Generic instance segmentation for object-oriented bin-picking

Matthieu Grard

Thèse de Doctorat de l'Université de Lyon

opérée au sein de l'École Centrale de Lyon

École Doctorale N°512 InfoMaths

Spécialité : Informatique

Generic Instance Segmentation
for Object-Oriented Bin-Picking

---

présentée et soutenue à huis clos par

Matthieu Grard

le 20 mai 2019

Thèse dirigée par **Pr Liming Chen**
co-encadrée par **Dr Emmanuel Dellandréa**

Jury

Mme Marie-Odile Berger		
Directrice de recherche à l'INRIA Nancy Grand Est	Présidente	
M. Thierry Chateau		
Professeur à l'Université Clermont Auvergne	Rapporteur	
M. Dimitris Samaras		
Professeur à Stony Brook University	Rapporteur	
M. Jean-Marc Odobez		
Directeur de recherche à l'Idiap Research Institute	Examinateur	
Mme Laetitia Leyrit		
Docteure à Siléane	Examinatrice	
M. Liming Chen		
Professeur à l'Ecole Centrale de Lyon	Directeur de thèse	
M. Emmanuel Dellandréa		
Maître de conférences à l'Ecole Centrale de Lyon	Co-encadrant de thèse	

Abstract

Referred to as robotic random bin-picking, a fast-expanding industrial task consists in robotizing the unloading of many object instances piled up in bulk, one at a time, for further processing such as kitting or part assembling. However, explicit object models are not always available in many bin-picking applications, especially in the food and automotive industries. Furthermore, object instances are often subject to intra-class variations, for example due to elastic deformations.

Object pose estimation techniques, which require an explicit model and assume rigid transformations, are therefore not suitable in such contexts. The alternative approach, which consists in detecting grasps without an explicit notion of object, proves hardly efficient when the object geometry makes bulk instances prone to occlusion and entanglement. These approaches also typically rely on a multi-view scene reconstruction that may be unfeasible due to transparent and shiny textures, or that reduces critically the time frame for image processing in high-throughput robotic applications.

In collaboration with Siléane, a French company in industrial robotics, we thus aim at developing a learning-based solution for localizing the most affordable instance of a pile from a single image, in open loop, without explicit object models. In the context of industrial bin-picking, our contribution is two-fold.

First, we propose a novel fully convolutional network (FCN) for jointly delineating instances and inferring the spatial layout at their boundaries. Indeed, the state-of-the-art methods for such a task rely on two independent streams for boundaries and occlusions respectively, whereas occlusions often cause boundaries. Specifically, the mainstream approach, which consists in isolating instances in boxes before detecting boundaries and occlusions, fails in bin-picking scenarios as a rectangle region often includes several instances. By contrast, our box proposal-free architecture recovers fine instance boundaries, augmented with their occluding side, from a unified scene representation. As a result, the proposed network outperforms the two-stream baselines on synthetic data and public real-world datasets.

Second, as FCNs require large training datasets that are not available

in bin-picking applications, we propose a simulation-based pipeline for generating training images using physics and rendering engines. Specifically, piles of instances are simulated and rendered with their ground-truth annotations from sets of texture images and meshes to which multiple random deformations are applied. We show that the proposed synthetic data is plausible for real-world applications in the sense that it enables the learning of deep representations transferable to real data. Through extensive experiments on a real-world robotic setup, our synthetically trained network outperforms the industrial baseline while achieving real-time performances. The proposed approach thus establishes a new baseline for model-free object-oriented bin-picking.

Keywords: computer vision, robotic bin-picking, deep learning, instance segmentation, occlusion detection, fully convolutional networks, synthetic training data

Résumé

Le dévracage robotisé est une tâche industrielle en forte croissance visant à automatiser le déchargement par unité d'une pile d'instances d'objet en vrac pour faciliter des traitements ultérieurs tels que la formation de kits ou l'assemblage de composants. Cependant, le modèle explicite des objets est souvent indisponible dans de nombreux secteurs industriels, notamment alimentaire et automobile, et les instances d'objet peuvent présenter des variations intra-classe, par exemple en raison de déformations élastiques.

