to segment affordances in a hand-occluded setting, separating the object and arm regions. To achieve the separation between arm and object, our fusion module weighs the feature maps to penalise pixels outside those regions [C3]. The pipeline improves the performance of lightweight affordance segmentation model overcoming the assumption of object in foreground and completely visible [90]. The designed affordance segmentation model trained on mixed-reality images of hand-occluded containers shows better generalisation performance than previous models to unseen backgrounds and object instances on both mixed-reality and real data. We publicly release the code of the pipeline and the trained object detection models[2], the hand occluded object affordance segmentation code and the trained model[3].

4. The extension of the annotation of a mixed-reality dataset consisting of hands occluding containers [120], to train and test the models on affordance segmentation, due to the lack of dataset in hand-occluded setting [C3]. We publicly release the affordance annotation of the mixed-reality dataset[4].

1.4 Organisation of the thesis

This thesis is organised as follows:

Chapter 1: we introduced the problem of visual affordance prediction with the overall framework considered in the thesis to locate objects, estimate their mass, and segment the regions of potential interaction for an agent. Also, we listed the contributions of the thesis.

[2] https://github.com/SEAlab-unige/ICECS-2021
[3] https://github.com/SEAlab-unige/acanet
[4] https://doi.org/10.5281/zenodo.5085800

Chapter 2: we review the literature of visual affordance prediction discussing different perspectives on this problem. We focus the analysis on the mass estimation and affordance segmentation subproblems, highlighting the challenges and the limitations of available approaches; and we present datasets use to train and test models with the measures commonly used to assess the performance.

Chapter 3: we present the proposed pipeline to detect containers of interest in a recording based on the average distance, and we describe the model to predict the container mass regardless of the content. We detail the training procedure and the validation of the proposed method compared to the state-of-the-art techniques.

Chapter 4: we describe the pipeline to localise the objects in a scene and to segment object affordances. We present the proposed fine-tuning procedure to detect objects regardless of their class to support lightweight affordance segmentation models in resource-constrained applications. Moreover, we describe the proposed model to segment affordances in case of hand occlusions, along with the fusion module to penalise pixels outside hand and object regions. We present the annotation extension of a mixed-reality dataset to train and evaluate models on hand-occluded container affordance segmentation task; we discuss the results of the object detectors trained with the proposed procedure and we assess their impact on the performance of lightweight affordance detection models; we discuss the results of the proposed affordance segmentation model compared with the state-of-the-art techniques on mixed-reality and real data.

Chapter 5: we summarise the achievements presented in this thesis, and we discuss the future work.

Appendix A: we introduce some concepts and architectures that are useful to understand the thesis.

Chapter 2

Literature review

In this chapter, we discuss the criteria used for the literature search (Sec. 2.1). We describe advantages and drawbacks of different approaches to visual affordance prediction (Sec. 2.2). We present an overview of the methods concerning the mass estimation (Sec. 2.3) and affordance segmentation (Sec. 2.4) subproblems. We describe the details and the limitations of existing datasets to train and test models (Sec. 2.5), and we report the main performance measures used to evaluate object detection, mass estimation, and affordance segmentation (Sec. 2.6). To conclude, we summarise the main concepts of the literature review (Sec. 2.7).

2.1 Literature search criteria

The literature search was conducted to find works related to the visual affordance prediction problem. The main databases to conduct the search were Google scholar, IEEE Xplore, Springer, and some works were found searching the references of other papers. The main keywords used during the search were: 'grasping', 'affordance', 'grasping detection', 'affordance detection', 'affordance segmentation'. We considered works related to Computer Vision and Robotics using only RGB or RGB-D input, from 2015 onwards because dataset started to grow in size including tens of thousands of annotated images and deep learning methods started to show better generalisation results than hand-crafted features methods. We divided the collected publications about affor-

dance prediction (total of 59 papers) into: preliminaries (26 papers), mass estimation of objects and hand-occluded containers (10 papers), affordance segmentation/detection (23 papers).

2.2 Preliminaries

The term 'affordance' was coined to indicate the actions that the environment offers to the agent [31]. This broad definition suggests the intrinsic complexity of the problem of understanding and predicting affordances that is studied in different fields such as Ecological Psychology [9, 110], Neuroscience [85], Human Computer Interaction [91], Computer Vision [44], and Robotics [57, 133]. The definitions given by psychologists are partially overlapping, due to the problem complexity that in humans involves perceiving, reasoning, and acting. For example, Gibson defined affordances as directly perceivable [31], while Stoffregen and Chemero as relation between agent and environment [9, 110]. In contrast with the direct perception of affordances, in Robotics and Computer Vision, the use of machine learning models implies building a representation of the environment [133]. In particular, in Robotics the concept of affordance is mapped into grasping that allows the connection between the object and the agent. Approaches are divided in two types [57]: one generates grasping poses based on the physical and geometric modelling of both the object and agent (analytical approach), while the other one learns from observed samples of grasping poses (data-driven approach). Analytical approaches assume a perfect knowledge of the system and the environment to compute the grasp pose, while data-driven approaches aim at generalising from the observed sample poses. Among these two, the most consistent approach with Neuroscience and Ecological Psychology is the grasp synthesis, because it uses the perception and image features to generalise to unknown categories [57]. Most of existing works in grasp synthesis with data-driven approaches tackled the problem considering the grasping as functional to the object pick-up, ignoring the grasping as functional to different types of action (definition of affordance) [6, 29]. In this thesis, we adopt the definition related to the functional interaction with an object

predicted from a visual input [44, 80, 85]. Through data-driven approaches we aim at identifying the regions that the agent can use to perform an action on the object (grasping), distinguishing them from the regions that the agent can use to perform actions with the object (e.g., cut, contain, pound). Although the segmentation of affordance regions is independent of the robotic arm, it can be mapped to the robotic arm primitives that execute the actions using off-the-shelf control strategies [24, 82, 129].

Visual affordance prediction can be tackled using different approaches e.g., recognition, grasping detection, keypoint detection, segmentation, each one represents a different perspective on the problem with its own advantages and drawbacks. The affordance recognition predicts the actions that can be performed with objects in the scene either from human demonstration [48, 87], or from images of the environment [112, 140]. However, the absence of a segmentation process to associate affordances to regions of objects constitutes a high level reasoning, not sufficient to allow the physical interaction of a robotic hand [48, 112, 140]. The position of graspable points on the object surface can be learned to identify where the agent can perform the interaction [70, 88, 127]. These methods are mostly trained on simple conditions such as objects on a tabletop, so the keypoint detection could fail with different backgrounds especially if the method is using the color information. Alternatively, the graspable regions of an object can be detected using a rectangle to represent the 5 degrees of freedom of a parallel plate gripper on the image plane [2, 3, 18, 61, 64, 92, 135]. In particular, 1 degree of freedom is used to encode the gripper rotation with respect to the horizontal axis, 2 for the translation of the gripper center (on the horizontal and vertical axis), 2 for the geometry of the gripper (1 for the opening width, and 1 for the fingers length). The underlying assumptions are the depth map availability to obtain the full 7 degrees of freedom representation of the gripper in the space (3 degrees of freedom for translation, 3 for rotation and 1 for the opening width) and the fact that the objects can be grasped almost everywhere on the surface which is not always true. Distinguishing functional regions helps to avoid grasping some regions e.g., the internal surface of a cup filled with liquid. Moreover, most of the scenarios targeted by the grasping detection techniques are in controlled conditions: objects on

a tabletop or on the floor, with a top-down framing of the camera. These conditions may cause models to fail the detection when deployed in other scenarios, where the view is not top-down or hand occlusion is present due to a person holding the object. Recently, affordance prediction problem has been approached as one shot affordance detection exploring the selection of objects of interest [69, 134], or the segmentation of affordance regions [41], based on the similarity between a support set of images and a query set of images. One of the limitations in these approaches is that the support image is supposed to be similar to the query ones implying that the object category in the scene should be known a priori.

2.3 Mass Estimation

The estimation of container properties such as its mass represents a crucial stage to prepare the interaction with objects. In a collaborative scenario like human-robot handover, the mass clue can help the robot planning the movement and regulating the force to hold the object during the object retrieval and the maneuvering [86]. Mass estimation methods can process different data sources (see Table 2.1) to overcome challenges such as transparent objects with noisy depth data, occlusions due to human manipulation, variation in filling level, shape, stiffness, material and pose, that affect the appearance of the container and the sound during a manipulation e.g., filling. The container mass can be indirectly retrieved by combining two properties: filling mass and empty container mass. The filling mass can be seen as the result of three contributions: filling type classification, filling level and container capacity estimation [123]. To perform filling type classification, audio is one of the most used modality, either processing spectrograms [15] or classical audio features [13]. Solutions for filling level estimation can exploit only audio modality [55] or the combination with visual data [54, 67]. Visual information represents the main modality used to estimate the container capacity. The detection problem is tackled either using heuristic methods based on the handover application [15], object detectors [67] or instance segmentation models [54]. The task of capacity estimation can be considered as a regression

Table 2.1 Literature methods for object mass estimation.

Ref	Input			Views	Task	
	RGB	D	A		FM	EM
Donaher et al. [25]	○	○	●	-	●	○
Liu et al. [67]	●	○	●	4	●	○
Iashin et al. [54]	●	●	●	4	●	○
Ishikawa et al. [55]	●	●	●	1	●	○
Christmann et al. [15]	●	●	●	1	●	○
Wang et al. [119]	●	●	○	1	○	●
Matsubara et al. [72]	●	●	○	3	○	●
Apicella et al. [C2]	●	●	○	1	○	●

KEY – ●: considered, ○: not considered, D: depth,
A: audio, Views: viewpoints, FM: filling mass,
EM: mass regardless of the content.

problem, employing either simple Convolutional Neural Networks on single fixed frontal view depth data [15] or distribution fitting via Gaussian processes using object category as a prior across multiple views [67]. Otherwise, the segmented container can be approximated to a primitive shape in 3D, computing capacity as a by-product [55], or using volume formulas [54].

Recently, few works have provided solutions to the problem of estimating the empty container mass from visual data [72, 119]. The detection of the container is tackled using lightweight object detector (YoloV5[5]), while the container mass is learned using an augmentation strategy to vary the dimension of containers modifying accordingly the mass, and a variance minimization to ensure consistent predictions of the same container at test time [119]. Alternatively, the container detection is performed by the Localisation and object Dimensions Estimator (LoDE) [126] using with Mask R-CNN [46] to predict the object mask, while the empty container mass is regressed using a custom Convolutional Neural Network that combines: the patch of the container extracted using a formula to select the most visible view (across frontal, left and right views of the scene), the symmetrically restored object mask from left side fixed view, and information about container dimensions (height, width at the

[5] https://github.com/ultralytics/yolov5

bottom, width at the top) [72]. However, this method is not suitable in case a single camera is available, because of the selection formula that requires more than one view. Moreover, LoDE obtains an inaccurate geometric information of the container when the depth map is incomplete, for example in case the container is partially occluded or transparent. Another limitation of LoDE is about the symmetry of the object because the optimisation process generates concentric circumferences, hence only cilindrically symmetric objects can be reconstructed.

2.4 Visual affordance segmentation

Methods for visual affordance segmentation identify functional regions of objects e.g., graspable regions, adapting models for semantic or instance segmentation from images, but considering affordance classes as semantic labels [11, 44]. The affordance segmentation training is performed either with full supervision or with weak supervision. Full supervision is possible when the whole regions in images are annotated. Weakly-supervised learning studies techniques to learn from point-wise annotation that is less expensive and time-consuming than dense annotation. As shown in Table 2.2, most of the models use supervised learning paradigm to detect objects and segment affordances, while very few models adopt weakly-supervised learning [20, 107], showing comparable performance to early state-of-the-art approaches. The majority of methods process RGB only images. RGB-D cameras are more expensive than RGB sensors and using depth maps does not provide a significant boost in terms of performance despite using additional information [8, 17, 62, 80]. To process the input depth, models exploit geometric features [80, 81], however these features seem to be dependent on the environment elements such as floor, tables and chairs, not fit when the framing contains only the target object and the table, which is the case of most affordance dataset (see Sec. 2.5). Methods using geometric hand-crafted features to learn affordances underperform compared to deep learning techniques [81]. Some of deep learning methods can be part of a two-stage approach including a detection step to locate objects of interest. Affordances are then predicted in the region cropped around

Table 2.2 Literature methods for affordance segmentation using RGB or RGB-D images. We exclude from this table other models that may be part of the affordance segmentation pipelines e.g., a separate object detector before the affordance segmentation model.

Model	**Col**	**Attention**			**OA**	**Aff. head**		**Obj. head**			**FS**
		SP	CH	SA		C	E	C	S	D	
S-HMP [80]	○	○	○	○	○	○	○	○	○	○	●
ED-RGBD [81]	○	○	○	○	○	○	○	○	○	○	●
ADNet [8]	○	○	○	○	○	○	○	○	○	○	●
PartNet [62]	○	○	○	○	●	○	○	○	○	○	●
Multi [17]	○	○	○	○	●	○	○	○	○	●	●
A4T [58]	○	○	○	○	●	●	○	●	○	●	●
DeepLabAff [107]	●	○	○	○	○	○	○	○	○	○	○
MobileNetAff [90]	●	○	○	○	○	○	○	○	○	○	●
BB-CNN [82]	●	○	○	○	○	○	○	○	○	○	●
RN50-F [53]	●	○	○	○	○	○	○	○	○	○	●
DRNAtt [38]	●	●	●	○	○	○	○	○	○	○	●
STRAP [20]	●	○	○	●	○	●	○	○	○	○	○
GSE [137]	●	○	●	○	○	○	○	○	○	○	●
SEANet [130]	●	●	●	○	○	○	●	○	○	○	●
ADOSMNet [10]	●	○	○	○	●	○	○	○	○	○	●
RANet [139]	●	○	●	○	●	○	○	○	○	○	●
BPN [129]	●	●	●	○	●	○	●	●	○	●	●
AffordanceNet [24]	●	○	○	○	●	○	○	●	○	●	●
ESPNet [115]	●	○	○	○	●	○	○	●	○	●	●
B-Mask-RCNN [79]	●	○	○	○	●	○	○	●	○	●	●
ACANet [C3]	●	○	○	○	●	○	○	○	●	○	●

KEYS – ●: considered, ○: not considered, Col: RGB only, OA: object-affordances relation, FS: full supervision, SP: spatial, CH: channel, SA: self-attention, C: classification, E: edge segmentation, S: segmentation, D: detection.

the detected object or the corresponding feature map [14, 16, 24, 44, 82, 129]. The two steps can be tackled independently in a cascade approach [10, 82], to extract the regions of interest, or learned in an end-to-end manner using multi-tasking [14, 24, 129]. For example, AffordanceNet [14, 24] replaces the instance segmentation branch of Mask R-CNN [46] to predict affordance classes of the regions of interest localised in the feature map. The existence of a detection model is the assumption of many models in the literature [38, 53, 90, 130, 137], that are trained and tested with cropped images containing the annotated object(s). The rationale behind the cascade approach is to

have two models, one for detection and one for segmentation, that can be designed or replaced independently to improve accuracy or computational complexity. The dependence on the detection step can result in predicting the presence of an object in the wrong region of the image (false positive) or missing to locate the object (false negative), especially when the object is occluded by a hand. These mistakes lead to a propagation of the errors affecting also the affordance segmentation model that may predicts affordance regions in parts of the image without an actual object.

Recently, the attention mechanisms gained popularity in the instance and semantic segmentation tasks and some methods were adapted to segment affordances focusing on the object area or selecting more relevant features [38, 130, 137, 139]. For example, DRNAtt [38], inspired by Dual Attention Network [30], uses Spatial Attention Module to model contextual information (similarity between features in each pixel position) and Channel Attention Module to model channel inter-dependencies (similarity among channels) [30]. Due to the use of spatial attention, that works with low-resolution feature maps, DRNAtt predicts false positives if large portions of the image belong to the background, because at low-resolutions details of objects in the input image may be lost. This limitation is common with other approaches adopting the spatial attention [129, 130]. Another type of attention mechanism is the Shared Gradient Attention, that predicts a spatial attention for each channel of a feature map. The attention map is then combined with features maps to predict affordance segmentation and semantic edge detection [130]. However, this type of model predicts false positives in case objects are hand-occluded, transparent, or in cluttered environments because of blurred or missing edges. Apart from the methods that restrict the affordance segmentation to objects regions of interest e.g., two-stage methods, few models exploit the relation between the object classes and the affordance regions [10, 129, 139]. The link between objects and affordances is learned in the relationship-aware module predicting the object categories through an attention mechanism that weights the feature maps channels [139]. The fact that the relationship-aware module uses the whole feature map could cause the model to fail in case of multiple objects and classes are present or in case the object is partially visible due to occlusions. The other

method to learn the relationship between objects and affordances is the relationship attention module that processes the cropped feature map of a Mask R-CNN like model [129]. In particular, a multi-layer perceptron predicts the weights (attention) used to penalise the predicted affordance categories from the compressed (pooled) representations of the predicted semantic edge, of the predicted affordances, and of the predicted object classes. All mentioned methods focus on scenes with objects that are placed on a tabletop, and are either fully visible or partially occluded due to clutter [38, 81, 130, 137, 139]. These objects are often opaque or textured and easily distinguishable from the background, causing the affordance segmentation models to fail when observed objects are in more challenging poses or occluded, as in the case of hand-held objects when manipulated by a person. There is only one close work that tackles the segmentation of affordances of hand-occluded objects [53]. This work uses a ResNet-FastFCN [122] with a pyramid parsing module [138] to predict high-resolution outputs by using global contextual information (usually beneficial for segmenting general scenes). However, the model ignores the hand and the global contextual information does not help the model to capture fine affordance predictions on the object of interest. This can result in inaccurate segmentation of the affordances (e.g, predicted on the hand region). Even if the occlusions are common in realistic scenarios, very few affordance segmentation methods explicitly handle them. Thus, the problem of segmenting the affordance when occlusions are present remains still open and could enable collaborative scenarios.

2.5 Datasets

The estimation of objects mass from images is underexplored in the literature, hence the limited presence of datasets. Image2mass [109], consists of images of single object on a white background crawled from Amazon with the mass provided by the object description in the online store. However, the white background could influence the generalisation performance of trained models to real scenes (see Fig. 2.1a). Most of

(a) Image2mass

(b) CCM

Fig. 2.1 Samples from mass estimation datasets: (a) Image2mass [109] and (b) CCM [125]. Image2mass has different object types from toys to tools with different materials and colors, distinguishable from the background (images taken from the paper). CCM focuses on manipulated containers with different view, background, lighting, and interactions. Images are resized at the same height for visualisation purpose.

existing solutions use the CORSMAL Containers Manipulation (CCM) dataset [125], that includes annotated audio-visual recordings of people interacting with containers. The annotation is performed by weighting with a scale the empty container and the

Fig. 2.2 Examples of manual annotation on real data [80, 82]. Mistakes are highlighted using orange rectangles: not every object in the image is annotated (1st column), or the annotation has missing regions (2nd and 4th column), the annotation is over the hand occluding the object (3rd column).

filled container, and the mass of the content is computed by means of subtraction. The challenges in CCM dataset are the presence of hand occlusions that happen when people interact with the containers, and the varying appearance of containers due to the material, shape, and filling (see Fig. 2.1b).

Datasets for affordance segmentation are annotated labelling the pixels of the object regions with an affordance class [14, 16, 58, 60, 80, 82, 107]. The definition of the classes is decided as a convention in each work based on the object categories [60, 80, 82]. In general, each part is labelled with the action performed by a human when using the object for the purpose it was designed e.g., the blade of the knife is the part that is designed to *cut*, while the handle is *graspable* by the agent to perform the cutting. Annotators are instructed with the convention by experts or they are the experts themselves. Recent annotation tools (e.g., LabelMe [103]) allow an annotator to create a closed polygon around the object region that by convention belongs to a certain class and to label the region with the appropriate class. Manual annotation is a time-consuming task and it is subject to different kinds of error. Some objects in the scene might not be annotated, especially in case of clutter or if they are small for

Table 2.3 Characteristics of existing datasets for visual affordance segmentation.

Dataset	Images	Real	Tr	TPV	HO
UMD [80]	28,843	●	○	●	○
Multi-View [60]	47,210	●	○	●	○
HANDAL [40]	308,000	●	○	●	○
TRANS-AFF [58]	1,346	●	●	●	◐
IIT-AFF [82]	8,835	●	●	◐	◐
FPHA-AFF [53]	4,300	●	●	○	●
CAD120-AFF [107]	3,090	●	○	●	●
AFF-Synth [14]	30,245	○	○	●	○
UMD-Synth [16]	37,200	○	○	●	○
CHOC-AFF [C3]	138,240	◐	●	●	●

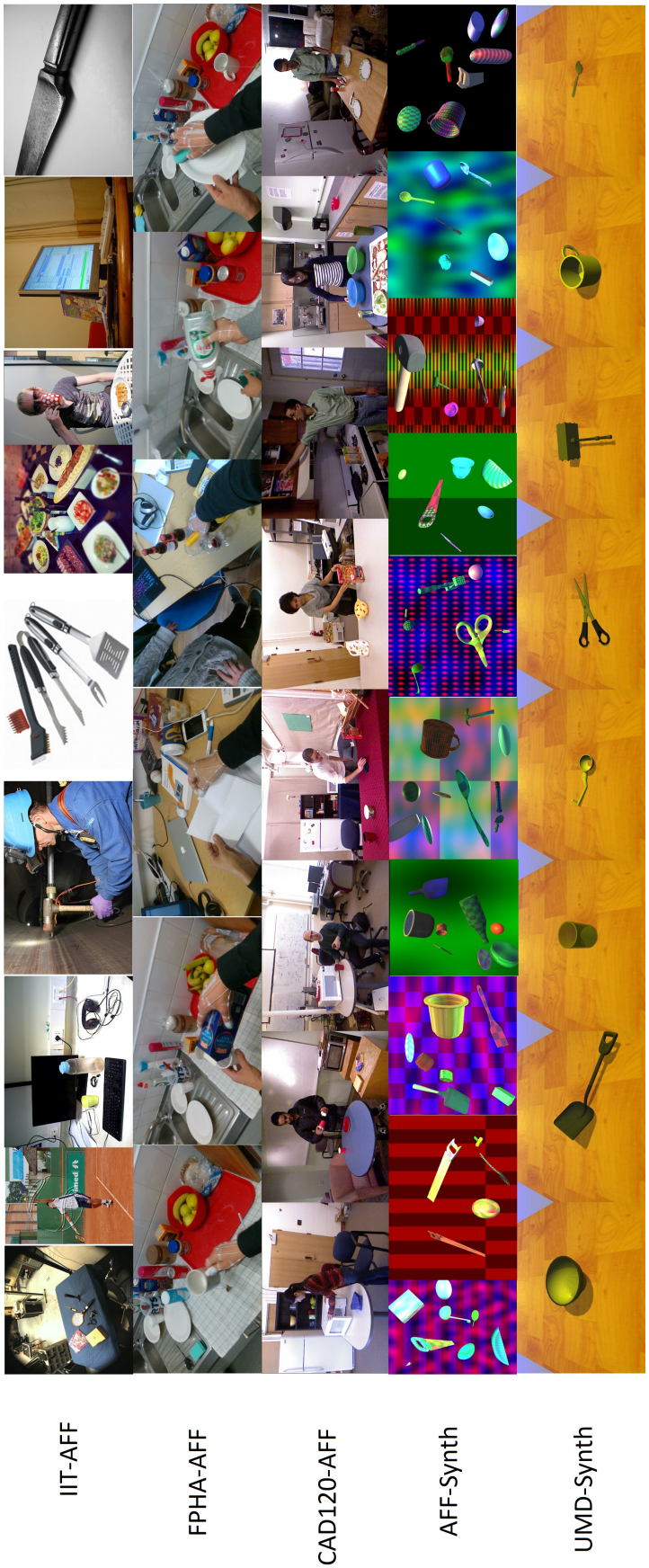
KEY - Real: real data, HO: hand-occlusion, Tr: transparency,
TPV: third person view, ●: considered, ○: not considered, ◐: mixed

example due to the distance from the camera. Moreover, the annotation might have holes or might not be completely over the object boundaries, because intensity values of the space are discretised in the pixels space, hence some pixels contain a blended intensity value between the object and the background. We show some samples of data manually annotated data in Fig. 2.2, highlighting human annotation mistakes e.g., missing affordance regions, or objects. In mixed-reality or synthetic datasets CAD models are annotated, the object (CAD model and texture) is created in a virtual scene and then a virtual camera renders the scene in an image [14, 16]. The projection of the synthetic scene in the camera frame is performed using synthetic lighting, and also the annotated regions can be projected in the camera frame to obtain the affordance annotation. The effort of the annotator is in labelling the CAD model regions.

The majority of datasets has real images captured from a third person perspective, and very few datasets contain hand-occlusions and transparent objects that increase the level of challenge (see Table 2.3). UMD [80] is composed by images of objects placed on a blue rotating table in the same environment, and has two types of annotations. In one case an annotator labelled pixels associating only one affordance class per region. In the other case, more than one annotator was involved to rank all affordance classes with the one associated to a region, allowing for multiple affordance classes for a

same region. Similarly, Multi-View [60] contains images of objects placed on a white rotating table keeping the same lighting, but includes a larger number of affordance classes and object categories. HANDAL [40] varies objects placement in outdoor and indoor environments such as a street or a living room, also containing clutter. Off-the-shelf methods predict the 6D pose of the objects in each frame and reconstruct the CAD model [121], then the annotator labels the handle of the CAD models as *graspable*. The final annotation mask is obtained by projecting the annotated CAD model in the camera frame through the estimated object and camera pose. The main limitations of HANDAL is the absence of hand occlusion and the fact that graspable regions are annotated only on objects having a handle. TRANS-AFF [58] is focused on transparent objects placed on a tabletop, increasing the difficulty to predict affordances, as objects are also in different poses. IIT-AFF [82] is composed by images of objects placed in a cluttered scene to better reflect a scenario with clutter and occlusions. One part of the dataset is collected from other datasets like ImageNet [109] varying the instances and the setting of the images. In CAD120-AFF [107] images are sampled from videos of humans performing activities in a realistic setting, e.g., kitchen, office, with more than one object in the scene and also hand occlusions. FPHA-AFF [53] has images of hand-held objects acquired from an egocentric point of view. However, the affordance annotation of this dataset is currently not publicly available and egocentric images contains arms from the bottom of the image, resulting in objects highly occluded by the hands. All the previous datasets are manually annotated but limited in their size, making the training of visual affordance segmentation model a problem. The need for thousands of labelled data has pushed towards the generation of synthetic datasets. UMD-Synth [16] simulates UMD in a synthetic manner and has objects in different backgrounds and object poses, but provides only one affordance per object region. AFF-Synth [14] has images generated using domain randomization techniques to overcome the gap between simulated and real data. Unlike UMD-Synth, each image of AFF-Synth includes multiple objects, some of them can be used as distractors during training, since they do not have affordance annotation. These synthetic datasets, however, are still small in size and do not contain occlusions, especially in the case of

objects held by a person. Fig. 2.3 shows sampled images from the mentioned datasets. Note that the object instances and background conditions vary among datasets, and that the categories available are overlapping mainly for containers such as mugs and cups.



(Figure continues in the next page)

Fig. 2.3 Sampled RGB images from affordance segmentation datasets: UMD [80], Multi-View [60], HANDAL [40], TRANS-AFF [58], IIT-AFF [82], FPHA-AFF [53], CAD120-AFF [107], AFF-Synth [14], UMD-Synth [16]. Datasets have different object instances and background conditions from single object on a plain colored background to clutter or hand-occluded objects. Images are resized at the same height keeping the aspect ratio for visualisation purpose.

2.6 Performance measures

Along with models and datasets, the literature provides performance measures i.e. quantitative ways to evaluate and compare models predictions. Performance measures compare the prediction with the annotation through a function that outputs a scalar value. Based on the function, the performance can be better with a small value and worse with a large value, or vice-versa.

To evaluate the performance of a mass estimation model on visual data, we use the relative absolute error (ε) and the mass estimation score (s) between the estimated measure $\hat{m}^j$, and the true measure m^j [124]. Given a set of recordings $\{v_j | j = 1, ..., V\}$, we compute the relative absolute error as:

$$\varepsilon(\hat{m}^j, m^j) = \frac{|\hat{m}^j - m^j|}{m^j}, \tag{2.1}$$

were j is the index of a single recording. The relative absolute error is an unbounded function greater than 0. ε has a value over 1 when the estimated mass is greater than the annotated mass, and close to 0 when the estimated mass is lower than the annotated mass. The score $s \in [0,1]$ is used to average the relative absolute error across all V recordings[6]:

$$s = \frac{1}{V} \sum_{j=1}^{V} \mathbb{1}_j \, e^{-\varepsilon(\hat{m}^j, m^j)}. \tag{2.2}$$

The value of the indicator function $\mathbb{1}_j \in \{0,1\}$ is 0 only when $\hat{m}$ in recording j is not estimated. The score has value 1 when the estimation error is 0 (the predicted mass and annotation are equal). This score might not be ideal to compare models in applications in which it is preferable predicting a greater mass value rather than a lower one e.g., a container composed by a hard material with dangerous filling. If the model predicts a mass lower than the actual value, the interaction between the robotic hand and the object could cause the container to slip or to spill the content.

The performance of object detectors on a set of images $\{I_n | n = 1, ..., N\}$ is evaluated using the mean Average Precision (mAP), that depends on the Jaccard Index. The

[6] https://corsmal.eecs.qmul.ac.uk/challenge.html

Jaccard Index or Intersection over Union (IoU) is a score in $[0,1]$ range used to count bounding boxes true positives, false positives, and false negatives, since it takes into account how much predicted bounding box is overlapped with the annotation and how much they are similar in size. Given a predicted object class (score is used to rank the predictions in descending order) and a IoU threshold, we first compute true positives (TP), false positives (FP), and false negatives (FN). A true positive is a predicted bounding box that has an IoU score with the bounding box of the same class over the threshold, a false positive is a predicted bounding box that has an IoU score with the bounding box of the same class below the threshold. A false negative is a ground truth bounding box that has not a corresponding prediction over the IoU threshold. Precision $P = \frac{TP}{TP+FP}$ and recall $R = \frac{TP}{TP+FN}$ can be computed, and precision-recall curve is built interpolating points. The mean Average Precision (mAP) score is calculated by taking the mean area of Precision-Recall curve over all classes and/or overall IoU thresholds. Instead of computing the integral of the curve to retrieve the area, an approximation is used by sampling points computed as:

$$mAP = \sum_{q=1}^{Q} (R_q - R_{q-1})\hat{P}(R_q)\,, \tag{2.3}$$

where Q is the number of interpolation points, $\hat{P}(R_q) = \max_{\tilde{R} \geq R_q} P(\tilde{R})$ is the precision value at R_q. The computation of the mean average precision depends on the chosen number of interpolation points, the more the better. For this reason, we adopted the standard COCO score that uses 101-point interpolated curve averaging the results also varying the IoU threshold.

To evaluate and compare the affordance segmentation models, we compute the per-class precision, recall, and Jaccard Index as percentages across all images of a given dataset $\{I_n | n = 1, ..., N\}$. Precision measures the percentage of true positives among all positive predicted pixels. Recall measures the percentage of true positive pixels with respect to the total number of positive pixels. The Jaccard Index measures how much two regions with the same support are comparable. In general, when comparing multi-dimensional data like segmentation mask, using a number to represent the comparison

leads to information loss e.g., in what regions the prediction and the the two images are different. To compute precision, recall, and Jaccard Index, we first compute true positives (TP), false positives (FP), and false negatives (FN) between prediction $\hat{S}_n$ and annotation S_n for each RGB image $I_n \in \mathscr{D}$ and for each class c. A true positive is a pixel $x \in I_n$ that is predicted as class c in $\hat{S}_n$ and the corresponding pixel in S_n is annotated as c. A false positive is a pixel $x \in I_n$ that is predicted as class c in $\hat{S}_n$, but not annotated as c in S_n. A false negative is a pixel $x \in I_n$ that is not predicted as class c in $\hat{S}_n$, but the corresponding pixel in S_n is annotated as c. We therefore compute the per-class precision, P, the per-class recall, R, and the per-class Jaccard Index, J, as:

$$P = \frac{\sum_{n=1}^{N} \sum_{x \in I_n} TP}{\sum_{n=1}^{N} \sum_{x \in I_n} TP + FP}, \tag{2.4}$$

$$R = \frac{\sum_{n=1}^{N} \sum_{x \in I_n} TP}{\sum_{n=1}^{N} \sum_{x \in I_n} TP + FN}, \tag{2.5}$$

$$J = \frac{\sum_{n=1}^{N} \sum_{x \in I_n} TP}{\sum_{n=1}^{N} \sum_{x \in I_n} TP + FP + FN}. \tag{2.6}$$

Note that we keep the notation simple and we did not include the index of the class c in the above equations, as all the results will refer to the per-class measures. One of the limitations of the described measures is that they do not take into account the distance of the mistake e.g., false positives, to the annotation. In affordance segmentation task it is preferable to have a region of false positives close to the annotation, rather than in another part of the image support because it means that the model predicts the affordance region close to the object rather than in the wrong portion of the image.

Another performance measure for affordance segmentation is F_β^w [71]. The key idea is to attribute different importance to different prediction errors, through weighting functions. Y is column-stack representation of the ground-truth segmentation mask S, while $\hat{Y}$ is column-stack representation of the predicted segmentation mask $\hat{S}$. Weight functions are a A, which captures the dependency between foreground and background

pixels, following the relation:

$$A(i,j) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z(i,j)^2}{2\sigma^2}} & \forall i,j \quad Y(i) = 1, Y(j) = 1 \\ 1 & \forall i,j \quad Y(i) = 0, i = j \\ 0 & otherwise \end{cases}, \tag{2.7}$$

where $z(i,j)$ is the Euclidean distance between pixel i and pixel j. σ^2 controls the influence of pixels that are farther away. The larger σ^2 is, the greater the influence of distant pixels.

The varying importance of pixels is captured through the function D that weighs false detections based on their distance from the foreground

$$D(i) = \begin{cases} 1 & \forall i, Y(i) = 1 \\ 2 - e^{\alpha \cdot \Delta(i)} & otherwise \end{cases}, \tag{2.8}$$

where $\Delta(i) = \min_{Y(j)=1} z(i,j)$. The constant α determines the decay rate.

The per-class F_β^w is computed as:

$$F_\beta^w = (1+\beta^2) \frac{P^w \cdot R^w}{\beta^2 \cdot P^w + R^w}, \tag{2.9}$$

where $P^w = \frac{TP^w}{TP^w + FP^w}$ is the weighted precision and $R^w = \frac{TP^w}{TP^w + FN^w}$ the weighted recall. In particular, $TP^w = (1 - E^w) \cdot Y$, $FP^w = E^w \cdot (1 - Y)$, $FN^w = E^w \cdot Y$, where $E^w = \min(E, EA) \cdot D$ is the combination between the error $E = |Y - \hat{Y}|$ and the weighting functions A and D. The parameters σ in A and α in D should be tuned for the specific application and dataset, because the measure takes into account the distance in terms of pixels, that is not the same as considering distances in the actual scene (two small objects might seem far from each other if the camera is very close to them). We used the default values to compare methods, as in the literature. Moreover, due to the mathematical definition, F_β^w can not be computed for the classes that are not in the annotated mask, hence a part of the prediction mistakes are not taken into account. To count all the mistakes, we computed the per-class F_β^w between the concatenation of

all the predictions and the concatenation of all the annotations, so that all classes are present [90]. However, this method might require a huge amount of memory to run based on the resolution of segmentation mask and the dataset size.

2.7 Summary

Visual Affordance Prediction is a challenging problem with different abstraction layers. In this chapter, we first described different perspectives on the affordance prediction problem, that can be approached as a classification, keypoint detection, grasp detection, segmentation. The main challenges are the varying appearance of objects due to factors such as shape, pose, lighting, material, and presence of occlusions either due to other objects or due to a human hand during an interaction. The literature review focused on the estimation of the object mass and the segmentation of graspable regions, that can support the control logic of a robotic-hand. The localisation is a common subproblem for mass estimation and graspable regions segmentation, and is approached using object detectors, or by leveraging multi-task learning. Mass estimation literature is mainly tackling the filling mass estimation and only two methods were designed for container mass estimation regardless of the content. Most of affordance segmentation methods adapt instance and semantic segmentation models to the task of affordance segmentation by changing the output classes, few models explore the relationship between the object and the affordances and the objects. Only one method was targeting the case of hand-occlusion, however from a first person perspective and without segmenting the hand that may cause the method to segment as graspable the human hand. Although object categories and instances in datasets vary, they are mostly placed on a tabletop with a simple background. Objects are often opaque or textured and easily distinguishable from the background, causing trained models to fail when objects are in more challenging poses or occluded. Even though in realistic scenarios occlusions are common, the state-of-the-art datasets and solutions considering them are very limited in number, hence the problem of predicting object properties and consequently affordances, remains still open. This thesis focuses on the mentioned

limitations, proposing a method to estimate container mass regardless of the content taking into account color and geometric information, a method to segment graspable regions separating them from the hand occluding the object, and the extension of annotation of a mixed-reality dataset of hand-occluded containers with affordances to train and test models.

Chapter 3

Container mass estimation

In this chapter, we describe the localisation and mass estimation sections of the overall framework (highlighted in Fig. 3.1). We present the proposed method to locate a manipulated container and estimate its mass independently of the presence of the content, from a fixed-frontal RGB-D camera (Sec. 3.1); we show the experimental setup and the quantitative comparison with other methods (Sec. 3.2); and we draw the chapter conclusions (Sec. 3.3).

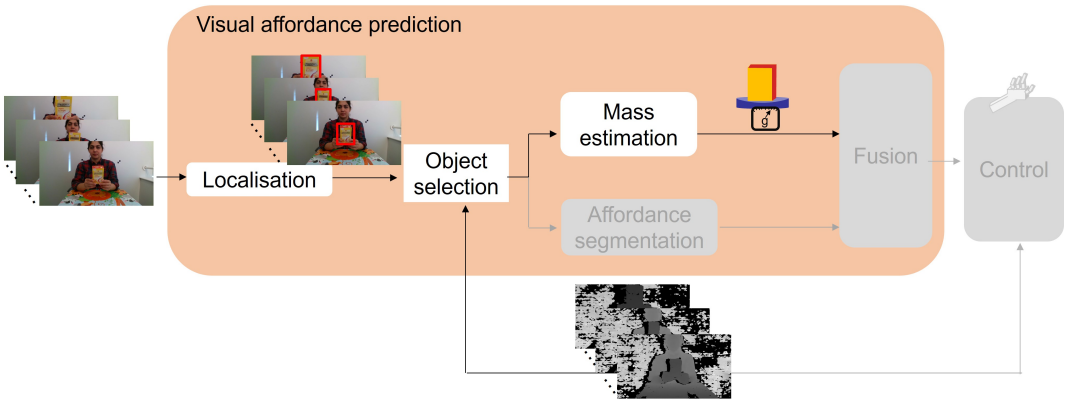


Fig. 3.1 Overall framework considering localisation and mass estimation phases. An instance segmentation model locates objects in the RGB image. The selection phase extracts the patches (image crops) and discards crops based on the average distance computed from the depth map. Then, the mass estimation model predicts the mass of the object. The other phases of the framework (affordance segmentation, fusion, and control) are not considered.

3.1 Container localisation and mass estimation

The proposed method runs on RGB-D sequences to predict the mass of the container manipulated by a human regardless of the content. The rationale behind the pipeline is to first detect the objects in the frames and to filter out the containers that are not manipulated during the recording. To perform this filtering, we use a heuristic approach based on the fact that in case of handover the manipulated container will be offered, hence it will be the object in the scene closest to the fixed-frontal camera. We select K candidate containers to be more robust to mistakes in the filtering, and we average K mass values (one per-candidate) predicted by a custom model. The model that predicts the mass regardless of the content is designed to combine the color information with the average distance and aspect ratio information, since the color provides hints about the material, while the average depth and aspect ratios about the geometry. Our method can be combined with existing techniques for filling mass estimation [123] to obtain the complete mass of the manipulated container, and is divided into three phases (see Fig. 3.2):

1. *Localisation*: we locate the containers in the RGB sequence every n frames using an instance segmentation model.
2. *Patches selection*: the method automatically selects the K-nearest containers based on the average of depth values on the segmentation mask, assuming that the manipulated container is the nearest with respect to the fixed frontal view.
3. *Mass estimation:* the mass estimation model predicts one mass value per patch (K predictions in total). The container mass ($\hat{m}$) regardless of the content (empty) is computed as the average of the predictions.

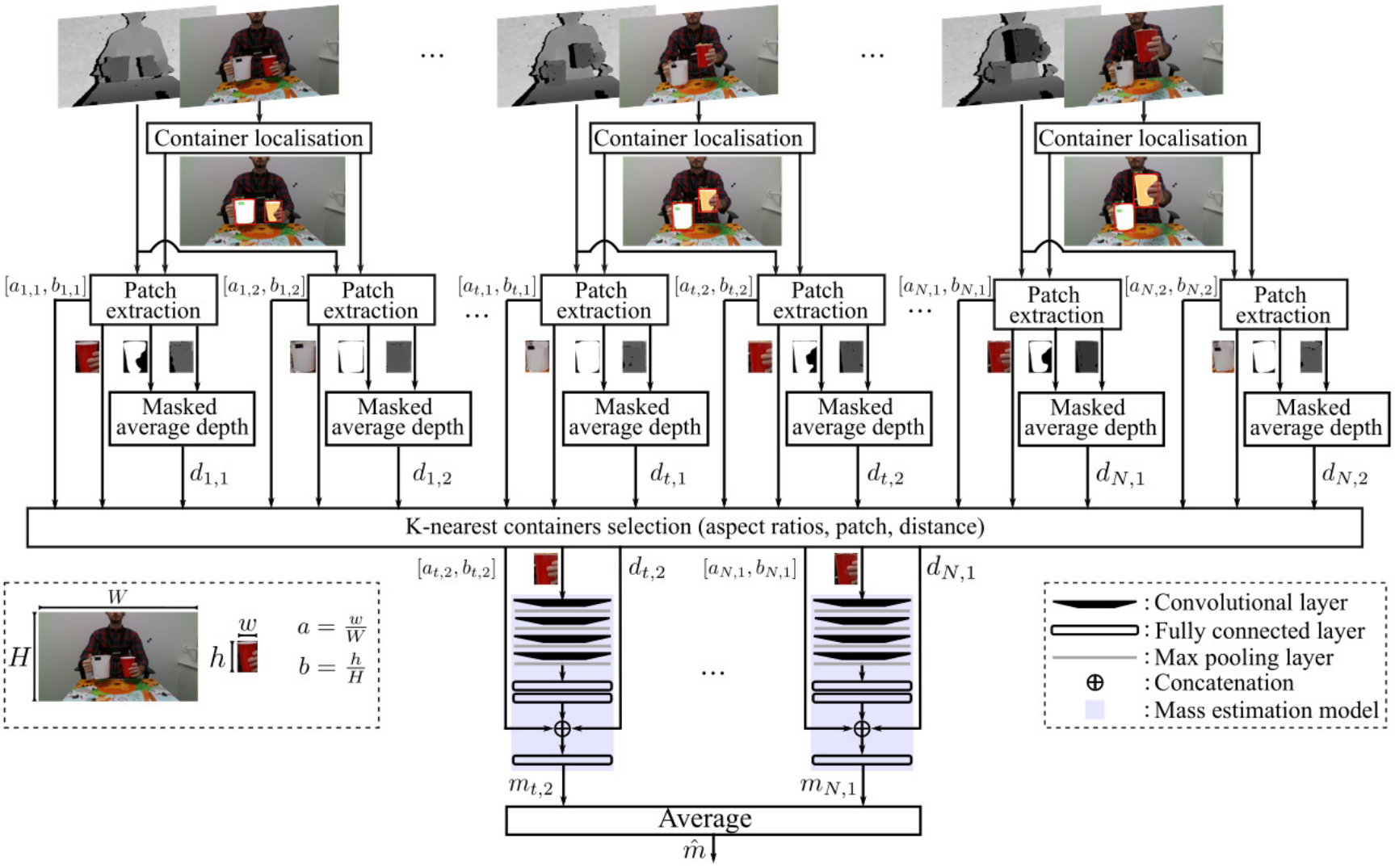


Fig. 3.2 Block diagram of the proposed approach. For each frame, containers are detected, then K nearest patches are selected leveraging the raw depth maps considered in the segmentation mask coordinates. The empty mass of each patch is predicted by the model which takes as input the RGB patch and triplets of values: aspect ratio width, a, and height, b, and average distance, d. The final empty mass estimation, $\hat{m}$, is the average of K mass predictions. Figure from [C2].

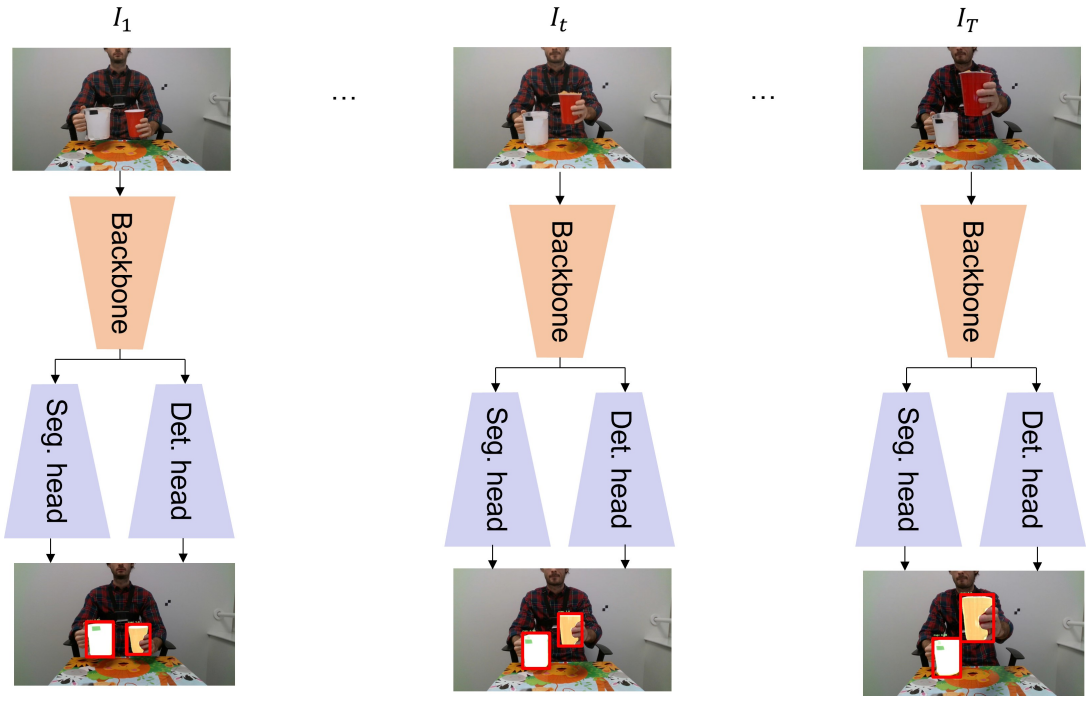
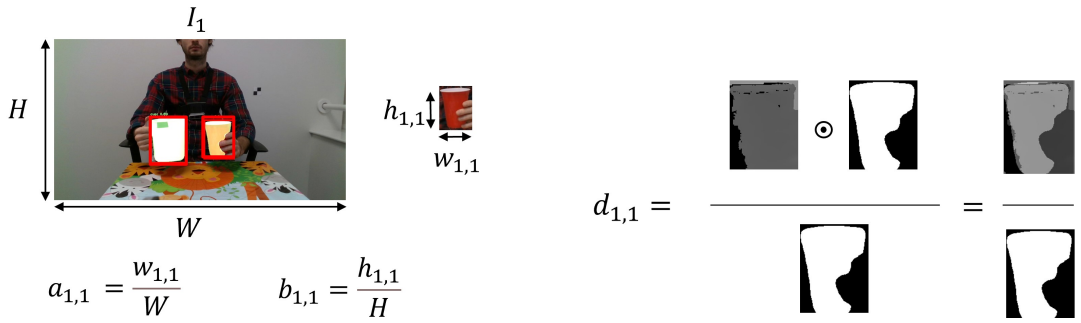


Fig. 3.3 Localisation phase performed by a generic instance segmentation architecture. In each frame, the model detects the objects through the detection head, and segments them using the segmentation head.

3.1.1 Localisation

The localisation phase (visualised in Fig. 3.3) aims at processing the RGB frames of the recording to detect objects. An instance segmentation model (more details in Appendix A.6), is fed with every RGB frame $I_t \in \{0,...,255\}^{W \times H \times 3}$ and outputs the object class $\theta_{t,o} \in \mathbb{N}$, the bounding boxes $\psi_{t,o} \in \mathbb{R}^4$ where the dimensionality is the top left corner coordinates of the rectangle, its width and height, and binary segmentation masks $S_{t,o} \in \{0,1\}^{W_o \times H_o}$ obtained thresholding the probability map, for each detected object o. We employ an instance segmentation model that predicts also the objects mask to isolate the pixels on the object region in the image. The instance segmentation model can output several classes, but if the object categories are known in advance they can be filtered to obtain only the categories of interest.



(a) Aspect ratio computation (b) Average depth computation

Fig. 3.4 Example of (a) aspect ratio and (b) average depth computation for a single object in a frame. The aspect ratios (a and b) are computed as division between the width and height of the patch and the width and height of the whole image. The average depth d is computed as division between the sum of depth values in the object mask pixels (obtained with the pixel-wise product $\odot$) and the number of pixels belonging to the object mask.

3.1.2 Patches selection

The patches selection phase aims at processing the output of the instance segmentation model used in the localisation phase to extract patches of the frames and select a subset of K detected objects in the frames sequence.

Given the RGB frame at time t (I_t), the outputs of the instance segmentation model from the localisation phase are the bounding boxes $\psi_{t,o}$, and the binary segmentation masks $S_{t,o}$ for each detected object o. We generate the RGB patch $I_{t,o}$ cropping I_t, the corresponding depth patch $I'_{t,o}$ cropping I'_t, in the regions delimited by the bounding boxes. As visualised in Fig. 3.4a, we compute the aspect ratio between the width of the crop and the original resolution of the image $a = \frac{W_o}{W}$, same for the height $b = \frac{H_o}{H}$. We compute also the masked average depth (see Fig. 3.4b) $d_{t,o}$ as the average of the depth values in the object mask:

$$d_{t,o} = \frac{\sum_x I'_{t,o}(x) \odot S_{t,o}(x)}{\sum_x S_{t,o}(x)}, \tag{3.1}$$

where x is the pixel position in the crops having the same support, $\odot$ is the Hadamard product or pixel-wise product between the binary segmentation mask $S_{t,o}$ and the corresponding depth map $I'_{t,o}$. The Hadamard product is used to consider only the

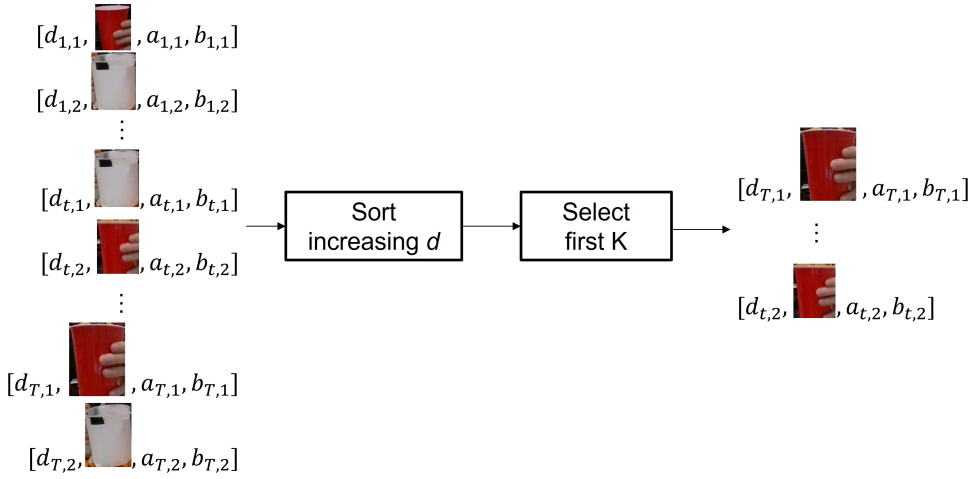


Fig. 3.5 Block diagram of the K-nearest candidates selection for a single recording. The set of candidate containers, the tuples depth, RGB patch, and aspect ratios are first sorted increasingly based on the average depth and the first K elements of the set are selected. The K selected candidates represent the closest patches to the fixed-frontal camera.

region of the depth map that is on the object surface, filtering out the regions of the depth map that are outside the predicted object, through the segmentation mask. To obtain candidate crops we repeat the operations of cropping, aspect ratio, and average depth computation, for all the frames until the end of the recording.