Les techniques d'estimation de pose, qui nécessitent un modèle explicite et supposent des transformations rigides, ne sont donc pas applicables dans de tels contextes. L'approche alternative consiste à détecter des prises sans notion explicite d'objet, ce qui pénalise fortement le dévracage lorsque l'enchevêtrement des instances est important. Ces approches s'appuient aussi sur une reconstruction multi-vues de la scène, difficile par exemple avec des emballages alimentaires brillants ou transparents, ou réduisant de manière critique le temps de cycle restant dans le cadre d'applications à haute cadence.

En collaboration avec Siléane, une entreprise française de robotique industrielle, l'objectif de ce travail est donc de développer une solution par apprentissage pour la localisation des instances les plus prenables d'un vrac à partir d'une seule image, en boucle ouverte, sans modèles d'objet explicites. Dans le contexte du dévracage industriel, notre contribution est double.

Premièrement, nous proposons un nouveau réseau pleinement convolutionnel (FCN) pour délinéer les instances et inférer un ordre spatial à leurs frontières. En effet, les méthodes état de l'art pour cette tâche reposent sur deux flux indépendants, respectivement pour les frontières et les occultations, alors que les occultations sont souvent sources de frontières. Plus précisément, l'approche courante, qui consiste à isoler les instances dans des boîtes avant de détecter les frontières et les occultations, se montre inadaptée aux scénarios de dévracage dans la mesure où une région rectangulaire inclut souvent plusieurs instances. A contrario, notre architecture sans détection préalable de régions détecte finement les frontières entre instances, ainsi que le bord occultant cor-

respondant, à partir d'une représentation unifiée de la scène.

Deuxièmement, comme les FCNs nécessitent de grands ensembles d'apprentissage qui ne sont pas disponibles dans les applications de dévracage, nous proposons une procédure par simulation pour générer des images d'apprentissage à partir de moteurs physique et de rendu. Plus précisément, des vracs d'instances sont simulés et rendus avec les annotations correspondantes à partir d'ensembles d'images de texture et de maillages auxquels sont appliquées de multiples déformations aléatoires. Nous montrons que les données synthétiques proposées sont vraisemblables pour des applications réelles au sens où elles permettent l'apprentissage de représentations profondes transférables à des données réelles. À travers de nombreuses expériences sur une maquette réelle avec robot, notre réseau entraîné sur données synthétiques surpasse la méthode industrielle de référence, tout en obtenant des performances temps réel. L'approche proposée établit ainsi une nouvelle référence pour le dévracage orienté-objet sans modèle d'objet explicite.

Mots-clés : vision par ordinateur, dévracage robotisé, apprentissage profond, segmentation en instances, détection des occultations, réseaux entièrement convolutionnels, données d'apprentissage synthétiques

The monster saw my determination in my face and gnashed his teeth in the impotence of anger. "Shall each man," cried he, "find a wife for his bosom, and each beast have his mate, and I be alone? [...] Beware, for I am fearless and therefore powerful. I will watch with the wiliness of a snake, that I may sting with its venom. Man, you shall repent of the injuries you inflict."

— Mary Shelley, "Frankeinstein"

x

Remerciements

Je tiens à remercier Liming Chen et Emmanuel Dellandréa pour avoir accepté avec enthousiasme d'encadrer mes travaux de thèse, pour le partage de leur expérience et leurs conseils pertinents, ainsi que la société Siléane de m'avoir fait pleinement confiance dans mon projet de thèse CIFRE qui, j'espère, dévoilera de nouveaux horizons.

Merci également à Florian pour son suivi éclairé, ses remarques sagaces, ses conseils techniques et philosophiques.

Merci à Fei pour sa présence, son entrain et son soutien constant malgré les difficultés, ainsi qu'à ma famille de m'avoir supporté dans mes choix académiques et professionnels.