We select K tuples from the set $\{(I_{t,o}, d_{t,o}, a_{t,o}, b_{t,o})\}$ based on the lowest masked average distance $d_{t,o}$, selecting the patches that are the closest to the fixed-frontal view. As visualised in Fig. 3.5, first the tuples are ordered increasingly based on the value of d, then the first K are selected representing the patches with the lowest average distance. The assumption is that there are at least K detections in the frames sequence. Based on the value of K the performance of the proposed solution may change. If K is large, the algorithm includes most candidates detected by the instance segmentation model, also the ones that are in the scene but are not manipulated by the human. If K is too low, few candidates are included in the set, so the prediction could by affected by the detection of objects that are not of interest in the scene.

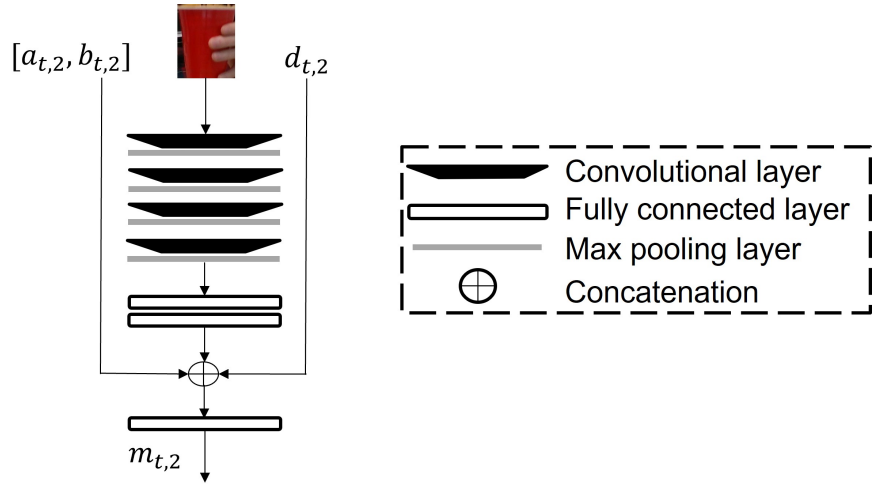


Fig. 3.6 Mass estimation model. The input is the RGB crop of the object, the aspect ratios a and b, and the average distance d, at the time instant t and for the second object in the frame. Figure from [C2].

3.1.3 Mass estimation

The mass estimation model is run on the K outputs of the patches selection phase and the average is computed to predict the container mass regardless of the content $\hat{m}$. Averaging the results allows to fuse the predictions and be less affected by potential mistakes of the localisation and patches selection phases. Objects in the recordings that are not manipulated could be detected in the localisation phase and could be part of the set of K candidates in case the average depth is lower than other patches, influencing the mass estimation phase.

The mass estimation model, shown in Fig. 3.6, takes as input RGB patches of the container, the width and height aspect ratios with respect to the original image resolution (a and b), and the average distance of the object from the camera (d). The idea behind the features ($f = [a, b, d]$) is to consider the geometrical information of the distance between the container and camera, along with the aspect ratios. Similarly to a method for container capacity estimation [15], the model has four convolutional layers, two Fully Connected (FC) layers, a concatenation of the output of the second FC layer with f, and one FC layer after the concatenation. Batch normalization and ReLU activations are used after each layer. Max Pooling is used after each convolutional layer. The size of convolutional kernels is $(3,3)$, paddings and strides are $(1,1)$, and

channel dimensions are $(32, 64, 64, 128)$. The size of Max Pooling kernel is $(2, 2)$. The output of the first two FC layers has dimension 64 and 6, respectively. Unlike the mentioned method [15] that uses depth patches and their aspect ratios $[a, b]$, our model takes as input RGB patches and the features f.

3.2 Validation

3.2.1 Methods under comparison

We compare our method with two baselines and two other methods for container mass estimation regardless of the content [72, 119]. Method 1 (*M1*) [119] employs a MobileNetV2 [106] with Coordinate attention [50] to predict the mass from the object crops extracted using YoloV5[5], while Method 2 (*M2*) [72] predicts the container mass using a custom Neural Network that combines unoccluded segmentation mask and geometric information. In M2 the container detection is performed using Localisation and object Dimensions Estimator (LoDE) [126] with Mask R-CNN [46], while the unoccluded object mask is obtained by symmetrically restoring the object mask [72]. In general, some similar features across the methods can be highlighted e.g., the employment of two-stage approach (first detection then mass estimation) to tackle the problem and the use of lightweight models to perform the mass prediction. As baselines for comparison, we consider a pseudo-random generator that draws the predictions from a uniform distribution in the interval $[1, 351]$ based on the Mersenne Twister algorithm [73] (*M4*), and an average method that computes the average of the mass annotations (*M5*).

3.2.2 Experimental setup

We evaluate our proposed model on the CORSMAL Containers Manipulation (CCM) dataset consisting of 1140 audio-visual recordings [125]. During each recording, a person interacts with a container (e.g., filling a cup with rice contained in a pitcher) and then prepares for the handover. Videos differ in conditions such as lighting, person's

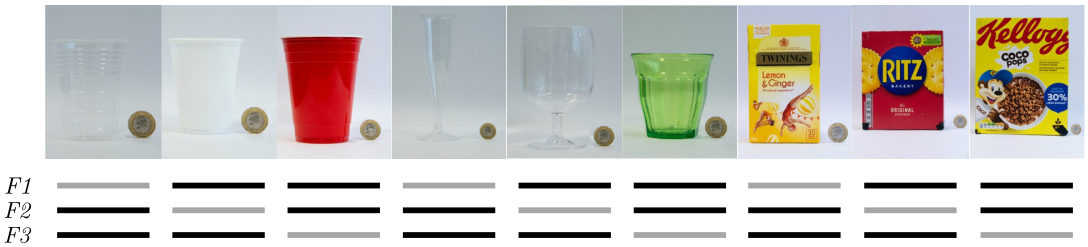


Fig. 3.7 3-fold cross-validation setups ($F1$, $F2$, $F3$) of the CCM training set. Each fold selects videos from one instance of each container type as testing set (—), while videos belonging to the other instances are used as training set (—). Figure from [C2].

Fig. 3.8 Public testing sets containers. The containers instances are different from the ones in the training set. Private testing set instances were not released.

clothing, hand occlusions. The total number of containers in the dataset is 15: 9 constitute the training set (684 recordings), the other 6 are evenly split into a public testing set (228 recordings) and private testing set (228 recordings).

To assess the generalization performance of the method during training, we first perform 3-fold cross-validation on CCM training set leaving one instance per category (*box*, *glass*, *cup*) out [54], see Fig. 3.7. For each testing fold, the training set includes containers belonging to the other two folds, which are split into training and validation sets using 80% and 20% as respective percentages of data. The validation set is used for model selection and the best model, the one with the lowest mean loss, is kept for the testing fold. We compute the per-class mean absolute error using Eq. 2.1, summing all the errors for a specific container type and dividing by the number of recordings with the specific type. The per-class scores are obtained in an analogous way, using Eq. 2.2.. The total score can be obtained using Eq. 2.2 and considering all the container types or by multiplying per-class score by the number of recordings with the specific type, summing, and then dividing by the total number of recordings.

Secondly, we evaluate the generalisation performance of our method on the CCM public and private testing sets having container instances different than the training

ones (see Fig. 3.8). To train our model, we randomly split the whole training set in training and validation with the 80% and 20% splits as before to include all available containers in the training phase. In this case, the best performing model (based on the lowest mean loss on the validation set) is tested through the CORSMAL Challenge which provides the results for the public and private testing set, since the annotations are private. In particular, we compare using our model, trained using the mentioned training and validations splits, with the baselines and the methods from the state-of-the-art considering the mass estimation score (see Sec. 2.6). On the public and private testing sets, we perform an analysis per container type and an overall analysis on the whole sets. We obtain the per-class score and the overall score as previously described in case of assessing the generalisation to different folds.

3.2.3 Training details

The mass estimation model is trained using the container crops extracted using the first two phases of our method (the localisation and patches selection) on the CCM dataset [125]. We perform the container localisation phase using Mask R-CNN [46] (more details about the architecture in Appendix A.6) pre-trained on COCO dataset [66], selecting only the output classes *cup*, *book*, *wine glass*, and *bottle* [54]. Mask R-CNN is applied to the entire dataset, similarly to Crop-CCM [77]. The main differences are that we use a single view of the scene, we do not restrict the model solely to cup and glass cases, and we do not perform a manual check of the results. The selection of the containers patches is indeed performed automatically during *K-nearest patches selection* phase. Images resolution is 1280×720, the detection threshold is set to 0.4 and every frame of the video is analysed ($n = 1$). In each frame, Mask R-CNN could find in general more than one object e.g., the pitcher used to fill the cup and the cup itself. The maximum number of considered containers candidates (K) is set to 5 (note that the model could also detect less patches in a recording). The rationale behind the parameter value choice is to have a trade-off to obtain enough patches to train the model and provide a more robust mass prediction, by averaging a number of candidates, as well as minimizing computational overhead. Our extracted dataset

Fig. 3.9 Sample patches of the extracted dataset. Black padding is applied before resizing to keep the same aspect ratio. Figure from [C2].

Table 3.1 Mass estimation model training details

Dataset	Setup	Aug. factor	Epochs	Loss	Learning rate	Batch
CCM crops	3-folds	3	100	MSE	0.0015	32
	whole	4	300			

consists of 3,408 patches, some of them are shown in Fig. 3.9. Almost 8% of these patches extracted from the training set represent the pitcher used to fill containers, which is not annotated in CCM dataset. This means that the first two steps of the procedure (localisation and patches selection) are effective in selecting the correct container to perform the mass estimation. The empty mass annotation of each video is applied to each extracted patch.

The mass estimation model is trained on the regression task following the details summarised in Tab. 3.1. The patches are resized to 112×112 resolution, using zero padding on the shorter dimension in order to maintain the proportions, and are normalized to [0, 1] range. The following transformations are employed to augment the patches dataset: horizontal flip with probability 50%, vertical flip with probability 50%, random rotation between 0 and 180 degrees without cropping the patch, color jitter which consists in randomly changing the brightness into $[0.8, 1.2]$ range, contrast into $[0.8, 1.2]$ range, saturation into $[0.8, 1.2]$ range and hue into $[-0.2, 0.2]$ range. During the 3-fold cross-validation experiment the training set is augmented of 3 times using the described transformations, and the mass estimation model is trained for 100 epochs. To validate our model using the public and private CCM testing sets, every patch in the cropped training set generates other 4 images using the described augmentation techniques and the model is trained for 300 epochs, as the number of training images increases with respect to 3-fold cross-validation. The aspect ratios, average distance

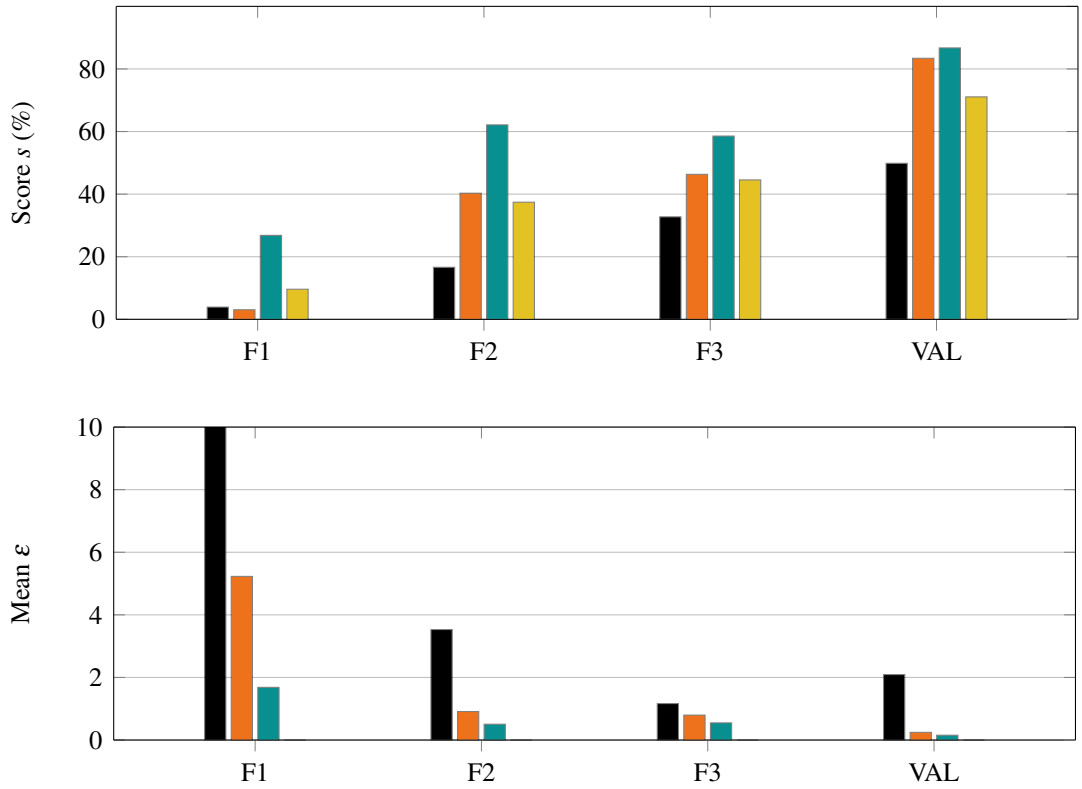


Fig. 3.10 Analysis per container type of 3-fold cross validation and random cross-validation of our proposed model for container empty mass estimation. Top: testing score s in percentage. Bottom: mean of relative absolute error ε. The maximum y-axis value is set to 10 for visualization purpose, the actual value for *cup* in *F1* is 22.95. Legend: — *cup*, — *glass*, — *box* — *total*

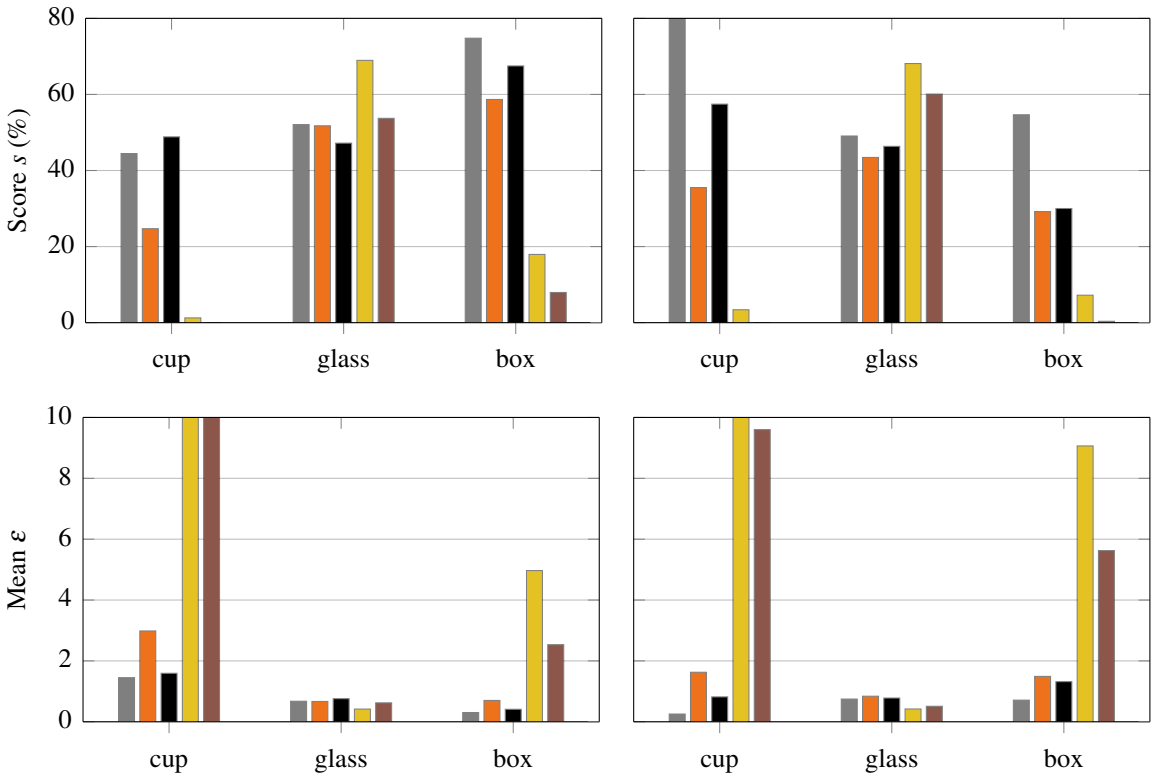
and empty mass labels are normalized using the minimum and maximum values retrieved from the training set. The following setup is common to the experiments: mean square error loss, batch size 32, Adam optimizer, learning rate 0.0015 with an exponential decay rate of 0.9985 and with decay steps equal to 20; weight decay is set to 0.001.

3.2.4 Results and discussion

Fig. 3.10 (top) analyses the per-class scores (colored based on *cup*, *glass* and *box*) and the *total* score for the three fold splits (*F1*, *F2*, *F3*) and for the whole validation set (*VAL*). The total score can be obtained using Eq. 2.2 or by multiplying per-class score by the number of instances, summing and then dividing by the total number of

instances. Fig. 3.10 (bottom) provides a complementary analysis through the per-class mean of relative absolute error ε. The trends in the scores are indeed opposite to the ones in the mean relative absolute error qualitatively meaning that when the error is low, the score tends to be high and vice-versa. Overall, Fig. 3.10 shows that the model does not generalize to testing containers significantly different from the training ones. The model achieves the highest score and the lowest mean error for the class *box*, probably due to the fact that they feature different colors and have a larger shape with respect to other classes. The mean relative absolute error in the *cup* cases suggests that the empty mass for this class is not properly learned. A possible explanation for the low performance on the first test fold is that the training images contain colored and opaque *cups*, whereas in the testing set *cups* are transparent, and the only transparent containers in the used training folds are *glasses*. Other folds statistics point out that the presence of similar containers between training and testing sets helps reducing the error and improving the score. The value of mean relative absolute error for *glass* and *box* classes falls in the $[0, 1]$ range, while for the *cup* class it decreases with respect to fold *F1*, yet it remains higher than 1. The last group of bars in Fig. 3.10 shows the results for the validation set predictions. The score value underlines that the chosen model is able to learn from recordings having the same containers, yet in different configurations, e.g., lighting conditions or filling. Also in this case, the class having the lowest score is *cup*.

Fig. 3.11 (top) shows the per-container type scores colored based on methods for the CCM public and private testing sets. Fig. 3.11 (bottom) provides a complementary analysis through the per-class mean of relative absolute error ε of the methods. Also in this case, the mean absolute error and the score have opposite behavior, qualitatively a high score is caused by a low error and vice-versa a high error causes a low score. In general, deep learning methods MobileNetV2 with Coordinate Attention [119] (M1), custom neural network [72] (M2), and ours (M3), obtain the highest score *s* for the class box in the public testing set, while cup in the private testing set. These models have the lowest scores in the class box in private testing set, and cup in the public one. M2 has lower results compared to M1 and M3 in the class box and cup (approximately



(a) Public testing set

(b) Private testing set

Fig. 3.11 Comparison of models performance per container type on the two CCM testing sets: (a) public and (b) private. Top: testing score s in percentage. Bottom: mean of relative absolute error ε. The maximum y-axis value of the score s and of the mean ε is set to 80 and 10 respectively for visualization purpose. In the public testing set, the error of the random method is 21.43 and the error of the average method is 12.25 for the class *cup*. In the private testing set, for the class *cup* the error of the random method is 17.45. Legend: — *M1*: MobileNetV2 with Coordinate Attention [119], — *M2*: custom neural network [72], — *M3 (Ours)*, — *M4*: random sampling, — *M5*: average mass.

20 percentage points). M1 has the best generalisation capabilities to different container instances. For both the public and private testing sets, random (M4) and average (M5) provide higher score values than other methods and lower errors in the class glass. The motivation behind this might be that mass values of glass instances are similar in the training and testing sets. For the other classes, M4 and M5 have lowest scores (under 20%) and highest mean relative absolute errors (over 8). In the public testing set, our method (M3) outperforms the others in the *cup* class, and is second best in the *box* class. In the private testing set, our method is second best in all classes.

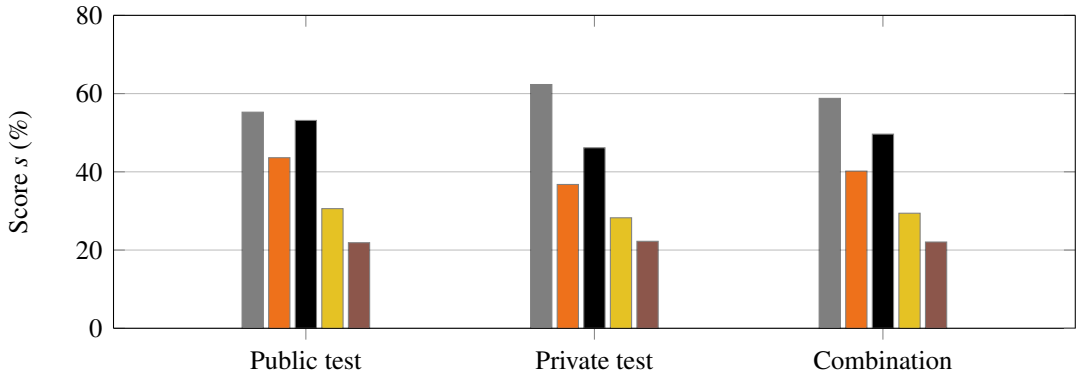


Fig. 3.12 Public, private, and combined testing scores *s* of container mass estimation solutions in percentage. Legend: — *M1*: MobileNetV2 with Coordinate Attention [119], — *M2*: custom neural network [72], — *M3 (Ours)*, — *M4*: random sampling, — *M5*: average mass.

The overall performance score of methods evaluated on private and public testing set are shown in Fig. 3.12. Our method (*M3*) achieves a higher score with respect to the custom neural network (*M2*), the random sampling (*M4*) and average mass (*M5*). Contrary to *M2*, our method does not use LoDE during the container detection phase and exploits only the fixed frontal view. The generalization properties on private testing set show that *M1* performs better than other models, probably due to the employment attention mechanisms and the adopted augmentation technique. Compared to this solution, our model is not pre-trained on other container properties estimation and features less parameters than MobileNetV2 model. The difficulty of the container mass estimation regardless of the content is highlighted by the fact that the combination of scores is under 60% for each model.

3.3 Summary

This chapter described our proposed method to analyze an RGB-D video, detect the container subject to manipulation based on its distance with respect to the fixed frontal view, and estimate its mass regardless of the content. Our method can be combined together with available models for the filling content mass estimation to obtain the complete mass of the container. The low percentage of pitcher patches in the training

set suggests that the proposed method is able to locate the manipulated container in CORSMAL Containers Manipulation (CCM) training recordings. In the experiments, the model learns from similar containers, but due to the variability in appearance of the different container instances gives a high estimation error with unseen containers, especially *cups*. Results on testing sets with never seen object instances show that our method outperforms most baselines; at the same time, the fact that values of scores are on average lower than 60% highlights the difficulty of the task. Although our method shows the second best results, it presents some limitations that we discuss in the following proposing also potential improvements. The method works frame-by-frame not considering the time dimension, and the patches selection phase is based on the assumption that the human offers the container towards the camera, hence the manipulated container will be the closest object to the fixed-frontal camera. Instead of computing the average depth values and select the K-nearest candidates based on the average depth, the manipulated container could be selected by analysing the human activity using the human pose and the time dimension [1, 74]. In this case, the model would predict the activity performed e.g., 'filling container' and 'offering container' to combine the information of the detected objects with the information of the arm offering the container. This is a potential improvement based on existing works in similar contexts, hence the performance is unknown. Considering the method as it is described in the chapter, during the container selection phase, the method extracts the average depth (d) by averaging the cropped depth map of the object in the object mask pixels predicted by the instance segmentation model. However the depth information might be incomplete or corrupted in case of transparencies, hence affecting the performance of the selection and also the subsequent mass estimation phase. To obtain a complete depth map and improve the average depth computation, techniques for depth estimation or completion could be used [58, 104, 136]. The generalisation of the mass estimation model to other object categories (even not containers) should be studied more in depth to understand its reliability. However, this might require to collect new data since the number of datasets for object mass estimation is limited, as discussed in Sec. 2.5.

Chapter 4

Affordance Segmentation of hand-occluded objects

In this chapter, we describe the phases of object localisation and affordance segmentation of the framework (highlighted in Fig. 4.1). We introduce the transfer learning procedure (fine-tuning) of lightweight models to detect the objects regardless of their category (objectness) targeting resource-constrained applications such as wearable robotics (Sec. 4.1); we describe the proposed affordance segmentation model targeting hand-occluded containers along with the annotation of the mixed-reality dataset used to train models (Sec. 4.2); we analyse the fine-tuning performance of the object detectors and their impact when used before lightweight affordance segmentation models, finally we compare our affordance segmentation model with previous methods from the literature (Sec 4.3); and we draw conclusions in Sec. 4.4.

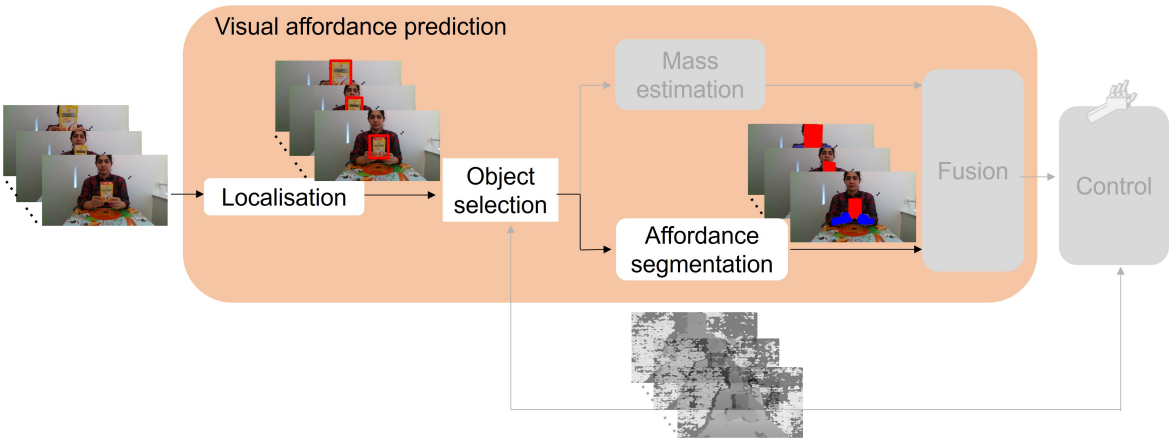


Fig. 4.1 Overall framework considering localisation and affordance segmentation phases. The object detector identifies objects in the RGB image and extracts the patches (image crops). Then, the affordance segmentation model identifies the *graspable* regions separating them from the *non-graspable* ones. The other phases of the framework (mass estimation, fusion, and control) are not considered.

4.1 Object detection for affordance segmentation

In semi-autonomous applications e.g., wearable robotic, the human is responsible for the coarse movements of the robotic hand in the environment, while some fine movements are controlled autonomously e.g., the finger movement [7, 116]. The active role of a person helps simplifying the problem, for example by offloading some control decisions, or leveraging human perception to retrieve useful information about objects in the environment [24]. Moreover, the person is supposed to act in the scene, trying to reach objects of interest to interact with them, hence (indirectly) moving the camera mounted on the robotic hand toward the object of interest.

In wearable robotic applications, the computer vision processing can be performed on board for example using dedicated hardware [83], to avoid the latency bottleneck of sending the data to an external computing unit and receiving the processed information. Due to costs and power consumption issues, resource-constrained devices are used, impacting the trade-off between accuracy and computational load of computer vision models. Table 4.1 shows hardware constraints of embedded systems that could be employed in wearable robotic applications [22, 28, 83], in terms of volatile memory

(RAM), non-volatile memory (Flash), and power consumption magnitude. The first four embedded systems are embedded Graphical Processing Units (GPUs) and their computational resources are orders of magnitude larger than the last two embedded systems (micro-controllers) impacting their consumption. The adoption of lightweight models in wearable applications becomes mandatory, because most models in the literature focus on the accuracy and not the computational cost, hence they have too many parameters to be stored or may require more memory than the one available to run [10, 24, 82, 130].

Table 4.1 Embedded systems hardware specifics representing plausible hardware constraints of wearable applications. Values are taken from the electronic systems website.

Manufacturer	**Model**	**RAM**	**Flash**	**Consumption**
NVIDIA	Jetson TX2	4 GB	16 GB + SD	W
NVIDIA	Jetson Nano	4 GB	SD	W
Intel	Movidius NCS	4 GB	-	$\approx$ W
Google	Coral	1-4 GB	8 GB	$\approx$ W
Greenwaves Tech.	GAP	8MB	64 MB	mW
ST	Stm32F7	512 kB	2 MB	mW

KEY – B: Byte, SD: external storage, -: not available, W: Watt, $\approx$: approximately.

Lightweight affordance segmentation models assume the object to be central and in foreground (framing issue) [90]. In general, this assumption cannot be guaranteed in the targeted scenario, hence we use a detection stage before the affordance segmentation. Our overall framework introduced in Sec. 1.1 (see Fig. 4.1) locates objects of interest in the scene using an object detector, then segments their affordances using an affordance segmentation model. The framework allows to consider localisation and affordance segmentation separately, decreasing training convergence problems in case of lightweight models. The employment of a single lightweight model to tackle the detection of objects and affordance segmentation could lead to unsatisfactory training

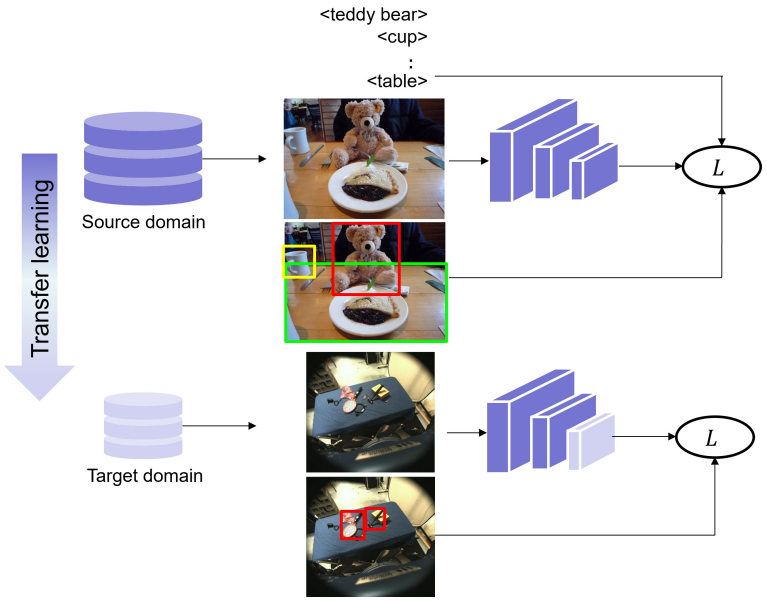


Fig. 4.2 Objectness fine-tuning block diagram. A model is pre-trained on a large dataset (source domain) for object detection, then the parameters are updated to match the target domain and distinguish between background and object minimising the loss function L. The RGB images are from [66, 82].

minimum. Additionally, our method provides modularity: each model can be replaced by a different architecture or an improved version (faster, more accurate or lighter).

4.1.1 Objectness fine-tuning

Since the human has an active role in the semi-autonomous applications and knows the category of the target object, we propose to learn the object localisation regardless of the class via fine-tuning (see Fig. 4.2). This process of separating background from objects is called 'objectness' detection [17]. Although the objectness detection was previously proposed in the affordance segmentation context, the focus was the accuracy, and not the trade-off between accuracy and computational load, important in resource-constrained scenarios. Moreover, the model performing objectness was using RG-D input and a Mask R-CNN approach [17], while we use RGB input and lightweight single shot detection. Given a lightweight object detection model we initialise the weights using the values pre-trained on COCO dataset [66] and fine-tune them on IIT-AFF [82] and UMD [80] obtaining two versions of the same model, one per target domain. We used the same training loss of the pre-trained configuration:

$$L = L_l + L_f, \tag{4.1}$$

where L_l is the localisation loss and the L_f is the classification loss.

The localisation loss L_l is a smooth L1 loss, and compared to the usual L1 loss reaches 0 more smoothly [96]:

$$L_l = \begin{cases} \frac{1}{2}(\psi - \hat{\psi})^2 & |\psi - \hat{\psi}| \leq \eta \\ \eta(|\psi - \hat{\psi}| - \frac{1}{2}\eta) & otherwise \end{cases}, \tag{4.2}$$

where ψ is the annotation of the bounding boxes, $\hat{\psi}$ is the predicted bounding box η controls the intersection between the two curves, and is set to 1. We simplify the notation writing ψ instead of the components of the bounding box vector.

The classification loss is a weighted sigmoid focal loss [65] that down-weights well classified examples and focuses on the hard examples. Given the predicted probability p of a class:

$$p_t = \begin{cases} p & \psi = 1 \\ 1 - p & otherwise \end{cases}, \tag{4.3}$$

$$\alpha_t = \begin{cases} \alpha & \psi = 1 \\ 1 - \alpha & otherwise \end{cases}, \tag{4.4}$$

$$L_f = -\alpha_t (1 - p_t)^{\gamma} log(p_t) \ , \tag{4.5}$$

where γ modulates the term multiplying the log value such that the easy examples (p_t close to 1) are down-weighted more than the hard ones (p_t close to 0), reducing their impact on the loss function, while α scales the values based on the annotation value. We set the number of object categories to 1 (objectness), $\alpha = 0.75$, and $\gamma = 2.0$.

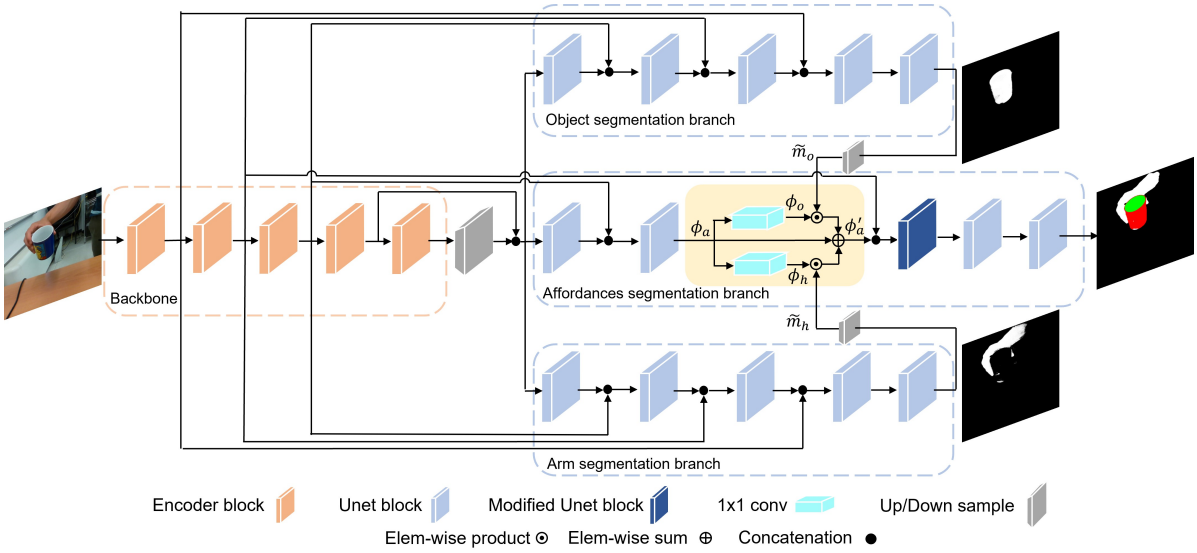


Fig. 4.3 ACANet, our proposed model for Arm-Container Affordance segmentation of hand-held containers. The object segmentation branch predicts the object mask, the arm segmentation branch predicts the arm mask, and the main branch fuses the predicted arm and object masks with the feature maps (ϕ_a) to predict the arm and affordances mask. The fusion block is highlighted in yellow. Figure from [C3].

4.2 The Arm-Container Affordance Network

In this section, we present our multi-branch architecture and the fusion module to perform affordance segmentation (see Fig. 4.3). We name the model Arm-Container Affordance Network (ACANet) as we consider containers for food and drinks, and *graspable* and *contain* as affordance classes. Assuming the input image has an object of interest, the model identifies the classes *graspable*, *contain*, *arm*, and *background*. The assumption about the object presence could impact the performance of the model in case the container is not in the image, predicting some affordance regions instead of *background*.

4.2.1 Multi-branch architecture

We devise our multi-branch architecture starting from a UNet-like architecture [100] that uses skip connections between the encoder and the decoder to preserve the information and help the gradient flow during back-propagation. We replace the encoder with a ResNet-18 [47] to include residual connections within the convolutional layers

of the encoder, easing the optimization problem [47]. Moreover, ResNet keeps the first pooling layer and replaces the others using a stride 2 in convolutional layers, unlike the original UNet encoder that halves the resolution of tensors using only pooling layers. To double the resolution in the decoders, we replace the UNet convolutional layers with non-trainable up-sampling layers based on nearest interpolation. For each decoder, we also modified the last convolutional layer from 1×1 kernel and no padding (original UNet) to 3×3 kernel, stride 1, and padding. This allows the last convolution to consider neighboring pixels information.

This architecture outputs a tensor $S \in [0,1]^{W \times H \times C}$ where each channel predicts one of the classes independently, and each pixel in a channel map is a probability such that $\sum_{c=0}^{C-1} S_c(i,j) = 1$, with $i \in \{1,..,W\}$ and $j \in \{1,...,H\}$ being the width and height indices, respectively. Including the class *arm* in S already handles the affordance segmentation of hand-occluded containers. However, we experimentally observed that the model first learns the classes with higher number of pixels in the annotation, affecting the prediction accuracy of the other classes.

We include two additional decoder branches that specialise in the segmentation of the arm and of the visible region of the object. Segmenting the object helps the model learn the area of the image where the affordances are. For simplicity, we refer to the three decoder branches as Arm, Object, and Affordance segmentation. The *Arm segmentation* branch predicts a probability map, $m_h \in [0,1]^{W \times H}$, that separates the region associated to the arm (composition of forearm and hand) from all the rest. The *Object segmentation* branch predicts a probability map, $m_o \in [0,1]^{W \times H}$, that separates the region associated to the visible object (held by the person hand) from all the rest. The *Affordance segmentation* branch fuses the feature maps with the arm and object maps, and predicts the segmentation tensor S.

4.2.2 Feature separation and fusion

Object and arm segmentation alone are insufficient to improve the segmentation accuracy under hand-occlusions. We therefore design a module that merges intermediate feature maps $\phi_a \in \mathbb{R}^{C' \times W' \times H'}$, with the down-sampled masks extracted by

the *Arm segmentation* branch ($\tilde{m}_h \in [0,1]^{W' \times H'}$) and the *Object segmentation* branch ($\tilde{m}_o \in [0,1]^{W' \times H'}$), respectively. We compute the intermediate feature maps within the *Affordance segmentation* branch by using two UNet blocks that process the feature maps outputted by the backbone. The object and arm masks are down-sampled using bi-linear interpolation to match the size of the intermediate feature maps.

Instead of directly combining the features maps with the segmentation masks, we first learn specialised features in the object and arm regions. Specifically, we convolve ϕ_a with a set of 1×1 filters to obtain the feature map related to the object, $\phi_o \in \mathbb{R}^{C' \times W' \times H'}$ (where C' is the number of filters), and with another set of 1×1 filters to obtain the feature map related to the hand, $\phi_h \in \mathbb{R}^{C' \times W' \times H'}$. We then perform a pixel-wise weighting of the feature maps ϕ_o and ϕ_h with the corresponding segmentation mask $\tilde{m}_o$ and $\tilde{m}_h$. This highly penalises the features outside of the predicted object (or arm) region. The merged feature maps are aggregated with the initial intermediate features as

$$\phi'_a = \phi_a + (\phi_h \odot \tilde{m}_h) + (\phi_o \odot \tilde{m}_o), \tag{4.6}$$

where $\odot$ is the Hadamard product, i.e., the element-wise product between each feature map in ϕ_o, ϕ_h and the down-sampled segmentation masks $\tilde{m}_o$, $\tilde{m}_h$[7].

4.2.3 Predicting object affordances and the hand

The *Affordance segmentation* branch uses the fused feature maps, ϕ'_a, as input to three UNet blocks to predict the output segmentation tensor S. Note that the feature maps ϕ'_a are concatenated via skip connection with the corresponding intermediate feature maps in the encoder before the first UNet block (see Fig. 4.3). We modify the first UNet block after the fusion module to improve the processing of the feature maps. Specifically, we increase the number of output channels of the first convolutional filter from 64 to 128, and we add another convolutional layer to decrease the channels from 128 to 64. Furthermore, we avoid concatenating low-level information (e.g., edges) from the backbone to the last three UNet blocks via skip connections. This

[7]Mathematical simplification as $\tilde{m}_o$ and $\tilde{m}_h$ should be repeated for each channel of the feature maps ϕ_o and ϕ_h.

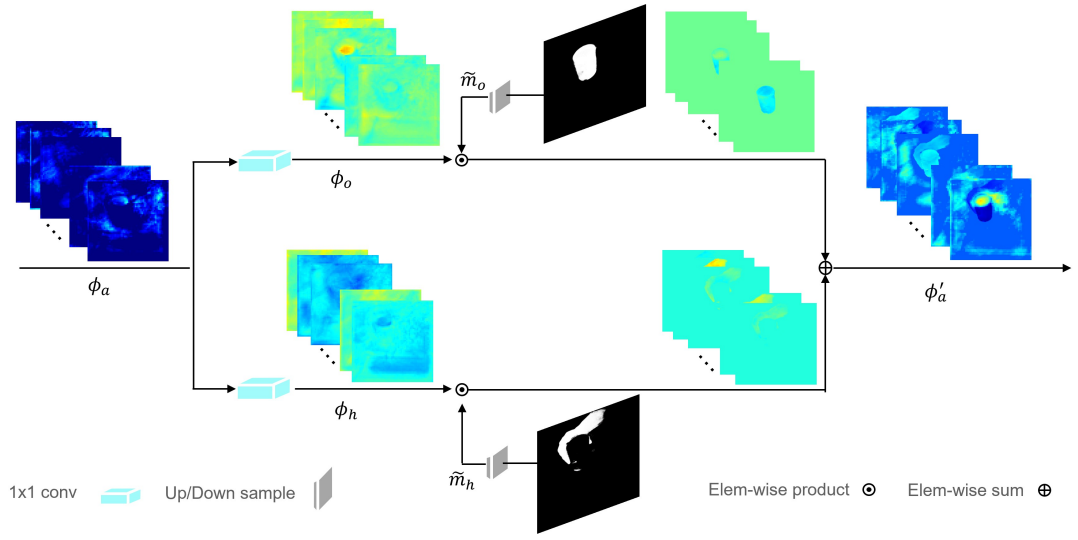


Fig. 4.4 Propagation of the features maps in the fusion module. The features ϕ_a are specialised into ϕ_o and ϕ_h and weighted using the object mask ($\tilde{m}_o$) and arm mask ($\tilde{m}_h$). Finally they are combined through the element wise sum. Feature maps are normalised and colored for visualisation purpose.

design choice helps the model preserve the changes in the feature maps ϕ'_a and predict affordances in the object region. The final segmentation map is: $S = \text{argmax}_c S$.

4.2.4 Loss functions

To train ACANet, we use a linear combination of a Dice loss [76] and two binary cross-entropy losses as:

$$L = L_a + \lambda_o L_o + \lambda_h L_h, \tag{4.7}$$

where the Dice loss, L_a, operates on the affordance branch outputs, the binary cross-entropy, L_o, operates on the object branch output, and the binary cross-entropy, L_h, operates on the arm branch output. The hyper-parameters λ_o and $\lambda_h \in \mathbb{R}$ control the impact of object and hand segmentation losses, respectively. L allows each branch to specialize for their segmentation task and influences the backbone to learn a common representation for all the branches.

The Dice loss, L_a, addresses the imbalance problem between classes, as the majority of pixels can be labelled as *background* [111]. Given a batch of B predicted

segmentation masks and corresponding annotations, the Dice loss is[8]:

$$L_a = 1 - \frac{1}{C}\sum_{c=0}^{C-1}\frac{2\sum_{i=1}^{B}\sum_{l=1}^{WH} y_{l,i}^{c}\hat{y}_{l,i}^{c}}{\delta + \sum_{i=1}^{B}\sum_{l=1}^{WH}\hat{y}_{l,i}^{c} + y_{l,i}^{c}}, \tag{4.8}$$

where $\hat{y} \in [0,1]^{WH\times C}$ and $y \in \{0,1\}^{WH\times C}$ are the reshaped predictions and annotations, respectively, with $\sum_{c=0}^{C-1} y_l^c = 1$.

The binary cross-entropy loss is used in binary classification and semantic segmentation tasks, considering each pixel as independent from the others. Given a batch of B predicted object segmentation masks and corresponding annotations, the binary cross-entropy loss for the object is:

$$L_o = -\frac{1}{B}\sum_{i=1}^{B}\sum_{l=1}^{WH} v_{l,i}\log(\hat{v}_{l,i}) + (1 - v_{l,i})\log(1 - \hat{v}_{l,i}), \tag{4.9}$$

where $\hat{v} \in [0,1]^{WH}$ is the reshaped vector of m_o and $v \in \{0,1\}^{WH}$ is the reshaped vector of the corresponding annotation. The binary cross-entropy for the hand, L_h, is similarly computed using the reshaped vector of m_h and corresponding reshaped annotation.

4.2.5 Mixed-reality affordance annotation

Training ACANet requires a large dataset with (exocentric) images of hand-occluded objects and segmentation annotation of both arm, object, and affordances. Such a dataset was not available, and collecting and manually annotating a new dataset is challenging, expensive, and time-consuming. We therefore complement an existing dataset, which has mixed-reality images of hand-occluded containers for object pose estimation [120], with visual affordance annotations. Using mixed-reality datasets can easily scale the generation of a larger number of images under different realistic backgrounds. Moreover, some existing works on hand-object reconstruction or object pose estimation achieved good performance when training on mixed-reality datasets despite the domain gap [45, 118].

[8]The margin $\varepsilon = 10^{-7}$ avoids numerical issues when $\hat{y} = y = 0$.

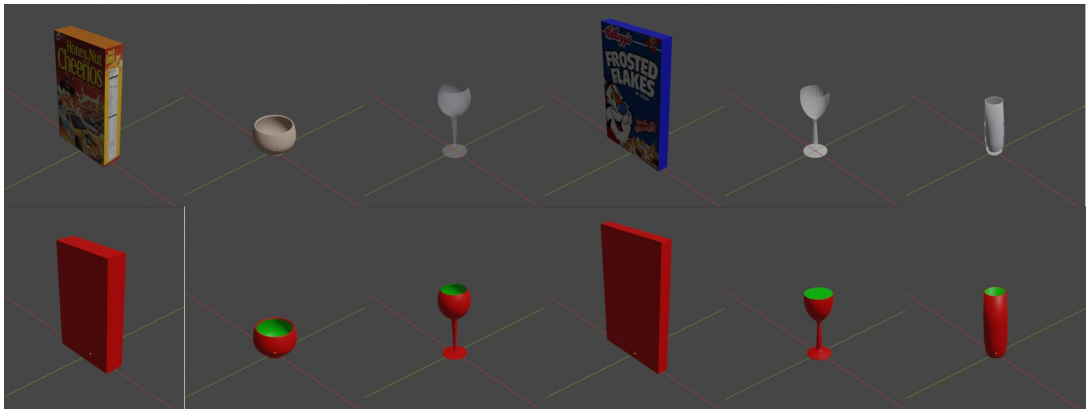


Fig. 4.5 Samples of annotated CAD models in Blender. In the first row the original CAD model with the texture, in the second row the affordance annotation on the surface of the objects. Key: — *graspable*, — *contain*

We use the publicly available CORSMAL Hand-Occluded Containers (CHOC) dataset that has 138,240 RGB images of 48 synthetic containers pseudo-realistically rendered on top of 30 different real backgrounds [120]. The dataset has 8,640 images with objects placed on top of a flat surface and 129,600 images of hand-held objects rendered in various locations and poses above the flat surface in the scene. The 48 containers are evenly distributed among 3 categories (box, stem, non-stem) and vary in their physical properties, such as size and shape, and appearance (textures, transparency). Hand-held objects were generated using synthetic forearms that hold the synthetic containers with three different visually plausible grasps and are orientated towards the pointing direction of the camera to simulate a potential offering of the object.

These characteristics and the available generation pipeline allows us to easily extend CHOC with the annotation of the *graspable* and *contain* affordances in addition to the existing annotations of the segmentation masks for the *arm* and *object*. We label the affordances on the surfaces of the 3D CAD models of the 48 containers using Blender [19]. In particular, we consider as *graspable* the external surface of boxes, stem, and non-stem objects, while the internal surface of stem and non-stem as *contain* (see some samples in Fig. 4.5). The internal surface of the boxes is not annotated because boxes are closed. We then use the object poses annotated in CHOC to project the CAD models with affordances on the image plane and render the affordance

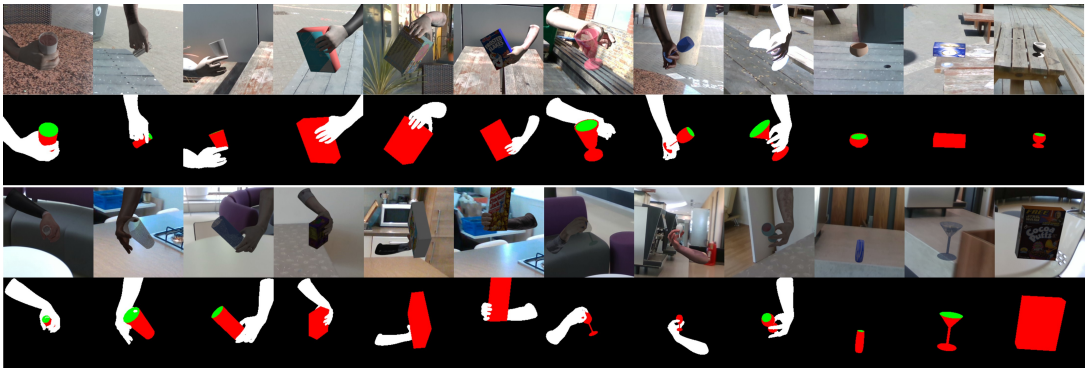


Fig. 4.6 Samples of cropped RGB images and segmentation maps of arms and object affordances from the annotated mixed-reality dataset, CHOC-AFF. Key: ▬ *background*, ▬ *graspable*, ▬ *contain*, ▭ *arm*

maps. When rendering the object mask back to the image plane to get the annotation, occlusions would not be considered, hence we perform a subtraction with the pixels belonging to the hand. The arm mask is already available in CHOC annotation, so in the affordance mask we replace the pixels of the object that are occluded by the hand with the *arm* class. For simplicity, we refer to this version of the dataset with annotations of the affordances as CHOC-AFF. Fig. 4.6 shows sampled images and annotations of CHOC-AFF varying the pose of the object and of the hand. We use CHOC-AFF for training and testing ACANet and other models, and we evaluate the generalisation of the models to real images.