Merci à mes collègues de Siléane et de l'équipe Imagine du Liris pour leur présence et le plaisir de nos conversations, en particulier à Romain, Amaury, Jean-Louis, Laetitia, Sébastien, Maxime.

Merci à Léo, Antoine, Marie, Eric, Jean de leur soutien parallèle sur le plan sportif qui m'a notamment permis de représenter au mieux l'Ecole Centrale de Lyon lors des compétitions nationales.

Merci aux membres du jury d'avoir accepté la responsabilité et la tâche, j'espère enrichissante, d'évaluer mon travail.

Merci enfin aux innombrables autres personnes qui ont contribué, directement ou indirectement, de près ou de loin, à l'accomplissement de ce travail mais que je ne peux toutes citer.

Contents

List of Figures

List of Tables

Notations

R Image resolution, typically $R = W \times H$ for an image of width $W \in \mathbb{N}^{\star}$ and height $H \in \mathbb{N}^{\star}$

$\mathcal{P}$ The set of pixel locations in an image of resolution R

$\mathbf{p} \in \mathcal{P}$ A pixel location in an image of resolution R

$\mathbf{M} \in \mathcal{V}^R$ A matrix of dimensions R whose values are in $\mathcal{V}$, *i.e.* $\forall \mathbf{p} \in \mathcal{P}$, $\mathbf{M_p} \in \mathcal{V}$. For example, if $\mathbf{M}$ is a RGB image of width W and height H, then $\mathcal{V} = \mathbb{R}^3$ and $R = W \times H$. If $\mathbf{M}$ is a binary image of resolution $W \times H$, then $\mathcal{V} = \{0, 1\}$ and $R = W \times H$.

$\mathbf{M} > \alpha$ A matrix $\mathbf{M} \in \mathcal{V}^R$ binarized using the threshold $\alpha \in \mathcal{V}$. $\forall \mathbf{p} \in \mathcal{P}, (\mathbf{M} > \alpha)_{\mathbf{p}} = 1$ if $\mathbf{M_p} > \alpha$ else 0. As a result, $(\mathbf{M} > \alpha) \in \{0, 1\}^R$.

$\mathcal{C}$ A set of instance candidates resulting from an image segmentation, *i.e.* $\mathcal{C} \subset \{0, 1\}^R$

$(\mathcal{C}, \geqslant)$ A set of instance candidates equipped with an order relation, *i.e.* there exists a function $s : \mathcal{C} \to \mathbb{R}$ such that: $\forall \mathbf{C}, \mathbf{C}' \in \mathcal{C}$, $s(\mathbf{C}) \geqslant s(\mathbf{C}')$ or $s(\mathbf{C}') \geqslant s(\mathbf{C})$. If $\mathbf{C}^{\star} = \max(\mathcal{C}, \geqslant)$ then $\forall \mathbf{C} \in \mathcal{C}$, $s(\mathbf{C}^{\star}) \geqslant s(\mathbf{C})$.

$|\mathcal{S}| \in \mathbb{N}$ The number of elements of a set $\mathcal{S}$

$\{\mathbf{M}_k\}_k$ Short designation for a set $\{\mathbf{M}_k\}_{k=1}^{N}$ of N elements, without explicitly specifying the number of elements. If $\mathcal{M} = \{\mathbf{M}_k\}_k$, then $|\mathcal{M}| = N$.

$\otimes$ The XNOR operation, *i.e.* $\forall x, x' \in \{0, 1\}$, $x \otimes x' = 1$ if $x = x'$ else 0. If $\mathbf{M}, \mathbf{M}' \in \{0, 1\}^R$, then $\forall \mathbf{p} \in \mathcal{P}$, $(\mathbf{M} \otimes \mathbf{M}')_{\mathbf{p}} = \mathbf{M_p} \otimes \mathbf{M}'_{\mathbf{p}}$.

Glossaries

FCN   Stands for Fully Convolutional Network, *i.e.* a deep convolutional network composed only of convolutional layers, typically for pixel classification tasks

Training   Refers to the phase during which the parameters of a FCN are optimized to approximate a user-defined function from annotated samples

Inference   Refers to the output of a trained FCN

Mikado   Refers to our synthetic data

Bicameral   Refers to the proposed network architecture. Comes from the latin words *bis* (twice) and *camera* (chamber).