4.3 Validation

4.3.1 Methods under comparison

For the objectness detection, we analyse the performance of two versions of MobileNetV3 object detector [51] trained with the proposed fine-tuning procedure, since these models are designed to target limited computational capabilities systems. Fig. 4.7 compares the main models for object detection and affordance segmentation through the number of parameters (Params), and the number of Floating Point Operations

(FLOPs) indicating how many calculations are performed during an inference phase. We chose these descriptors because they depend on the models, while other descriptors such as the inference time depends on the hardware used. Most of the affordance literature methods adapt powerful object detectors and instance segmentation models like R-FCNResNet [82], Faster R-CNN [10], Mask R-CNN [14, 17, 24, 129]. Due to the fact that some of the trained models for the affordance segmentation are not publicly available, we used the reported descriptors in the original publications as a proxy for the actual values. The employment of these models in a resource-constrained scenario is practically impossible due to the number of parameters and required computational power. On the contrary, MobileNetV3 models provide a suitable solution working with constrained systems. The gain in terms of parameters with respect to R-FCNResNet-101 is more than 77×, and more than 64× for FLOPs in case of Large version. These numbers are more than doubled considering Small version. It is evident the advantage in terms of memory footprint, since on-device memory store of the model depends on the number of parameters. To analyse the impact of object detection models on the affordance segmentation problem in case of resource-constrained scenario, we use lightweight affordance segmentation models [90] that use MobileNet feature extractor: MobileNetV1_UNET (V1U) and a customized version of small MobileNetV3_LRASPP (V3L). Fig. 4.7 shows that also in case of affordance segmentation models that the computational cost of lightweight models is significantly lower than the one of other models used in the literature like AffordanceNet [24, 17, 14], or DRNAtt [38]. The number of parameters of V3L has a reduction factor 32x compared to V1U, while 16x in terms of GFLOPs. This reflects in both memory footprint and inference speed.

In case the target is different from resource-constrained systems, more computational demanding models can be employed. We compare our proposed model to segment the arm and the affordances (ACANet) against a baseline and two state-of-the-art methods: ResNet18-UNet (RN18-U), ResNet50-FastFCN (RN50-F) [53], and DRNAtt [38]. RN50-F combines a ResNet-50 based encoder with a pyramid scene parsing module [138]. We leave the auxiliary loss of the pyramid scene parsing module

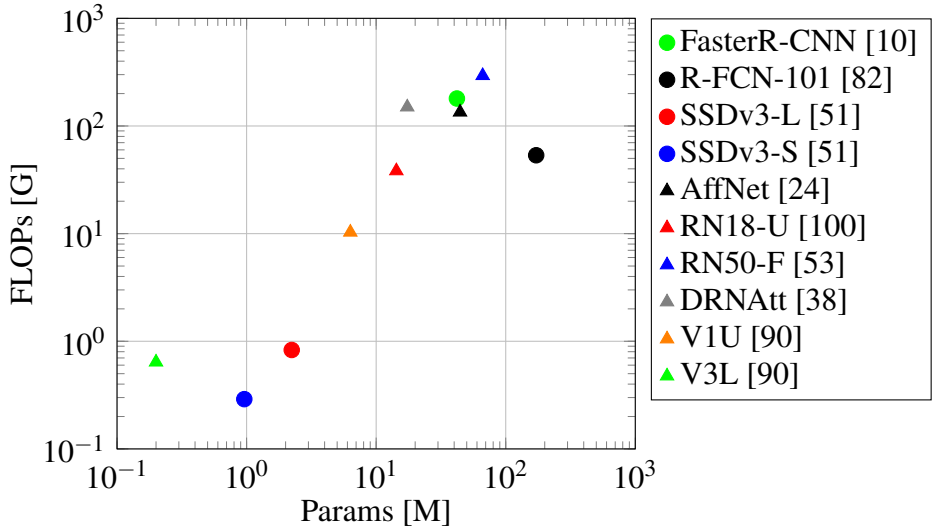


Fig. 4.7 Number of parameters in millions [M] and Floating Point Operations (FLOPS) in giga [G] of object detection and affordance segmentation models used in the literature. Legend: • object detection model, ▲ affordance segmentation model.

to propagate the gradients, setting the weight to 0.2 [122]. RN18-U is the single-branch baseline behind ACANet and has a UNet-like architecture with a ResNet-18 based encoder. To reduce the dimensionality in the backbone, the first pooling layer is kept, while the others are replaced with 1×1 convolutional layers with stride 2, to learn the compression. The upsampling layers perform nearest interpolation and the last layer is a 3×3 convolution with stride 1 and padding to maintain resolution. DRNAtt is the best performing model on the UMD dataset [11]. The model is composed mainly by three parts: the Dilated Residual Network (DRN) [132] for feature extraction, a combination of Spatial Attention Module and Channel Attention Module [30] for feature enhancement, and an upsampling layer based on pixel shuffling operation [108]. We implemented RN18-U and re-implemented DRNAtt to directly segment both the object affordances and the arm. We changed the channels of the last layer in RN50-F[9] to segment only affordances of objects as per the original implementation in case of egocentric data [53]. We perform also an ablation study to assess the contribution of the modified UNet layer and the missing skip connection in the performance of ACANet:

[9] Original RN50-F implementation [122] is available at https://github.com/wuhuikai/FastFCN

- *-UNet* is ACANet without the modified UNet layer. The UNet block is the same block of the other branches.
- *Skip* is ACANet with an additional skip connection after the second last UNet block to concatenate low-level information from the backbone.

4.3.2 Experimental setup

The proposed objectness fine-tuning is evaluated through three sets of experiments. The first one concerns the object detection and compares the mean Average Precision, *mAP* (discussed in Sec. 2.6), of models using all the object classes and the one-class configuration on UMD and IIT-AFF testing sets. Additionally, a complementary analysis is shown by collecting the percentage of images in which occurs at least one detection to analyse the detection capabilities. The second group of experiments has the purpose of linking object detection analysis to affordance detection application. Object detection patches are extracted from the test sets and the histograms of affordance class pixels are compared with the distribution obtained using ground truth bounding boxes. The motivation of this test is to assess whether the object detector is able to detect objects of interest in the scene. The focus is still on the object detection capabilities. The third group of tests compares the affordance detection capabilities of the pipeline with the baseline which consists in the employment of the sole affordance detector [90]. In this way, both solutions are considered in the same distance and framing conditions. The performance measure used to evaluate methods in the third group of experiments is F_β^w [71], that weights the prediction errors using the distance to the annotation (additional information in Sec. 2.6).

For our experiments on hand-occluded affordance segmentation, we split CHOC-AFF into *training* set and *validation* set to train the models, and two *testing* sets to evaluate the models generalisation to different backgrounds and different object instances. We also select and annotate images from two existing public datasets for hand-object pose estimation or reconstruction to evaluate the models in real conditions. For CHOC-AFF, the training set has $89,856$ images with 26 out of 30 backgrounds and 36 out of 48 containers (12 per object category). The validation set has $17,280$ images

with all 30 backgrounds and 6 container instances (2 per object category) different from the ones in training set. The first testing set has 13,824 images and evaluates the generalisation performance of the models to the same training object instances in 4 backgrounds not seen during training. The second testing set has 17,280 images and evaluates the generalisation performance of the models to 6 object instances (2 per object category) not seen during training in all 30 backgrounds [120]. For the two testing sets in real conditions, we consider HO-3D [42] and CCM [125] due to the presence of various challenges, such as presence of the human body, real interactions, and different object instances and hand-object poses. HO-3D is a multi-view video dataset of people manipulating different types of objects. We selected 150 frames of mugs and boxes as containers from the lateral and frontal cameras (with respect to the arm), keeping a diversity in object and hand poses. We used the segmentation provided by the authors as annotation for the classes *arm*[10] and *graspable*, whereas we manually annotated the class *contain*. CCM is a dataset of multi-view sequences of people manipulating containers with different contents, and then offering the objects to a fixed robot arm. The offering phase allows us to evaluate the models under more realistic human grasps and object poses in a human-robot collaboration scenario, and more challenging conditions caused by different background and lighting settings. Moreover, containers can vary in their physical appearance (e.g., transparency, texture) or be affected by the presence of content. We selected the last frame (offering phase) of 150 sequences from a side perspective, diversifying objects, hand poses, and scene color settings. We manually annotated the affordance classes *contain* and *graspable*, and the class *arm* of only the hand(s) in contact with the offered container.

4.3.3 Training details

In the training phase, UMD [80] one instance per each object is left out and reserved to the test set (5135 images), while the rest of images is splitted in 85% for training and 15% for validation. IIT-AFF [82] followed splits provided with the dataset (2651 images for test set). Both MobileNetV3 object detectors are fine-tuned starting from

[10]Note that the forearm is not annotated.

Table 4.2 Training details of object detection and affordance segmentation models.

Dataset	Steps[k]	Max epochs	Batch	Lr	Schedule
UMD	800	-	20	.4	Cosine decay
IIT-AFF	650	-			
CHOC-AFF	-	100	2	.001	Linear decay

COCO dataset [66] checkpoints[11]. We use the input resolution of 320×320 that is consistent with lightweight models. The training details are reported in Table 4.2. In particular, in case of UMD $800,000$ is the number of total steps, batch size is 20, the learning rate parameters are left as default: cosine decay base is 0.4, the warmup value is 0.13333 and $2,000$ warmup steps. The training hyperparameters on IIT-AFF dataset are the same of UMD with the exception of the number of steps that is set to $650,000$. The difference in number of steps between the two dataset is to avoid overfitting, since the number of available images in IIT-AFF is lower than in UMD. To avoid the distorsion of input images, especially in IIT-AFF case in which there are different resolutions, images are padded with zeros to maintain the aspect ratios while reducing width and height to fit models' input. Additionally, we perform random horizontal flip and random crop augmentation procedures to reduce the overfitting phenomenon.

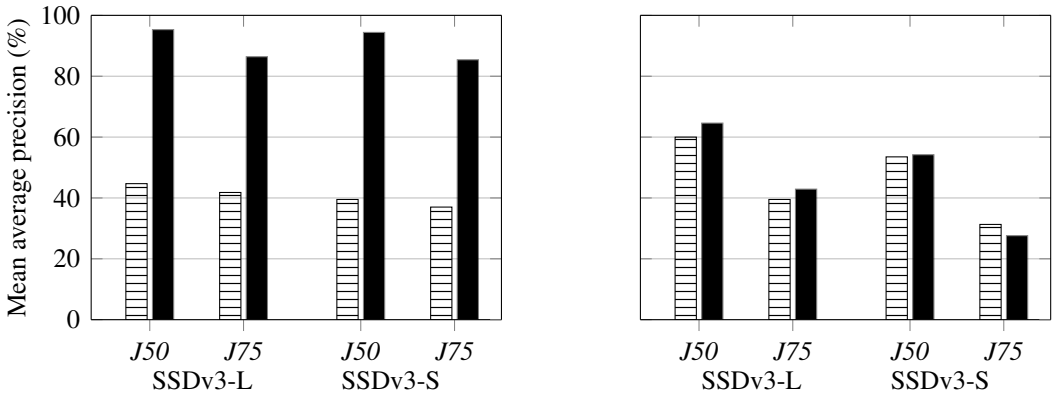
During training, we fine-tune all encoders, pre-trained on ImageNet [102], to start from a better initialisation than random. For ACANet, we set the hyper-parameters λ_o and λ_h to 1. For all models, we set the batch size to 2, the initial learning rate to 0.001, and we use the mini-batch Gradient Descent algorithm as optimizer with a momentum of 0.9 and a weight decay of 0.0001. For all models except RN50-F, we schedule the learning rate to decrease by a factor of 0.5, if there is no increase of the mean Intersection over Union in the validation set for 3 consecutive epochs. For RN50-F, we set the learning rate schedule following the original setup. We use a Dice Loss to penalise the errors for RN18-U. We use the standard cross-entropy loss for DRNAtt and RN50-F (with auxiliary weight set to 0.2). We use early stopping with a patience of 10 epochs to reduce overfitting, and set the maximum number

[11] https://github.com/tensorflow/models/tree/master/research/object_detection

of epochs to 100. Images can be of different resolutions and therefore we apply a cropping square window of fixed size to avoid distorsions or adding padding. To train and test the affordance segmentation phase of the framework, decoupling it from the object detection, we crop a $W \times W$ window around the center of the bounding box obtained from the object mask annotation to restrict the visual field and obtain an object centric view. This cropping procedure is equivalent to assume a perfect object detector. However, the cropping window can go out of the support of the image if the bounding box is close to the image border. In this case, we extend the side of the window that is inside the image support to avoid padding. In case the bounding box is bigger than the cropping window, we crop the image inside the bounding box and resize it to the window size. We apply this cropping procedure to all the images both in training and testing phases. During training, we also use data augmentation to further increase the diversity of the images. Specifically, for each input image, we apply the following sequence of transformations: resize by a factor randomly sampled in the interval $[1, 1.5]$ to avoid degrading quality; center crop the resized image with a $W \times H$ window to restore the original image resolution; and horizontal flip with a probability of 0.5 to simulate the other arm. We set the window size to $W = H = 480$ (minimum resolution for RN50-F), as higher resolutions degrade the image quality.

4.3.4 Results and discussion

Fig. 4.8 compares the test set Mean average precision (*mAP*) of MobileNetV3 object detectors Small (*SSDv3-S*) and Large (*SSDv3-L*) on UMD and IIT-AFF testing sets. Models are tested using both setups: all classes or objectness (one class). We chose two thresholds of Jaccard index, or Intersection over union, (*J50* and *J75*) to analyse how the performance varies. It can be expected that the higher the threshold the lower the mean average precision because with higher threshold a lower number of predicted boxes will overlap and have comparable size compared to the annotation. Focusing only on object localisation improves significantly detection score in case of UMD testing set. The *mAP* increases of more than 2 times for both model using $J50$ and $J75$. Using the proposed objectness fine-tuning has different results on IIT-AFF testing



(a) UMD (b) IIT-AFF

Fig. 4.8 Mean Average Precision on affordance testing sets: (a) UMD and (b) IIT-AFF. SSDv3-L and SSDv3-S indicate two versions of MobileNetV3 object detectors, *J50* and *J75* denote the selected Jaccard index threshold. Legend: ▭ all classes, ▬ objectness (one class).

set. The improvement in IIT-AFF test set is almost negligible in case of *SSDv3-S*, while it is more substantial in case of *SSDv3-L*. IIT-AFF results are affected by the partial clutter and occlusion conditions which characterise part of the images. The perfect operating condition is offered by the clean framing of UMD dataset. In case of wearable application, this condition could be obtained through a smarter camera placing.

An additional way to measure the improvement of the detection capabilities of the object detectors is to analyse in how many images there is at least one detection, since there is at least one object in each image. The percentage of images in which there is no detection is the complementary one. Fig. 4.9 shows the percentages of UMD and IIT-AFF testing set images in which the object detector predicts at least one object. For both SSDv3-L and SSDv3-S, the percentages increase passing from all classes (*ALL*) to objectness (*ONE*) configuration. Consequently, the number of images in which there is no detection decreases passing from all classes (*ALL*) to objectness (*ONE*) configuration. The improvement is more evident in case of UMD testing set due to the lower complexity of this dataset. The obtained results indicate that using the proposed objectness fine-tuning leads to better object detection performance in the targeted scenario.

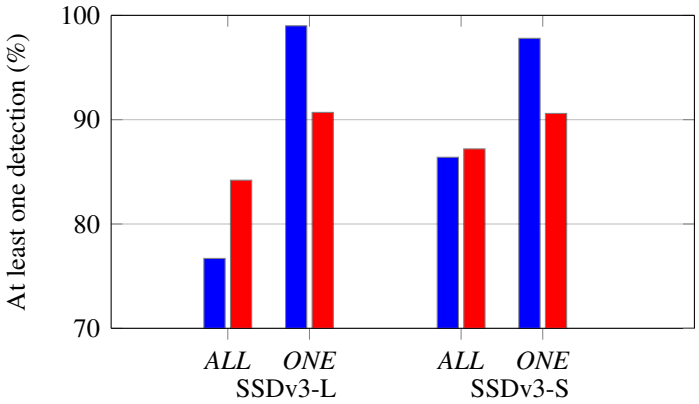
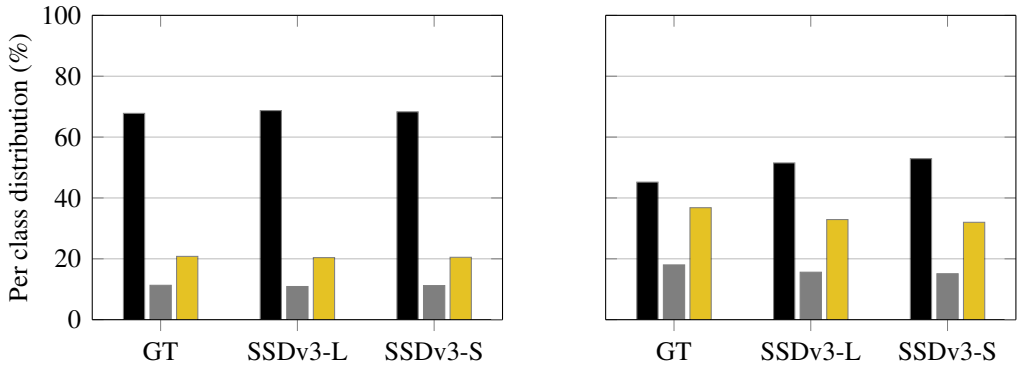


Fig. 4.9 Percentage of images with at least one detection. *ALL* and *ONE* indicate the detection configuration, SSDv3-L and SSDv3-S the employed MobileNetV3 version. Legend: — *UMD*, — *IIT-AFF*.

The previous set of experiments focused only on assessing the detection capabilities of the pipeline using the proposed objectness fine-tuning. To understand if the objectness detector locates objects useful for the affordance, we compare the affordance classes distribution in ground truth bounding boxes with the distribution affordance classes predicted by the whole pipeline. Fig. 4.10 shows the percentages of patches pixels belonging to *graspable*, *non-graspable*, and to *background* in UMD test set, Fig. 4.10 (a), and IIT-AFF test set, Fig. 4.10 (b). *GT* denotes the distribution of pixels in the ground truth patches, SSDv3-L and SSDv3-S the distributions of patches extracted by the two object detectors with the objectness fine-tuning. It is possible to note that the distribution of classes in affordance segmentation problem is highly imbalanced, even after grouping all *non-graspable* classes into one and considering only object crops. The distribution remains almost the same in UMD case, confirming that detected patches represent objects of interest (see Fig. 4.10(a)), while it slightly changes in IIT-AFF test set, due to the more challenging characteristics of the dataset e.g. presence of more than one object, and the fact that not all the instances present in the images are annotated.

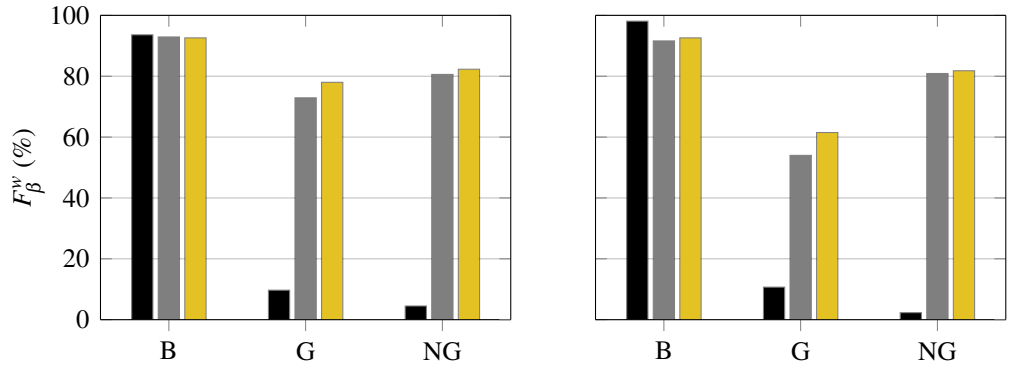
Fig. 4.11 shows the results of the last experiment to validate the objectness fine-tuning in which we compare the per-class F_β^w score obtained for both IIT-AFF and UMD testing sets. We consider the affordance classes *graspable* (G), *non-graspable*



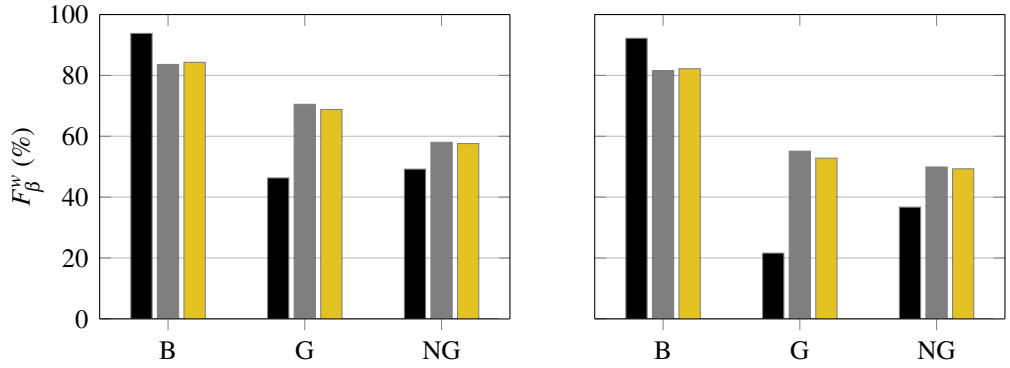
(a) UMD (b) IIT-AFF

Fig. 4.10 Per class distribution of object detection pixels belonging to affordance classes in testing sets: (a) UMD, (b) IIT-AFF. SSDv3-L and SSDv3-S denote the distributions originated by the patches extracted using the two architectures of MobileNetV3-based object detectors, while GT the ground patches present in the dataset. Legend: — *background*, — *graspable*, — *non-graspable*.

(NG), and the class *background* (B). As a baseline, we consider the case in which the first part of the pipeline, the objectness detector, is missing and only the lightweight affordance segmentation models [90] are used. Both object detector SSDv3-L and SSDv3-S with objectness fine-tuning are tested, as well as both lightweight affordance segmentation models MobileNetV1-UNET (V1U) and MobileNetV3 LR-ASPP (V3L). The proposed solution outperforms the baseline on both testing sets in the affordance classes. The high score in the background class indicates that affordance models are more incline to predict the background, due to the imbalance distribution of affordance classes. In UMD testing set the improvement in F_β^w introduced by the objectness detectors is more than 6 times compared to the baseline. The pipeline employing SSDv3-S obtains a higher score compared to the one using SSDv3-L, which is in contrast with mAP values in Fig 4.8. In IIT-AFF testing set the proposed pipeline improves the affordance segmentation score compared to the baseline for *graspable* and *non-graspable* classes. The fact that the improvement has a lower magnitude compared to the UMD testing set is due to the characteristics of IIT-AFF. Some images contain objects in foreground, hence the baseline is sufficient to predict the affordance mask. Another phenomenon which influences IIT-AFF results is the fact that in cluttered scenes also other objects may be present in the patches, affecting the



(a) UMD testing set

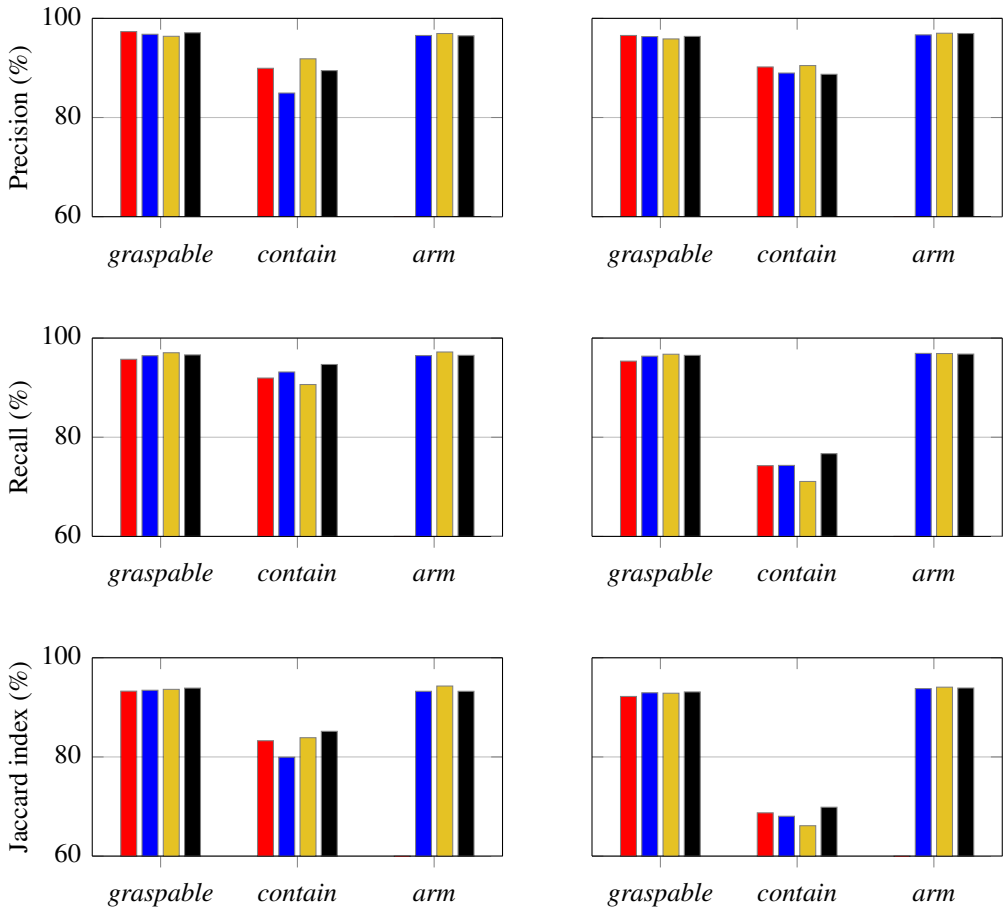


(b) IIT-AFF testing set

Fig. 4.11 F_β^w score on testing sets: (a) UMD, (b) IIT-AFF. SSDv3-L and SSDv3-S are the two versions of MobileNetV3-based object detectors. V1U is the affordance detector MobileNetV1-UNET and V3L is MobileNetV3 LR-ASPP. Segmentation classes are *background* (B), *graspable* (G) and *non-graspable* (NG). Legend: ▬ *No detector*, ▬ SSDv3-L, ▬ SSDv3-S.

score computation. The results of object detectors show that the objectness fine-tuning improves the detection performance of lightweight object detectors on UMD and IIT-AFF testing sets, and consequently the performance of affordance segmentation models in cascade.

Fig. 4.12 and Fig. 4.13 compare the performance of the affordance segmentation models on the mixed-reality and real testing sets. For the discussion and ranking of the methods, we consider Jaccard index J as the reference performance measure. For simplicity Recall and Precision will be referred to as R and P respectively. Overall, our method (ACANet) outperforms the other models on all datasets and for most of the



(a) CHOC-B (b) CHOC-I

Fig. 4.12 Comparison of the affordance and arm segmentation results between the models on the two mixed-reality testing sets: (a) CHOC-B, (b) CHOC-I. Legend: — *RN50-F*, — *RN18-U*, — *DRNAtt*, — *ACANet* (ours).

classes. This means that ACANet achieves better generalisation to other backgrounds and object instances in the testing sets.

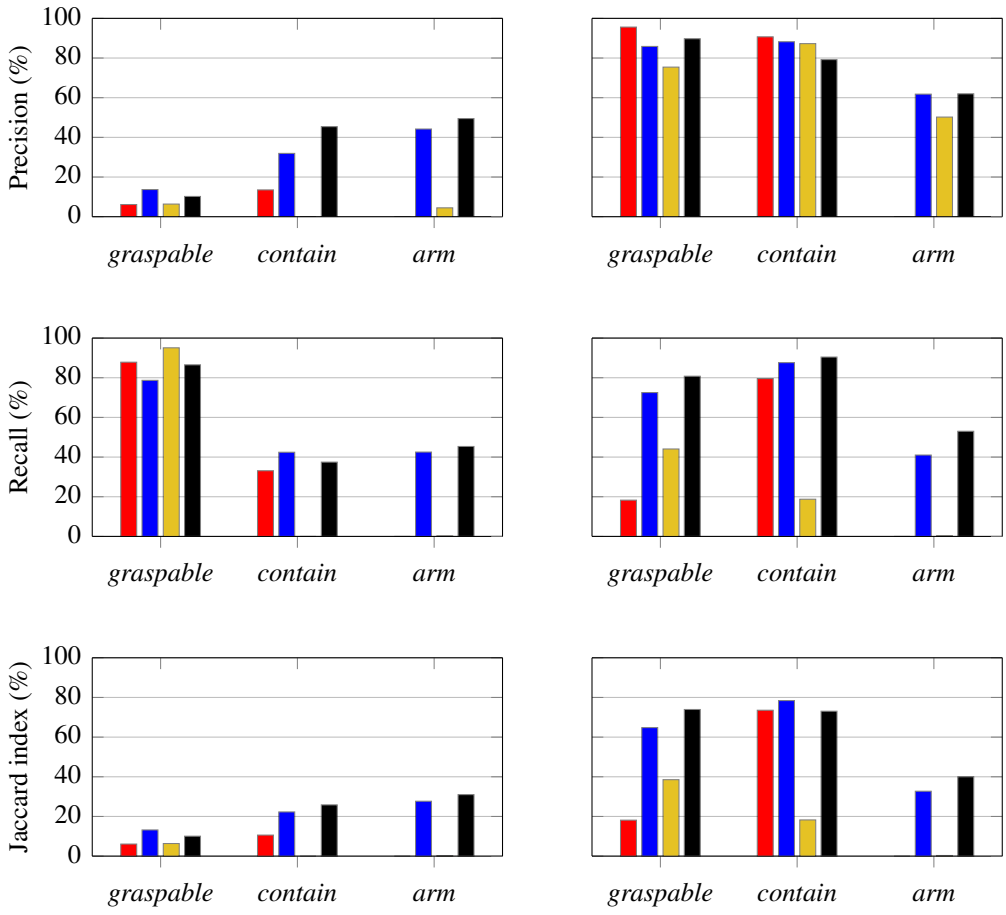
The performance on the mixed-reality testing sets (see Fig. 4.12) is similar among the models and J is higher than 90% for the classes *graspable* and *arm*. Models predict a low number of false positives and false negatives, resulting in a high value for precision and recall (R, $P > 95\%$). The class *contain* is the most challenging and J drops to the interval [79%, 85%] for the first testing set with unseen backgrounds (CHOC-B) and to the interval [66%, 70%] for the second testing set with unseen object instances (CHOC-I). ACANet outperforms the other models for the classes *graspable*

and *contain* on both testing sets, whereas DRNAtt has the highest Jaccard Index for the class *arm*. RN50-F has the lowest performance, except for J in *contain* and P in *graspable*. This result shows that adding the *arm* class helps improve the performance. In CHOC-B, models except DRNAtt predict more false positives than negatives for the class *contain* ($P \in [84\%, 91\%]$, $R > 90\%$). On the contrary, the number of false negatives for the class *contain* is higher than false positives ($R \in [71\%, 76\%]$, $P > 88\%$) in CHOC-I.

The HO-3D and CCM testing sets allow us to assess the generalisation capabilities of the models to images acquired in real scenarios (see Fig. 4.13), given the known problem of the gap between synthetic and real data. As expected, performance of the models is lower in the real testing sets than in the mixed reality testing sets due to the domain shift. This can be observed especially for the classes *graspable* and *arm*.

In HO-3D, ACANet outperforms the other models for the classes *graspable* and *arm*. However, all models tend to predict the wrong class in the *graspable* and *arm* regions ($P > R$), and even ACANet has a high number of false positives and false negatives for the class *arm* ($J = 40\%$). The performance for the class *arm* is penalised for all models due to the lack of annotation of the forearm, and the presence of the human body and challenging arm poses. For the class *contain*, DRNAtt predicts a high number of false negatives that affect the final performance ($J = 18.25\%$), whereas the other models predict a lower number of false positives than DRNAtt, resulting in a higher Jaccard index ($J \in [73.07\%, 78.42\%]$).

In CCM, the tablecloth and the presence of the human body are the main challenges for the models, causing a performance drop compared to the other datasets. In the presence of the tablecloth, models tend to predict *graspable* in most regions of the image. This results in a large difference between P and R (e.g., 76 percentage points for ACANet). ACANet achieves the best performance for the classes *contain* ($J = 25.83\%$) and *arm* ($J = 31\%$). DRNAtt does not generalise to the real images of CCM with $J \leq 1\%$ for the class *arm* and $J = 6.35\%$ for the class *graspable*, whereas the class *contain* is not predicted. This is caused by a large number of false positives towards the class *graspable* ($P = 6.37\%$) and a large number of false positives and false



(a) CCM (b) HO-3D

Fig. 4.13 Comparison of the affordance and arm segmentation results between the models on the two real testing sets: (a) CCM, (b) HO-3D. Legend: — *RN50-F*, — *RN18-U*, — *DRNAtt*, — *ACANet* (ours).

negatives for the class *arm* ($P = 4.47\%$, $R = 0.24\%$). The higher performance of ACANet compared to DRNAtt and RN18-U shows that learning arm and object features separately is better than learning affordances directly.

Fig. 4.14 and Fig. 4.15 show and compare the affordance segmentation predictions of the models on sample images from the mixed reality and real testing sets respectively. We chose images with objects and hand poses that are challenging and never seen in training, e.g., holding a box from the bottom; and with different backgrounds and lighting or different object appearances. Visually, ACANet achieves the most accurate segmentation for the arm and the object affordances. For CHOC-B (see Fig. 4.14), the

predictions have more false positives in the background because the object instances are seen during training, whereas there are false positives or false negatives in the object region for CHOC-I because the object instances change with respect to training. All models show a high number of false positives for the class *graspable* in the CCM testing set (see Fig. 4.15), especially when there is a colorful tablecloth (4th, 5th, 6th and 8th columns). In these cases, RN50-F and DRNAtt predict the widest regions of *graspable* false positives in the image, over the tablecloth, the human body, and also the wall behind the human. These results suggest that the learned features do not generalise to the CCM dataset setting. The false positive regions predicted by RN18-U and ACANet are mostly on the tablecloth and on the human body, but the *arm* and *contain* regions are correctly predicted on the human hand and the cup. Moreover, the predictions of RN18-U and ACANet are close to the annotation when the setting is similar to the training one, i.e., there is no colorful tablecloth nor the human body/face, but just the arm holding the container. We also chose to show the results for a transparent cup as transparency can be a challenge for the models due to the not clearly defined borders (2nd and 8th column). For example, the background environment or the content in the container can influence the segmentation predicted by the models. In the sampled image (2nd column), the cup is filled with a content, and the models are able to correctly predict both the *contain* and *graspable* regions, except for DRNAtt. For HO-3D, ACANet predictions show better affordance segmentation and more complete masks compared to other models, but the number of false positives of the class *arm* increases in presence of the human face (5th and 8th columns).

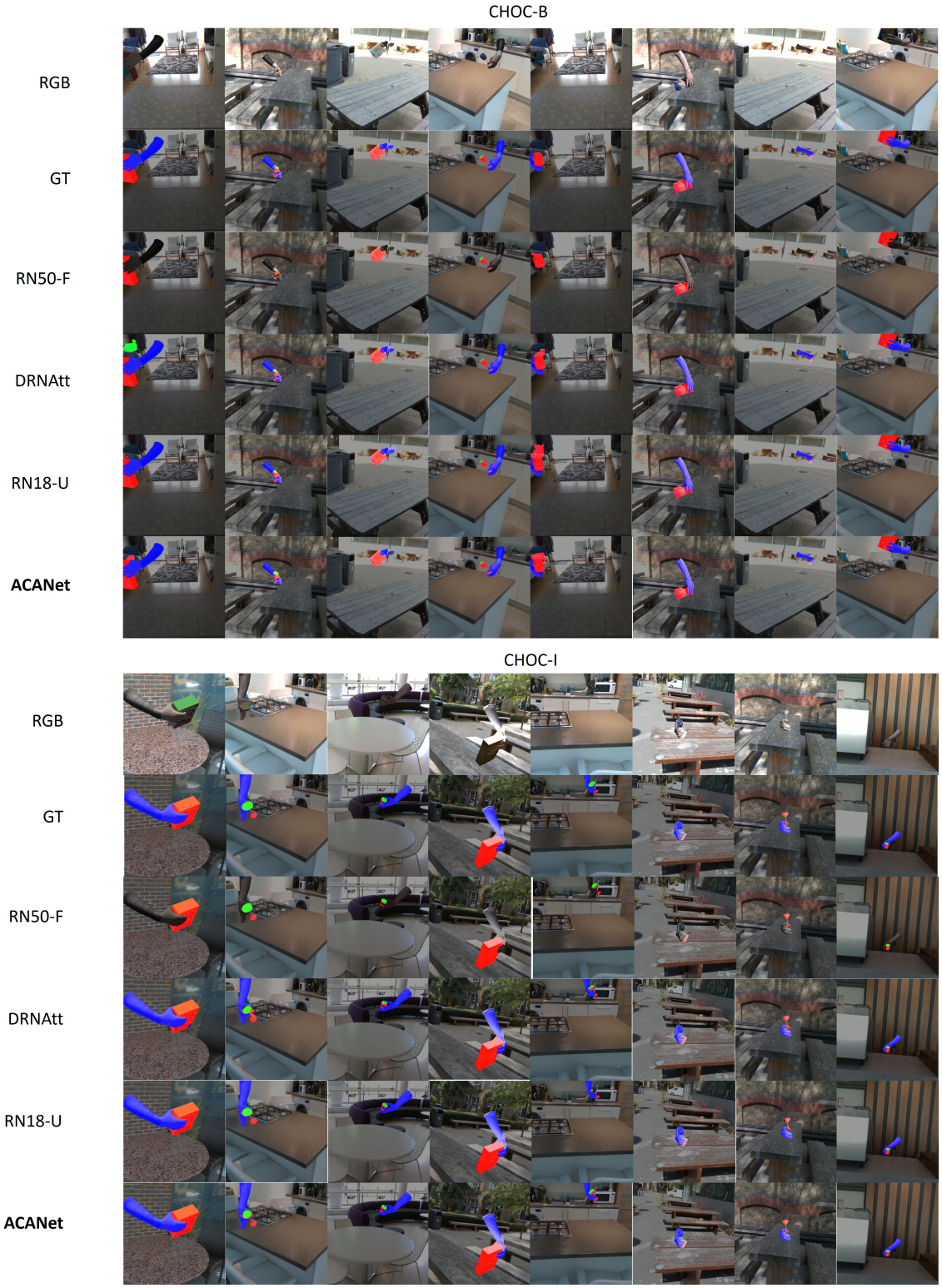


Fig. 4.14 Comparison of the predicted affordance and hand masks of the models on sampled images from the mixed-reality testing sets. The segmentation masks are overlayed on the RGB images. KEY - GT: ground-truth, — *graspable*, — *contain*, — *arm*.

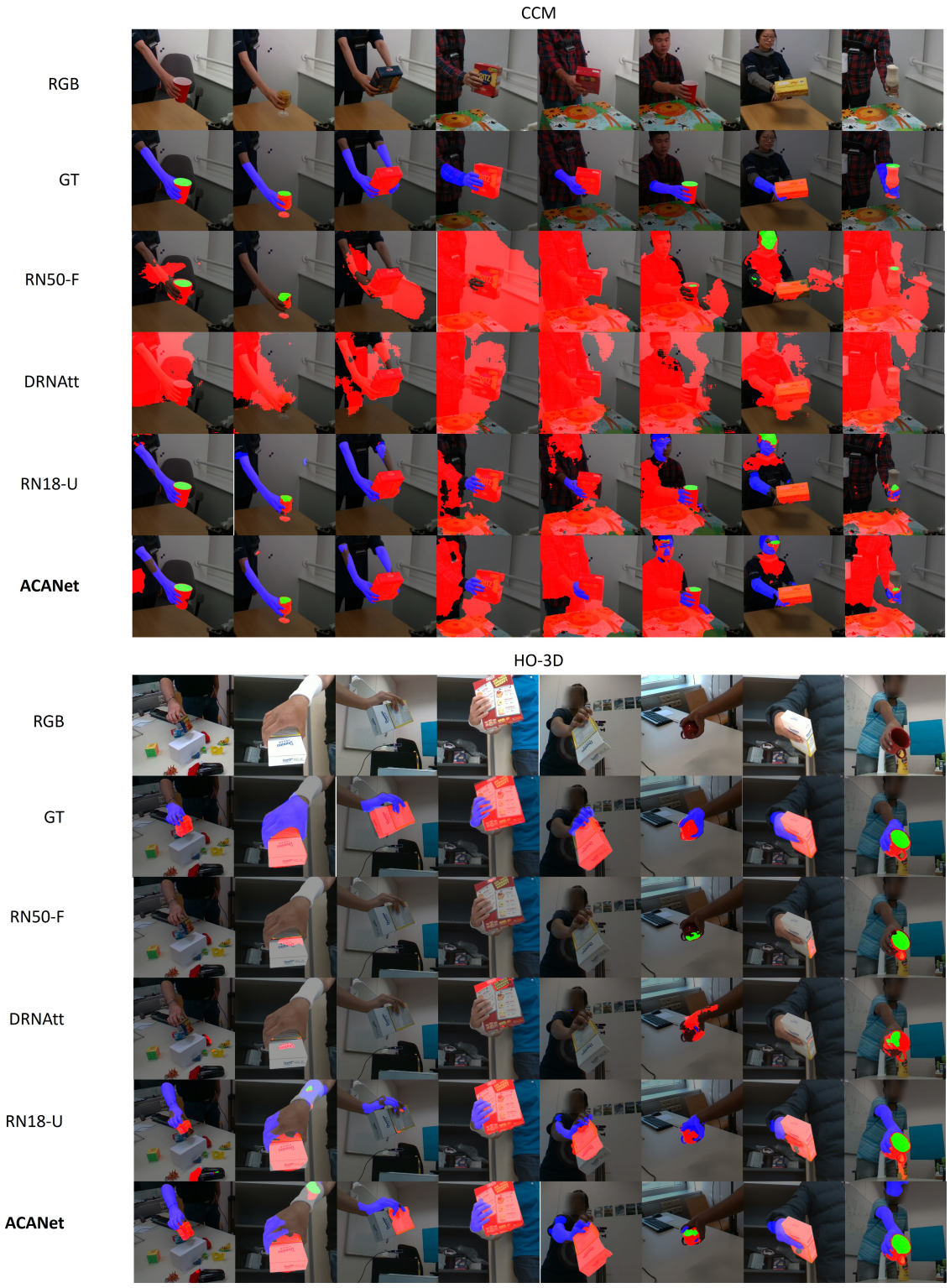


Fig. 4.15 Comparison of the predicted affordance and hand masks of the models on sampled images from the real testing sets. The segmentation masks are overlayed on the RGB images. KEY - GT: ground-truth, — *graspable*, — *contain*, — *arm*.

Table 4.3 Hand-occluded affordance segmentation models: size and computational cost.

	ACANet	RN18-U	RN50-F	DRNAtt
# parameters	20.82	14.32	66.40	17.38
GFLOPs	85.17	38.28	239.33	149.90

KEY – # parameters: number of parameters (in millions), GFLOPs: Giga Floating-point operations, RN50-F: ResNet50-FastFCN [53], RN18-U: ResNet18-UNET, DRNAtt [38].

We briefly discuss the complexity of the models measured in number of parameters (in millions) and number of operations (as Giga Floating-point or GFLOPs). Table 4.3 shows that RN50-F has the highest amount of parameters and operations, but the generalisation performance is worse than ACANet. DRNatt has 3 millions fewer parameters than ACANet, but $1.7\times$ more operations than ACANet and the performance on real-data is worse in all classes. Finally, ACANet has an increased number of parameters (6 millions) and GFLOPs (2.2 times) compared to the baseline RN18-U, thus contributing to the higher performance in almost all classes and testing sets. The optimisation of ACANet computational load is outside of the scope of this study, however, one way to reduce the parameters and the number of Floating-point operations of ACANet could be to perform the object and arm segmentation in the same branch.

Table 4.4 shows the results of the ablations Skip and -UNet, compared with ACANet. In most testing sets, ACANet has the best results for the affordance classes, while the modified version of the UNet block is responsible for a performance drop in the *arm* class (-UNet has higher J values). For what concerns the CHOC-B testing set, ACANet outperforms the variations in the affordance classes, while in CHOC-I ACANet improves the *graspable* class. In HO-3D, ACANet outperforms other methods in the affordance classes, while -UNet has an improvement in the arm class of 5 percentage points. In CCM, ACANet is the best performing model for the categories *contain*, and in *arm* with a margin of 3.18 percentage points from the -UNet version, while Skip improves of about 2 percentage points the *graspable* score. These result

Table 4.4 Comparison of the affordance and arm segmentation results between ACANet and its variations on the two mixed-reality test sets and on the two real test sets.

Test set	Model	*graspable*			*contain*			*arm*		
		P	*R*	*J*	*P*	*R*	*J*	*P*	*R*	*J*
CHOC-B	**ACANet**	**97.09**	**96.60**	**93.88**	**89.46**	**94.67**	**85.17**	96.48	96.52	93.24
	- UNet	96.39	96.48	93.11	89.20	94.20	84.55	**96.54**	**96.68**	**93.45**
	Skip	96.29	96.37	92.92	87.25	94.18	82.79	96.38	95.94	92.61
CHOC-I	**ACANet**	**96.36**	96.51	**93.11**	88.72	76.68	69.86	96.94	96.77	93.90
	- UNet	96.25	**96.55**	93.04	**88.97**	76.02	69.47	96.83	**96.93**	**93.95**
	Skip	96.09	96.43	92.79	86.71	**78.85**	**70.35**	**96.99**	96.36	93.57
HO-3D	**ACANet**	**89.72**	80.78	**73.93**	79.20	**90.43**	**73.07**	**61.95**	53.02	40.00
	- UNet	87.58	**81.33**	72.92	**79.67**	87.84	71.75	59.96	**64.35**	**45.01**
	Skip	87.05	72.36	65.33	79.35	86.56	70.65	58.84	37.26	29.55
CCM	**ACANet**	10.22	**86.50**	10.06	**45.40**	37.46	**25.83**	49.47	**45.35**	**31.00**
	- UNet	10.34	83.66	10.13	44.76	34.03	23.97	43.80	40.89	26.82
	Skip	**12.41**	80.79	**12.05**	25.81	**43.72**	19.37	**54.72**	31.49	24.98

Highlighted in **bold** the proposed model, and the best performing results.
KEY – P: per-class precision, P: per-class recall, J: per-class Jaccard Index,
CHOC-B: the CHOC-AFF testing set with new backgrounds,
CHOC-I: the CHOC-AFF testing set with new instances.

show that ACANet is on average the best performing version and that the ablations Skip and -UNet changes provide worse results in most classes.

4.4 Summary

In this chapter, we tackled the visual affordance segmentation using our framework targeting two settings: objects on a tabletop targeting semi-autonomous applications and hand-occluded containers. In the former case, we considered the active role of the human and we proposed a fine-tuning approach to locate the objects in the scene without learning their classes (objectness detector). In the latter setting, we proposed ACANet, a multi-branch convolutional neural network that fuses object and hand segmentation mask with the affordance features to specialise filters for object and hand regions. Experiments highlight that our approach helps increasing the mAP score of lightweight object detectors in UMD and IIT testing sets. The

proposed pipeline consisting in the sequence of objectness detection and affordance segmentation allows to improve performance with respect to the baseline which considers only the lightweight affordance segmentation. This is due to the fact that a small amount of objects in the datasets are in foreground, hence the assumptions selected affordance models are not satisfied. Training ACANet on an annotated dataset with mixed-reality images of hand-held containers leads to better generalisation to real images with containers not seen in training and with new backgrounds (e.g., on the HO-3D dataset or on-the-fly acquired images) compared to other models. Moreover, ACANet outperforms alternative methods when segmenting the *graspable* area and the person's *arm*, and can achieve a more accurate segmentation even when the objects are in more challenging poses caused by how the person holds the object.

Chapter 5

Conclusion

5.1 Summary of achievements

In this thesis, we tackled Visual Affordance Prediction i.e., the localisation of objects of interest, the identification of object regions an agent can interact with, and the prediction of the object mass. The mass estimation can be used by the agent to plan the motion towards the object and to regulate the force to perform the intended action. The other part of the framework is focused on identifying the functional regions of objects that can guide the agent to reach the object and to perform the action. The prediction of object properties and affordances from visual data presents challenges due to the variations in object shape and material, and occlusions due to other objects in the scene or human manipulation, that influence the appearance of objects, hence performance of models.

In the following, we discuss the proposed contributions and the obtained results:

1. We designed a strategy to locate the objects of interest in the scene. In case of human manipulation our procedure takes into account the eventual distance of the object with respect to a fixed-frontal camera. Our strategy performs instance segmentation in each frame of a recording using a pre-trained model, stores the RGB crops, the depth crops, and the average distance of the object obtained averaging the masked depth values. Finally, based on an heuristic, the method selects the K candidates that are closer to the camera (lower average

depth). This strategy produces a low percentage of false positives (8%) when used on the CORSMAL containers manipulation training set [125], allowing for the collection of a small dataset to train a mass estimation model. In case of wearable robotic application, we proposed a fine-tuning strategy to learn the detection of objects regardless of the category (objectness) following the rationale that the human already knows what is the object of interest. We targeted lightweight object detectors to open up to the wearable applications in which resource-constrained devices are employed. We showed that the objectness fine-tuning improves the detection performance of lightweight models on two testing sets for affordance segmentation.

2. We designed a mass estimation model that learns to combine color and geometric (object crop aspect ratio compared to the whole image, and average depth) information of object patches to predict the container mass regardless of the content. Our mass estimation model shows high relative absolute error when predicting the mass of objects substantially different from the ones in the training set (see cup class of the first fold in Fig. 3.7), while the error is lower in case of similar containers. Our model outperforms most baselines on the challenging CORSMAL Containers Manipulation testing sets consisting of people manipulating container instances never seen during training.

3. We designed a pipeline composed by the sequence of objectness detection and affordance segmentation models to restrict the input of affordance segmentation models to objects, instead of the whole scene. This cascade of models is proposed to overcome the assumption of lightweight affordance segmentation models that objects should be in foreground and completely visible. The pipeline improves the performance of lightweight affordance segmentation model on two affordance segmentation testing sets. We designed a multi-branch UNet-like model (ACANet) to segment affordances tackling the hand-occluded setting. The auxiliary branches separate the arm and object, while the fusion module weighs the feature maps of the main branch penalising pixels outside those regions. ACANet is trained using the extended annotation of mixed-reality images of

hand-occluded containers and shows better generalisation performance than previous models to unseen backgrounds and object instances on both mixed-reality and real data.

4. We extended the annotation of a mixed-reality dataset consisting of synthetic hands occluding synthetic containers over a real background to cover the lack of a dataset tackling hand-occluded object affordance segmentation.

5.2 Future work

How to design a method that can accurately predict the object affordance in case of hand occlusions is still an open challenge. In the affordance prediction literature objects are mostly placed on a tabletop, hence considering hand-occluded objects is still underexplored and can enable assistive and collaborative applications. The hand-occluded setting poses additional challenges to vision models because objects are partially visible. Indeed, almost no vision model trained without hand occlusion can accurately predict the functional regions when the objects are occluded by the hand. There are numerous examples of complete working systems that may benefit this thesis contributions, mainly in the field of human-robot collaboration and wearable robotics. In assistive robotics [39] and manufacturing applications [12], an autonomous robot may take an object from the human hand and place it at a designated location (e.g. a table) [105, 128] or perform an action with the exchanged object e.g. assembling pieces [12]. The overall framework showed in Fig. 1.3 allows the vision system to locate the object, predict the mass and the graspable regions since human may hold the object in different ways. Then the robot plans how to interact with the object avoiding the human hand and how to perform the action. Another example is in the wearable robotics field, in which a human wears an additional robotic arm mounted as an extension [117] or with a prosthetic hand [22, 83]. The use of our framework is similar to the one described before, however these applications are semi-autonomous, meaning that the human is responsible for the macro-movements in the environment,

while the autonomous part is in the control of micro-movements e.g., movements of the fingers, or rotating the wrist.

In the following, we discuss future directions for our work to encourage research in this area, mainly concerning: data collection, large-scale egocentric dataset, mass and graspable regions fusion, and robotic experiments.

Containers are common objects in everyday life and they have high variability in instances, materials and setting since they can be manipulated, filled, exchanged, affecting their appearance. There are also other objects that could be of interest in assistive applications, such as tools already present in other affordance dataset [40, 80, 82]: hammers, pans, scissors, forks, and knives. The collection of data through mixed-reality [120] and fully synthetic techniques [14, 17] is a widely adopted approach, with simulated images that are more and more realistic thanks to advancements in rendering and the availability of new tools [19, 37]. One of the limitations of models trained using synthetic data is the "reality gap" i.e., vision models can underperform on real data which is the reason why we assessed the performance of models with real data in Sec. 4.3.4. The main cause of the potential performance drop lays in the fact that vision models process color information and the synthetic pixels may have different color distribution than data captured from a real camera. Current methods that adopt synthetic datasets apply augmentation techniques like domain randomisation [14] that changes the colors and the texture patterns of images to potentially cover the intensity changes in real images or replace the background with images taken from other dataset [45], mitigating the effect of the reality gap. One direction for future work could be to extend the study of hand-occluded setting also to different object categories than containers, including techniques to render more realistic objects and arms, and analysing the impact of the real backgrounds including more scenes and lighting conditions to further improve the performance on real data. Alternatively, some recent works proposed solutions to annotate real datasets using Neural Radiance Field [27] techniques or 6D pose and object reconstruction techniques [40]. In these cases the direction would be to adapt existing techniques to obtain the affordance annotation e.g., annotating the affordance on CAD models, that still remains a time

consuming and expensive task. This approach would allow to obtain the annotation only for the chosen object instances, so the generalisation to unseen instances or different scenes would not be guaranteed.

Recently, some large-scale egocentric datasets were collected capturing people that perform different actions in different environments and worlwide locations such as EPIC-KITCHENS [21] and Ego4D [36]. EPIC-KITCHENS totals 55 hours (11.5 million frames) of recordings captured from a camera mounted on the head of a person performing daily kitchen activities (almost 150 actions), from cleaning the dishes to cooking, including natural interactions with kitchenware and appliances. Ego4D has more than 3000 hours of recordings with the camera wearer performing daily-life indoor and outdoor activities spanning scenarios like house-hold, workplace, leisure. The egocentric viewpoint could be used to tackle applications involving humanoid robots [114, 75], or also in case of augmented or virtual reality. The availability of a large number of images captured all around the world allows to improve the models generalisation properties increasing the variability of performed actions, objects in the scene, and lighting. The large number of actions and objects allows to increase the number of seen hand-object interactions during training, and increase the number of object instances that may not be the same in different geographical locations. Moreover, large scale datasets usually are captured from multiple sources to open the potential application, hence multiple modalities are captured e.g., eye-gaze, text, audio, kinetic measures (e.g. accelerations and rotations of cameras). Training models in the multi-modal setting or multi-tasking indeed leads to improvements in the learned representation [89, 98]. Compared to the third-person perspective, the egocentric perspective introduces additional challenges e.g., self-motion and a higher degree of occlusions during the manipulation of objects.

The final phases of the overall framework i.e., fusing the mass and the graspable regions information was not addressed in this thesis. The fusion between the graspable region and the mass provides the information about where the agent is supposed to interact with the object to perform the intended action, guiding the motion planning, the force regulation hence the end effector grasping. The integration of mass and graspable

regions information could be learned from human demonstration, since humans interact differently based on the objects mass [63]. Humans have different ways of grasping objects depending on the action they want to perform and how the mass influences the action through the physics of the interaction. A recent trend is the intersection of physics and machine learning [59] that consists in embedding the physics knowledge of a phenomenon in a machine learning model. In this case, the direction would be to design a model that embeds the physics of the object manipulated by the human hand to predict the most appropriate region of interaction taking into account the mass, with the aim of using such model on a robot. Alternatively, the vision based grasping could be learned through reinforcement learning techniques [84], through the sequence of actions and the regions that lead to the success of the interaction. This last approach could require a long time due to the number of trial the robot needs to perform to train the model, or it could be run in simulations to speed up the training, in any case requiring a proper definition of success metric to reward the model.

Real robotic experiments could show the improvements that the presented framework has with respect to the existing solutions for example in collaborative applications [105, 101, 128] and more importantly show the failure cases that will highlight aspects to work on to further improve the framework. The affordance segmentation models can be used with existing models for grasping with the additional depth map [3, 29, 35] to select the end effector poses that lay inside the predicted graspable regions. Potential limitations of this approach could be in presence of transparent objects that cause noisy depth maps, however some works that study how to reconstruct the depth map of transparent objects could be integrated in the method [58]. Moreover, the integration of computer vision models in robotics may be subject to runtime constraints to obtain reactive systems. While fully autonomous robots can employ powerful Graphical Processing Units (GPUs) to run computer vision models [24, 82, 129, 130], in semi-autonomous applications like prosthetics the available computational capability of electronic systems is reduced due to cost, consumption, and weight issues [83]. When the computation is performed on-board, the constraints are similar to the ones of smartphones, embedded GPUs, or even microcontrollers.

To scale down the models computational load different directions can be explored either at design level or deployment level, and they are not mutually exclusive. At the design level, Neural architecture search techniques adds to the parameters optimisation problem also the problem of finding the combination of neural network layers [26]. By penalising the computational cost in the loss function some resource-constrained solutions can be found. Alternatively, knowledge distillation techniques can be used to train more lightweight student models with the supervision of cumbersome teacher models [49]. At the deployment level there are different ways of reducing the computational load of models such as quantisation that reduces the number of bits used to represent the weights and the operations of models, hence impacting the memory required to store the model. In this case, the processing speed improves only if the quantisation is supported by the hardware [56]. Lastly, pruning techniques select keep some of the model parameters based on the performance measures, discarding the remaining ones, hence impacting both storage and speed [43].

References

[1] D. Ahn, S. Kim, H. Hong, and B. C. Ko. Star-transformer: a spatio-temporal cross attention transformer for human action recognition. In *IEEE Winter Conf. Appl. Comput. Vis.*, 2023.

[2] S. Ainetter and F. Fraundorfer. End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from RGB. In *IEEE Int. Conf. Robotics Autom.*, 2021.

[3] U. Asif, J. Tang, and S. Harrer. Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices. In *International Joint Conferences on Artificial Intelligence*, 2018.

[4] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. In *arXiv:2004.10934v1 [cs.CV]*, 2020.

[6] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis—a survey. *IEEE Trans. Robotics*, 30(2):289–309, 2013.

[7] M. N. Castro and S. Dosen. Continuous semi-autonomous prosthesis control using a depth sensor on the hand. *Frontiers in Neurorobotics*, 16:814973, 2022.

[8] K. Chaudhary, K. Okada, M. Inaba, and X. Chen. Predicting part affordances of objects using two-stream fully convolutional network with multimodal inputs. In *IEEE Int. Conf. Intell. Robot Syst.*, 2018.

[9] A. Chemero. An outline of a theory of affordances. *Ecological Psychology*, 15:181 – 195, 2003.

[10] D. Chen, D. Kong, J. Li, S. Wang, and B. Yin. Adosmnet: a novel visual affordance detection network with object shape mask guided feature encoders. *Multimedia Tools and Applications*, pages 1–25, 2023.

[11] D. Chen, D. Kong, J. Li, S. Wang, and B. Yin. A survey of visual affordance recognition based on deep learning. *IEEE Trans. Big Data*, pages 1–20, 2023.

[12] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13, 2016.

[13] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259v2 [cs.CL]*, 2014.

[14] A. D. Christensen, D. Lehotskỳ, M. W. Jørgensen, and D. Chrysostomou. Learning to segment object affordances on synthetic data for task-oriented robotic handovers. In *Brit. Mach. Vis. Conf.*, 2022.

[15] G. Christmann and J. Song. 2020 CORSMAL Challenge - Team NTNU-ERCReport, 2020. https://corsmal.eecs.qmul.ac.uk/resources/challenge/2020.11.30_CORSMAL_NTNU-ERC_Report.pdf.

[16] F. Chu, R. Xu, and P. Vela. Learning affordance segmentation for real-world robotic manipulation via synthetic images. *IEEE Robotics Autom. Lett.*, 4(2):1140–1147, 2019.

[17] F.-J. Chu, R. Xu, L. Seguin, and P. A. Vela. Toward Affordance Detection and Ranking on Novel Objects for Real-World Robotic Manipulation. *IEEE Robotics Autom. Lett.*, 4(4):4070–4077, 2019.

[18] F.-J. Chu, R. Xu, and P. A. Vela. Real-world multiobject, multigrasp detection. *IEEE Robotics Autom. Lett.*, 3(4):3355–3362, 2018.

[19] Blender Online Community. Blender - a 3D modelling and rendering package. Stichting Blender Foundation, Amsterdam, 2018. Blender Foundation.

[20] L. Cui, X. Chen, H. Zhao, G. Zhou, and Y. Zhu. Strap: Structured object affordance segmentation with point supervision. *arXiv:2304.08492v1 [cs.CV]*, 2023.

[21] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Eur. Conf. Comput. Vis.*, 2018.

[22] J. DeGol, A. Akhtar, B. Manja, and T. Bretl. Automatic grasp selection using a camera in a hand prosthesis. In *Annu. Int. Conf. IEEE Eng. Med. Biol. - Proc.*, 2016.

[23] S. Deng, X. Xu, C. Wu, K. Chen, and K. Jia. 3D affordancenet: A benchmark for visual object affordance understanding. In *Conf. Comput. Vis. Pattern Recognit.*, 2021.

[24] T. Do, A. Nguyen, and I. Reid. AffordanceNet: An end-to-end deep learning approach for object affordance detection. In *IEEE Int. Conf. Robotics Autom.*, 2018.

[25] S. Donaher, A. Xompero, and A. Cavallaro. Audio classification of the content of food containers and drinking glasses. In *Eur. Signal Process. Conf.* IEEE, 2021.

[26] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[27] F. Erich, N. Chiba, Y. Yoshiyasu, N. Ando, R. Hanai, and Y. Domae. Neural-labeling: A versatile toolset for labeling vision datasets using neural radiance fields. *IEEE Int. Conf. Comput. Vis.*, 2023.

[28] J. Fajardo, V. Ferman, D. Cardona, G. Maldonado, A. Lemus, and E. Rohmer. Galileo hand: An anthropomorphic and affordable upper-limb prosthesis. *IEEE access*, 8:81365–81377, 2020.

[29] H.-S. Fang, C. Wang, M. Gou, and C. Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Conf. Comput. Vis. Pattern Recognit.*, 2020.

[30] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu. Dual attention network for scene segmentation. In *Conf. Comput. Vis. Pattern Recognit.*, 2019.

[31] J. J. Gibson and L. Carmichael. *The senses considered as perceptual systems*, volume 2. Houghton Mifflin Boston, 1966.

[32] R. Girshick. Fast r-cnn. In *IEEE Int. Conf. Comput. Vis.*, 2015.

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conf. Comput. Vis. Pattern Recognit.*, 2014.

[34] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[35] M. Gou, H.-S. Fang, Z. Zhu, S. Xu, C. Wang, and C. Lu. RGB matters: Learning 7-DoF grasp poses on monocular RGBD images. In *IEEE Int. Conf. Robotics Autom.*, 2021.

[36] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4D: Around the world in 3,000 hours of egocentric video. In *Conf. Comput. Vis. Pattern Recognit.*, 2022.

[37] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, et al. Kubric: A scalable dataset generator. In *Conf. Comput. Vis. Pattern Recognit.*, 2022.

[38] Q. Gu, J. Su, and L. Yuan. Visual affordance detection using an efficient attention convolutional neural network. *Neurocomputing*, 440:36–44, 2021.

[39] Y. Gu, A. Thobbi, and W. Sheng. Human-robot collaborative manipulation through imitation and reinforcement learning. In *IEEE International Conference on Information and Automation*, 2011.

[40] A. Guo, B. Wen, J. Yuan, J. Tremblay, S. Tyree, J. Smith, and S. Birchfield. Handal: A dataset of real-world manipulable object categories with pose annotations, affordances, and reconstructions. In *IEEE Int. Conf. Intell. Robot Syst.*, 2023.

[41] D. Hadjivelichkov, S. Zwane, L. Agapito, M. P. Deisenroth, and D. Kanoulas. One-shot transfer of affordance regions? affcorrs! In *Conference on Robot Learning*. PMLR, 2023.

[42] S. Hampali, M. Rad, M. Oberweger, and V. Lepetit. Honnotate: A method for 3D annotation of hand and object poses. In *Conf. Comput. Vis. Pattern Recognit.*, 2020.

[43] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representation*, 2015.

[44] M. Hassanin, S. Khan, and M. Tahtali. Visual affordance and function understanding: A survey. *ACM Computing Surveys (CSUR)*, 54(3):1–35, 2021.

[45] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid. Learning joint reconstruction of hands and manipulated objects. In *Conf. Comput. Vis. Pattern Recognit.*, 2019.

[46] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE Int. Conf. Comput. Vis.*, 2017.

[47] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conf. Comput. Vis. Pattern Recognit.*, 2016.

[48] K. Hedvig, R. Javier, and K. Danica. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, 2011.

[49] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *Adv. Neural Inf. Process. Syst.*, 2015.

[50] Q. Hou, D. Zhou, and J. Feng. Coordinate attention for efficient mobile network design. In *Conf. Comput. Vis. Pattern Recognit.*, 2021.

[51] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al. Searching for mobilenetv3. In *Conf. Comput. Vis. Pattern Recognit.*, 2019.

[52] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861v1 [cs.CV], 2017.

[53] S.M Hussain.S, L. Liu, W. Xu, and C. Lu. FPHA-Afford: A domain-specific benchmark dataset for occluded object affordance estimation in human-object-robot interaction. In *IEEE Int. Conf. Image Process.*, 2020.

[54] V. Iashin, F. Palermo, G. Solak, and C. Coppola. Top-1 CORSMAL Challenge 2020 Submission: Filling Mass Estimation Using Multi-Modal Observations of Human-Robot Handovers. In *IEEE Conf. Pattern Recognit. Workshops and Challenges*, 2021.

[55] R. Ishikawa, Y. Nagao, R. Hachiuma, and H. Saito. Audio-Visual Hybrid Approach for Filling Mass Estimation. In *IEEE Conf. Pattern Recognit. Workshops and Challenges*, 2021.

[56] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Conf. Comput. Vis. Pattern Recognit.*, 2018.

[57] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor. Affordances in psychology, neuroscience, and robotics: A survey. *IEEE Trans. Cogn. Dev. Syst.*, 10(1):4–25, 2016.

[58] J. Jiang, G. Cao, T. Do, and S. Luo. A4t: Hierarchical affordance detection for transparent objects depth reconstruction and manipulation. *IEEE Robotics Autom. Lett.*, 7(4):9826–9833, 2022.

[59] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[60] Z. O. Khalifa and S. A. A. Shah. Towards visual affordance learning: A benchmark for affordance segmentation and recognition. *arXiv:2203.14092v2 [cs.CV]*, 2022.

[61] S. Kumra and C. Kanan. Robotic grasp detection using deep convolutional neural networks. In *IEEE Int. Conf. Intell. Robot Syst.*, 2017.

[62] S. R. Lakani, A. J. Rodríguez-Sánchez, and J. Piater. Towards Affordance Detection for Robot Manipulation using Affordance for Parts and Parts for Affordance. *Autonomous Robots*, 43(5):1155–1172, 2019.

[63] L. Lastrico, N. F. Duarte, A. Carfì, F. Rea, A. Sciutti, F. Mastrogiovanni, and J. Santos-Victor. Expressing and inferring action carefulness in human-to-robot handovers. *IEEE Int. Conf. Intell. Robot Syst.*, 2023.

[64] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *Int. J. Robot. Res.*, 34(4-5):705–724, 2015.

[65] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Conf. Comput. Vis. Pattern Recognit.*, 2017.

[66] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *Eur. Conf. Comput. Vis.*, 2014.

[67] Q. Liu, F. Feng, C. Lan, and R. H. M. Chan. VA2Mass: Towards the Fluid Filling Mass Estimation via Integration of Vision and Audio Learning. In *IEEE Conf. Pattern Recognit. Workshops and Challenges*, 2021.

[68] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Eur. Conf. Comput. Vis.* Springer, 2016.

[69] H. Luo, W. Zhai, J. Zhang, Y. Cao, and D. Tao. One-shot affordance detection. *International Joint Conference on Artificial Intelligence*, 2021.

[70] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation, 2019.

[71] R. Margolin, L. Zelnik-Manor, and A. Tal. How to evaluate foreground maps? In *Conf. Comput. Vis. Pattern Recognit.*, 2014.

[72] T. Matsubara, S. Otsuki, Y. Wada, H. Matsuo, T. Komatsu, Y. Iioka, K. Sugiura, and H. Saito. Shared Transformer Encoder with Mask-Based 3D Model Estimation for Container Mass Estimation. In *IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2022.

[73] M. Matsumoto and T. Nishimura. Mersenne Twister: a 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Trans. Modeling Comput. and Simulation*, 8(1):3–30, 1998.

[74] V. Mazzia, S. Angarano, F. Salvetti, F. Angelini, and M. Chiaberge. Action transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, 124:108487, 2022.

[75] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. Von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, et al. The icub humanoid robot: An open-systems platform for research in cognitive development. *Neural networks*, 23(8-9):1125–1134, 2010.

[76] F. Milletari, N. Navab, and S. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Int. Conf. 3D Vision*, 2016.

[77] A. Modas, Alessio Xompero, R. Sanchez-Matilla, P. Frossard, and A. Cavallaro. Improving Filling Level Classification with Adversarial training. In *IEEE Int. Conf. Image Process.*, 2021.

[78] A. Mousavian, C. Eppner, and D. Fox. 6-DOF graspnet: Variational grasp generation for object manipulation. In *IEEE Int. Conf. Comput. Vis.*, 2019.

[79] L. Mur-Labadia, R. Martinez-Cantin, and J. J. Guerrero. Bayesian deep learning for affordance segmentation in images. In *IEEE Int. Conf. Robotics Autom.*, 2023.

[80] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos. Affordance detection of tool parts from geometric features. In *IEEE Int. Conf. Robotics Autom.*, 2015.

[81] A. Nguyen, D. Kanoulas, D. Caldwell, and N. Tsagarakis. Detecting object affordances with convolutional neural networks. In *IEEE Int. Conf. Intell. Robot Syst.*, 2016.

[82] A. Nguyen, D. Kanoulas, D. Caldwell, and N. Tsagarakis. Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *IEEE Int. Conf. Intell. Robot Syst.*, 2017.

[83] A. T. Nguyen, M. W. Drealan, D. K. Luu, M. Jiang, J. Xu, J. Cheng, Q. Zhao, E. W. Keefer, and Z. Yang. A portable, self-contained neuroprosthetic hand with deep learning-based finger control. *Journal of neural engineering*, 18(5):056051, 2021.

[84] C. Oh, Y. L. Pang, and A. Cavallaro. Ohpl: One-shot hand-eye policy learner. In *IEEE Int. Conf. Intell. Robot Syst.* IEEE, 2021.

[85] F. Osiurak, Y. Rossetti, and A. Badets. What is an affordance? 40 years later. *Neuroscience & Biobehavioral Reviews*, 77:403–417, 2017.

[86] Y. L. Pang, A. Xompero, C. Oh, and A. Cavallaro. Towards safe human-to-robot handovers of unknown containers. In *IEEE Int. Conf. Robot and Human Interactive Comm.*, 2021.

[87] A. Pieropan, C. H. Ek, and H. Kjellström. Functional object descriptors for human activity modeling. In *IEEE Int. Conf. Robotics Autom.* IEEE, 2013.

[88] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese. Keto: Learning keypoint representations for tool manipulation. In *IEEE Int. Conf. Robotics Autom.*, 2020.

[89] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *Int. Conf. on Machine Learning*, 2021.

[90] E. Ragusa, C. Gianoglio, S. Dosen, and P. Gastaldo. Hardware-aware affordance detection for application in portable embedded systems. *IEEE Access*, 9:123178–123193, 2021.

[91] J. Raskin. *The humane interface: new directions for designing interactive systems*. Addison-Wesley Professional, 2000.

[92] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *IEEE Int. Conf. Robotics Autom.*, 2015.

[93] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Conf. Comput. Vis. Pattern Recognit.*, 2016.

[94] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Conf. Comput. Vis. Pattern Recognit.*, 2017.

[95] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. In *arXiv:1804.02767v1 [cs.CV]*, 2018.

[96] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Adv. Neural Inf. Process. Syst.*, 2015.

[97] S. Rezapour Lakani, A. Rodríguez-Sánchez, and J. Piater. Towards affordance detection for robot manipulation using affordance for parts and parts for affordance. *Auton. Robots*, 43:1155–1172, 2019.

[98] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Conf. Comput. Vis. Pattern Recognit.*, 2022.

[99] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *International Conference on Learning Representation*, 2014.

[100] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Int. Conf. Med. Image Comput. Comput. Assist. Interv.*, 2015.

[101] P. Rosenberger, A. Cosgun, R. Newbury, J. Kwan, V. Ortenzi, P. Corke, and M. Grafinger. Object-independent human-to-robot handovers using real time robotic vision. *IEEE Robotics Autom. Lett.*, 6(1):17–23, 2020.

[102] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[103] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *Int. J. Comput. Vis.*, 77:157–173, 2008.

[104] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. Clear grasp: 3D shape estimation of transparent objects for manipulation. In *IEEE Int. Conf. Robotics Autom.* IEEE, 2020.

[105] R. Sanchez-Matilla, K. Chatzilygeroudis, A. Modas, N. F. Duarte, A. Xompero, P. Frossard, A. Billard, and A. Cavallaro. Benchmark for human-to-robot handovers of unseen containers with unknown filling. *IEEE Robotics Autom. Lett.*, 5(2):1642–1649, 2020.

[106] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Conf. Comput. Vis. Pattern Recognit.*, 2018.

[107] J. Sawatzky, A. Srikantha, and J. Gall. Weakly supervised affordance detection. In *Conf. Comput. Vis. Pattern Recognit.*, 2017.

[108] W. Shi, J. Caballero, F. Huszár, J. Totz, A. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Conf. Comput. Vis. Pattern Recognit.*, 2016.

[109] T. Standley, O. Sener, D. Chen, and S. Savarese. image2mass: Estimating the mass of an object from its image. In *Conference on Robot Learning*. PMLR, 2017.

[110] T. A. Stoffregen. Affordances as properties of the animal-environment system. *Ecological Psychology*, 15(2):115–134, 2003.

[111] C. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Int. Conf. Med. Image Comput. Comput. Assist. Interv., DLMIA and ML-CDS Workshops*. Springer, 2017.

[112] J. Sun, J. L. Moore, A. Bobick, and J. M. Rehg. Learning visual object categories for robot affordance prediction. *Int. J. Robot. Res.*, 29(2-3):174–197, 2010.

[113] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Conf. Comput. Vis. Pattern Recognit.*, 2019.

[114] N. G. Tsagarakis, D. G. Caldwell, F. Negrello, W. Choi, L. Baccelliere, V.-G. Loc, J. Noorden, L. Muratore, A. Margan, A. Cardellino, et al. Walk-man: A high-performance humanoid platform for realistic environments. *Journal of Field Robotics*, 34(7):1225–1259, 2017.

[115] C.-Y. Tsai, H.-P. Lin, and Y.-C. Chiu. An esp-based lightweight model for joint object detection and affordance segmentation. In *Asia-Pacific Conference on Intelligent Robot Systems*. IEEE, 2021.

[116] F. Vasile, E. Maiettini, G. Pasquale, A. Florio, N. Boccardo, and L. Natale. Grasp pre-shape selection by synthetic training: Eye-in-hand shared control on the hannes prosthesis. In *IEEE Int. Conf. Intell. Robot Syst.*, 2022.

[117] V. Vatsal and G. Hoffman. Design and analysis of a wearable robotic forearm. In *IEEE Int. Conf. Robotics Autom.* IEEE, 2018.

[118] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *Conf. Comput. Vis. Pattern Recognit.*, 2019.

[119] H. Wang, C. Zhu, Z. Ma, and C. Oh. Improving Generalization of Deep Networks for Estimating Physical Properties of Containers and Fillings. In *IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2022.

[120] X. Weber, A. Xompero, and A. Cavallaro. A mixed-reality dataset for category-level 6D pose and size estimation of hand-occluded containers. *arXiv:2211.10470v1 [cs.CV]*, 2022.

[121] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Müller, A. Evans, D. Fox, J. Kautz, and S. Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In *Conf. Comput. Vis. Pattern Recognit.*, 2023.

[122] H. Wu, J. Zhang, K. Huang, K. Liang, and Y. Yu. FastFCN: Rethinking dilated convolution in the backbone for semantic segmentation. *arXiv:1903.11816v1 [cs.CV]*, 2019.

[123] A. Xompero, S. Donaher, V. Iashin, F. Palermo, G. Solak, C. Coppola, R. Ishikawa, Y. Nagao, R. Hachiuma, Q. Liu, et al. The corsmal benchmark for the prediction of the properties of containers. *IEEE Access*, 10:41388–41402, 2022.

[124] A. Xompero, Y. L. Pang, T. Patten, A. Prabhakar, B. Calli, and A. Cavallaro. Audio-visual object classification for human-robot collaboration. In *IEEE Int. Conf. Acoustics, Speech and Signal Process.* IEEE, 2022.

[125] A. Xompero, R. Sanchez-Matilla, R. Mazzon, and A. Cavallaro. CORSMAL Containers Manipulation, 2020. (1.0) [Data set]. Queen Mary University of London. https://doi.org/10.17636/101CORSMAL1.

[126] A. Xompero, R. Sanchez-Matilla, A. Modas, P. Frossard, and A. Cavallaro. Multi-View Shape Estimation of Transparent Containers. In *IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2020.

[127] R. Xu, F.-J. Chu, C. Tang, W. Liu, and P. A. Vela. An affordance keypoint detection network for robot manipulation. *IEEE Robotics Autom. Lett.*, 6(2):2870–2877, 2021.

[128] W. Yang, C. Paxton, A. Mousavian, Y. Chao, M. Cakmak, and D. Fox. Reactive human-to-robot handovers of arbitrary objects. In *IEEE Int. Conf. Robotics Autom.*, 2021.

[129] C. Yin and Q. Zhang. Object affordance detection with boundary-preserving network for robotic manipulation tasks. *Neural. Comput. Appl.*, 34(20):17963–17980, 2022.

[130] C. Yin, Q. Zhang, and W. Ren. A new semantic edge aware network for object affordance detection. *J. Intelligent & Robotic Systems*, 104(1):1–16, 2022.

[131] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Adv. Neural Inf. Process. Syst.*, 2014.

[132] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Conf. Comput. Vis. Pattern Recognit.*, 2017.

[133] P. Zech, S. Haller, S. R. Lakani, B. Ridge, E. Ugur, and J. Piater. Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271, 2017.

[134] W. Zhai, H. Luo, J. Zhang, Y. Cao, and D. Tao. One-shot object affordance detection in the wild. *Int. J. Comput. Vis.*, 130(10):2472–2500, 2022.

[135] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng. Roi-based robotic grasp detection for object overlapping scenes. In *IEEE Int. Conf. Intell. Robot Syst.*, 2019.

[136] Y. Zhang and T. Funkhouser. Deep depth completion of a single RGB-D image. In *Conf. Comput. Vis. Pattern Recognit.*, 2018.

[137] Y. Zhang, H. Li, T. Ren, Y. Dou, and Q. Li. Multi-scale fusion and global semantic encoding for affordance detection. In *Int. Joint Conf. on Neural Networks*, 2022.

[138] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Conf. Comput. Vis. Pattern Recognit.*, 2017.

[139] X. Zhao, Y. Cao, and Y. Kang. Object affordance detection with relationship-aware network. *Neural. Comput. Appl.*, 32(18):14321–14333, 2020.

[140] X. Zheng, Z. Zeng, and J. Zhang. High-level object affordance recognition. In *IEEE Int. Conf. Robot. Autom.*, 2018.

Appendix A. Background

In this appendix, we present some concepts that are useful to understand the thesis. We outline the concepts that are taken for granted in the intersection between machine learning and computer vision (Sec. A.1), and we dive more deeply in the modules (Sec. A.3) and architectures that are used either in the baselines or in the proposed methods. In particular, we discuss the main architectures to tackle semantic segmentation (Sec. A.4), object detection (Sec. A.5), and instance segmentation (Sec. A.6).

A.1 Preliminaries

In this section, we briefly highlight some concepts that are taken for granted in the thesis and we refer the reader to machine learning books for additional details [4, 34].

In this thesis, we consider computer vision models (neural networks) i.e., the composition of parametric functions that maps the input image to a prediction. These functions are also called layers and can perform different operations such as convolution that is a weighted sum between the weights of a filter and the values of a feature map obtained sliding the kernel on the featrue map, pooling that reduces the features resolution, feed-forward a weighted sum among the all inputs of the layer and the weights of the layer. Models are trained using the supervised machine learning paradigm i.e., the parameters are learned through a cost function (loss) that evaluates the error between the model prediction and the annotation. The annotation could be for example a number $m \in \mathbb{R}_{>0}$ representing the mass of the object in the image. During the training phase the backpropagation procedure updates the model parameters in a certain layer through the chain rule based on the current value, the learning rate, and

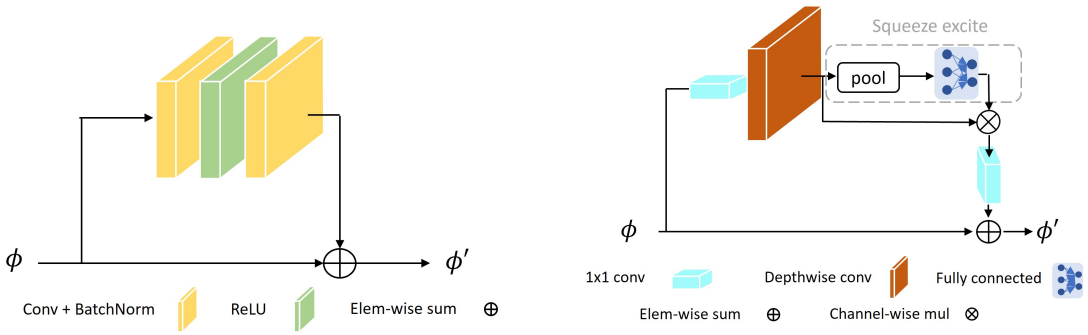
the gradient of the loss function with respect to the current layer. During the inference phase, the weights of the model are kept constant (frozen) and the information flows from the first layers of the model to the last one that outputs the prediction.

A.2 Transfer learning

Transfer learning consists in using the features of a model learned in a source task on a target task, assuming that the two tasks are similar. An example of similar tasks are scene segmentation (source) [138] and affordance segmentation (target) [81]. Transfer learning is a widely used practice because it offers the possibility of using the representation of the source domain in which the training set may have millions of images in domains where the size of the training set is not large enough to train a model from scratch limiting the overfitting phenomenon. Models pre-trained on large datasets (millions of images) learn feature representations that are indeed general enough to be used for similar tasks in vision [131]. In general, the representation can be transferred either from an architecture to another one, or maintaining the same architecture.

An example of transfer learning from an architecture to another one is knowledge distillation [49]: a pre-trained model (teacher) is used to supervise the training of another model (student) for example by adding a penalisation term in the loss that takes into account the matching between the output of the last layer of the teacher and the student, or also outputs of intermediate layers if they are compatible in size [99]. Knowledge distillation can be used for example to obtain a student model that is smaller with respect to the teacher, increasing the inference speed and reducing the computational cost without a minor drop in performance than training the smaller model from scratch.

Another common transfer learning approach, used in this thesis, is the fine-tuning that transfers the representation in the same architecture [131]. Fine-tuning is the process of adjusting the weights of a model on the target domain starting from the pre-trained version on the source domain. Fine-tuning is widely adopted when the



(a) (b)

Fig. A.1 Feature extraction blocks: (a) Residual block [47], (b) MobileNetV3 block [51].

target domain is similar to the source one and when pre-trained weights are available, because it can help reducing the amount of computations during training and improve the performance [131]. After the initialisation with pre-trained weights, the first layers of the model can be frozen to avoid the update during the backpropagation phase, leaving only the last layers that are more task specific to update their weights. An alternative approach consists in updating all the layers but starting from the pre-trained initialisation. In this way also the first features are adapted to the target domain, however the computational cost of this procedure is higher than in the former case.

A.3 Feature extraction

The backbone of a model is the part that performs feature extraction from the input image. Due to numerical reasons, the image could be pre-processed e.g., by scaling the pixel values based on the mean value and the standard deviation, or mapping the pixel values in the $[0, 1]$ range. The features extracted by applying the weights of the model on the image are used by the head of the model (the last layers) to predict the output.

Residual Network

Residual Network (ResNet) is one of the most used models in the computer vision community, and is composed by a stack of residual blocks [47]. A residual block

(see Fig. A.1a) consists of a skip connection that combines the input of the block with its processed verision $\phi' = \phi + \omega(\phi)$, where x' is the output of the block, ω can be a stack of convolutional layers with non-linear activation. In the original paper, the combination is an element-wise sum, and the skip connection avoids adding parameters to the block. Residual blocks are useful to increase the number of layers (hence parameters) limiting the accuracy degradation problem that was observed when increasing the number of stacked layers. Residual connections ease the optimisation problem, compared to models that do not employ them, exhibiting lower training error increasing the number of stacked layers. As a consequence, ResNets obtained higher accuracies than the compared models [47].

Mobile Network

Mobile Network (MobileNet) is a family of convolutional neural networks designed to perform on systems that are resource-constrained e.g., smartphones [51, 52, 106]. The first MobileNet version (MobileNetV1) is based on the depthwise separable convolutions idea, that allows to save computations hence gaining in speed with a reduced accuracy drop on common benchmarks [52]. In particular, instead of performing the standard convolution operation between a feature map and the kernels of a layer that filters and combines the feature maps, the depthwise separable convolution splits the computation into two layers, one layer for filtering and the other layer for combining. The depthwise convolution applies a single filter to each input channel, while the pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. The second MobileNet version (MobileNetV2), introduced the inverted residual bottleneck layer to remove non-linearities in the layers performing a reduction of feature maps dimensionality to maintain the representation capability [106]. First, a point-wise convolution expands the number of channels of the input feature map, applying also a nonlinear activation. Next, a depth-wise convolution performs spatial filtering using 3×3 kernels, followed by non-linear activation. Finally, the spatially-filtered feature map is projected back to a lower number of channels using another point-wise convolution. When the depth-wise uses a stride 1 the output

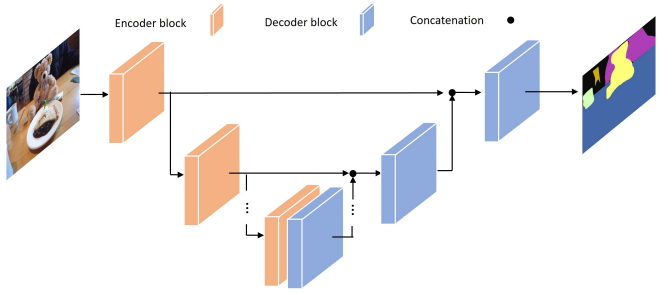
feature map and the input feature map have the same dimensionality and a residual connection is added to ease the optimisation during training. The third MobileNet version (MobileNetV3) is the result of a Neural Architecture Search that trades-off accuracy and latency combining MNasNet lightweight attention modules [113] based on squeeze and excitation in MobileNetV2 bottleneck blocks (see Fig. A.1b). Additionally, MobileNetV3 introduces new non-linear activation function (h-swish) that improves the computational cost with respect to the swish activation function based on the sigmoid [51].

A.4 Architectures for semantic segmentation

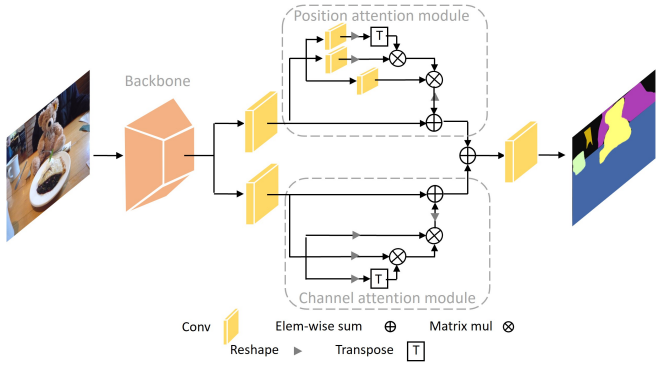
The semantic segmentation task associates each pixel in the input image to a class belonging to a pre-defined set. Most of the architectures can be divided into the backbone that performs feature extraction, and the head that maps the extracted features to the input space (or a fraction). The backbone can be replaced by any feature extractor, such as the ones introduced in Sec. A.3, as long as the feature map dimensions are compatible.

UNet

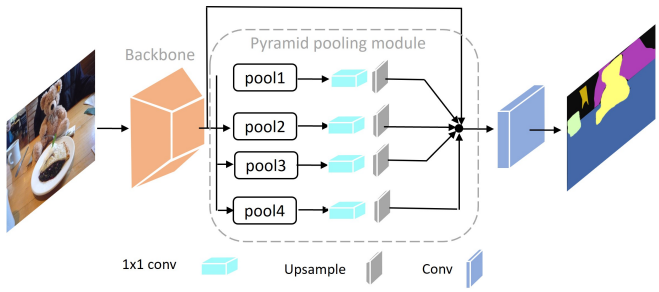
UNet is an encoder-decoder model for semantic segmentation (see Fig. A.2a) [100]. The first part of the network is called encoder because it extracts the features using convolutional layers with non-linear activations, and downsampling layers (pooling), compressing the input image to a tensor of lower width and height, but higher number of channels. The decoder maps the compressed representation back to the input space, using a symmetric structure to the encoder. The feature maps flowing in the decoder are processed using convolutional layers with non-linear activations and upsampling layers. The outputs of the encoder layers are forwarded through skip connections and concatenated with the corresponding outputs of the decoder layers to avoid the information loss when reconstructing the information. The concatenation is performed along the channel dimension.



(a) UNet



(b) DANet



(c) PSPNet

Fig. A.2 Block diagrams of semantic segmentation architectures: (a) UNet, (b) DANet, (c) PSPNet. Block diagrams are simplified to visualise the main architecture blocks. In UNet, the encoder block contains also a downsampling layer, while the decoder block the upsampling one to match tensor dimensionality.

Dual Attention Network

Dual Attention Network (DANet) combines a position attention module to learn the spatial interdependencies of features and a channel attention module to model channel interdependencies in the extracted features (see Fig. A.2b) [30]. In the former module, the self-attention mechanism captures similarity between any two positions of the feature maps through matrix multiplication and softmax activation. At a given channel of the output feature map, each pixel position is obtained by aggregating features at

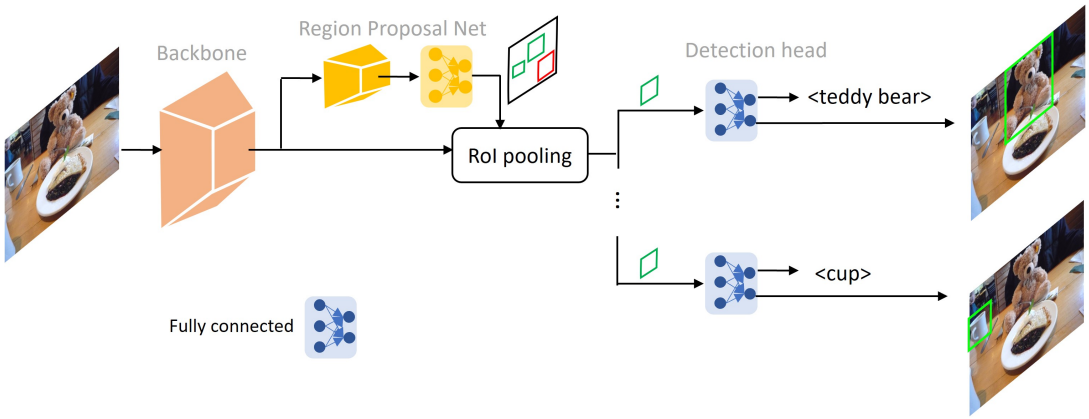
all positions using the attention map values as weights. Therefore, any two positions with similar features can contribute mutual improvement regardless of their distance in spatial dimension. In the channel attention module, the self-attention mechanism captures the channel dependencies between any two channel of the feature maps, and processes each channel using a sum weighted with the attention map values. Finally, the outputs of the two attention modules are processed with a convolutional layer, and they are fused with an element-wise sum. At last a convolution layer predicts the segmentation mask.

Pyramid Scene Parsing Network

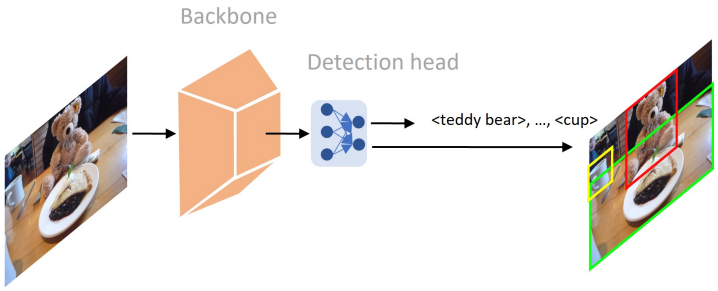
Pyramid Scene Parsing Network (PSPNet) is based on the pyramid parsing module that processes the feature map after the feature extraction phase, working on different sub-regions (see Fig. A.2c) [138]. The pyramid pooling module fuses features under four different scales to learn both global and local information at different resolutions. The coarsest level is a global pooling that produces the lowest resolution feature map. The following pyramid levels separate the feature map into different sub-regions and forms pooled representation for different locations. The output of different levels in the pyramid pooling module contains the feature map with varied sizes. 1×1 convolution layer after each level maps the number of channels to 1. An upsample layer modifies the resolution of the feature maps to match the width and height of the pyramid pooling module input. The upsampled feature maps are concatenated with the pyramid pooling module input along the channel dimension to form the output of the pyramid parsing module, carrying both local and global information. The convolutional layer maps the final feature representation into the per-pixel prediction.

A.5 Architectures for object detection

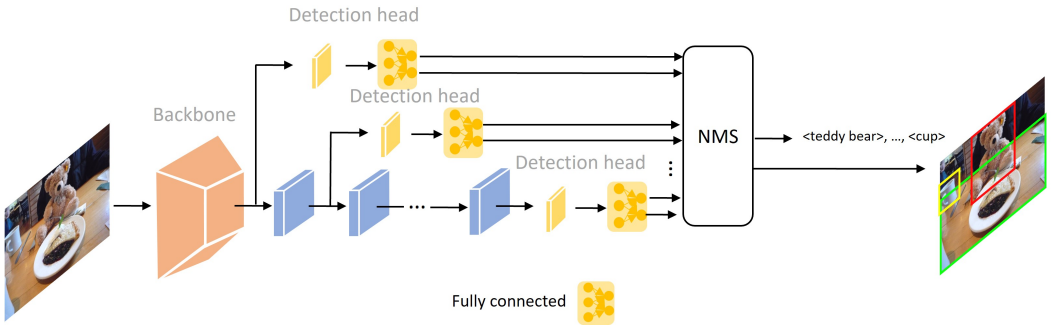
The task of object detection associates each object in an image with a category label and a bounding box. The feature extraction can be performed using any backbone. Object detection method based on convolutional neural networks can be divided into



(a) Faster R-CNN



(b) Yolo



(c) SSD

Fig. A.3 Block diagrams of object detection architectures: (a) Faster R-CNN, (b)YOLOv1, (c) SSD. Block diagrams are simplified to visualise the main architecture blocks. KEY – NMS: Non-maximum suppression.

two sets: single stage and double stage. This division depends on how the feature maps of the backbone are processed. Two stage detectors consider only some regions of the feature maps based on a proposal phase [96], while single stage detectors process the whole feature map in the detection head, without a proposal phase. Single stage detectors are more efficient and faster than double stage detectors, but less accurate.

Region-based Convolutional Neural Network

Region-based Convolutional Neural Network (R-CNN) is a family of two stage object detectors [33]. The first version of R-CNN uses existing methods to propose regions of interest and then crops and warps the regions in the image to the same fixed resolution. A convolutional neural network with a classifier on top processes the RGB images crops to predict the class. The second version is Fast R-CNN network that processes the whole input image with a Convolutional Neural Network instead of processing only the proposed regions [32]. Also in this case the proposed regions boundaries are assumed to be given by another algorithm e.g. selective search. The regions of interest are cropped in the feature map and resized to the same fixed resolution using a pooling layer. Two different fully connected heads process the pooling layer output to predict the class of the object (or background) in the region of interest and the bounding box offsets to adjust the proposed region boundary. Faster R-CNN completes Fast R-CNN with a fully convolutional region proposal network that processes the feature maps extracted from the input image, introducing the anchor boxes concept (see Fig. A.3a) [96]. Anchor boxes are fixed-size bounding boxes having various shapes and sizes to cover the likely locations of objects in the image and are used as priors. The Region Proposal Network uses a 3×3 convolutional layer that slides on the extracted feature maps, and two 1×1 convolutional layers, one for bounding boxes prediction (more precisely the offset with respect to the anchor boxes), the other for an objectness score i.e., if the box contains an object or not. The Fast R-CNN that comes after the Region Proposal Network evaluates the predicted regions of interest as previously described.

You Only Look Once Network

You Only Look Once (YOLO) is a family of convolutional neural networks designed to perform one shot object detection processing the feature map extracted from the input image without a region proposal phase [93]. The convolutional feature extraction phase outputs a feature map that is processed by the fully connected head of the model. Since the backbone preserves the locality of the input image, the feature

map can be seen as a grid divided into cells, where each cell corresponds to a patch in the input image. The head predicts for each position of the feature map (hence each cell in the original image) the bounding boxes center, the height and width of the bounding box, normalised with respect to the image, a confidence score and the object class. Non-maximum suppression and the confidence scores can be used to improve the robustness of the predictions discarding overlapping predictions(see Fig. A.3b). Successive versions of YOLO gradually updates parts of the existing model, integrating the mechanism of anchor boxes in the model (YOLOv2) [94]; predicting bounding boxes from different feature maps scales (YOLOv3) [95]; integrating batch normalisation, residual connections and updating the training strategy (YOLOv4) [5]; and tuning the anchor boxes to the data (YOLOv5)[5].

Single-Shot Multibox detector

Single-Shot Multibox detector (SSD) is another example of single stage object detection model predicting the class and the bounding boxes with a single forward pass of feature maps in the network(see Fig. A.3c) [68]. At prediction time, the network generates scores for the presence of each object category in each anchor box and adjustments to the box. Additionally, the network combines predictions from multiple feature maps with different resolutions to handle objects of various size. To target resource-constrained systems, the computational cost of SSD can be reduced by replacing standard convolutions with depthwise separable convolutions (SSDLite) [106].

A.6 Architectures for instance segmentation

The task of instance segmentation associates each detected object with a segmentation mask. The input is an RGB image and the outputs of the model are the class, the bounding box, and the mask.

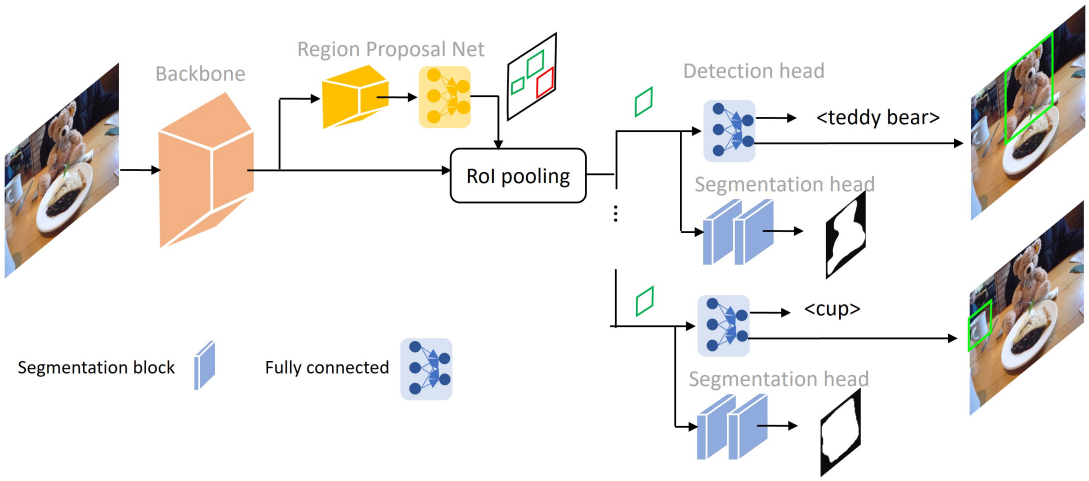


Fig. A.4 Mask R-CNN block diagram. The diagram is simplified to visualise the main architecture blocks.

Mask R-CNN

Mask R-CNN is one of the most used models for instance segmentation [46], and extends the object detector Faster R-CNN to perform instance segmentation (see Fig. A.4). The backbone outputs a feature map that is first processed by region proposal network. Faster R-CNN detection branch processes the cropped regions of interest of the feature map predicting the class and the bounding box offset for non-background regions, while an additional convolutional branch predicts a fixed-size binary segmentation mask. Authors introduced RoI average pooling that crops the feature map in the detected regions using bi-linear interpolation to limit the quantisation of the feature map introduced by the pooling operation. The mask segmentation branch is composed by convolutional and upsample layers.

Appendix B. Other research merits

Journal Papers

[J1] E. Ragusa, T. Apicella, C. Gianoglio, R. Zunino, and P. Gastaldo. Design and deployment of an image polarity detector with visual attention. *Cognitive Computation*, 1-13, 2021

[J2] V. Pandelea, E. Ragusa, T. Apicella, P. Gastaldo, and E. Cambria. Emotion recognition on edge devices: Training and deployment. *MDPI Sensors*, 21(13), 4496, 2021.

Conference papers

[C4] A. Albanese, T. Taccioli, T. Apicella, D. Brunelli, and E. Ragusa. Design and Deployment of an Efficient Landing Pad Detector. *International Conference on System-Integrated Intelligence*, 2022.

[C5] T. Apicella, E. Ragusa, A. Canepa, and Paolo Gastaldo. A Data-Driven Method for Reliability Estimation of Auxiliary Power Consumption Prediction in Commercial Electric Vehicles. *International Conference on Applications in Electronics Pervading Industry, Environment and Society (ApplePies)*, 2021.