Visual Representations
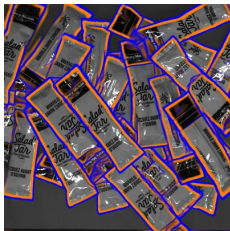
The instance boundaries (blue) and the occluding boundary sides (orange). Shortly, referred to as boundaries and occlusions respectively.
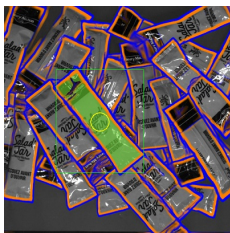
In addition to the boundaries and occlusions, the bounding box (green rectangle) and pixels (colored region) of the most affordable instance, and the corresponding grasp (yellow circle)
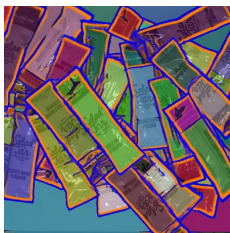
In addition to the boundaries and occlusions, the pixels (colored regions) of the most affordable instance candidate and other instance candidates. The colors are **randomly** distributed.

Chapter 1

Introduction

Robots have become the most versatile human tools in the sense that they can help or replace humans for increasingly complex real-world interactions. The joint ever-growing amount of data and hardware capability have enabled computers equipped with learning algorithms to achieve human-level performances in a wide range of applications, such as detecting and manipulating novel objects in unknown environments.

In collaboration with Siléane[1], a French company in industrial robotics, we aim in this context at generalizing the robotization of an industrial task for which humans are unfit because of the unbearable repetitiveness, the hazardous environments in which the task is executed, or the superhuman skills that are required. Namely *random bin-picking*, this task consists in unpiling object instances to feed automated lines. Current automation strategies resort to strong priors on the object or the gripper, thereby preventing any generalization across many reference products although the notion of instance is object category and gripper-agnostic.

We thus propose a generic learning-based solution towards object model-free and large-scale applications. Specifically, we address two main questions:

- **How to best learn generalizable representations for object-oriented bin-picking without explicit object and gripper models?**

- **As collecting annotated data in such application domains is unfeasible, can one leverage synthetic training data?**

In this chapter, we further detail our motivations and objectives, with respect to the difficulties encountered in the real-world applications and the challenges raised by our application context.

---

[1]www.sileane.com

1.1   Application Context

In the context of industrial robotics, automated lines in various applications, such as kitting, component assembling, order processing, or waste sorting, require to be continuously fed on manufactured products or waste materials. For practical reasons, these objects are conveyed to the line entry point piled up in bulk in containers. *Robotic random bin-picking*, shortly bin-picking, which consists in unloading object instances piled up in bulk one by one using a robotic arm, is thereby a key step towards fully automated lines. Automating such a task however raises a number of application-dependent difficulties:

- A pile of objects can be either *homogeneous*, *i.e.* containing many instances of the same object, or *heterogeneous*, *i.e.* one or many instances of many different objects, as illustrated by Figure 1.1.

Figure 1.1: Examples of piles. Objects can be low or highly textured, piled up in homogeneous or heterogeneous configurations.

- In both homogeneous and heterogeneous cases, automated lines may require hundreds of different reference products to build complex systems or supply the need of user-driven customization.

- Object instances may present some intra-class variations, typically due to elastic deformations or design variations (see Figure 1.2).

- Some object geometry may induce piles prone to strong occlusions and entanglements between instances. As a result, attempting to extract an occluded instance penalizes the robot operating cycle, and consequently the overall yield of a line.

Geometric deformations   Texture variations

Figure 1.2: Examples of intra-class variations in real-world applications. In such cases, a bin-picking approach based on the explicit object model is hardly appropriate.

- Prior knowledge on the objects to unpile may be limited. For example, in the automotive industry, CAD models are not always available because the manufactured components come from many different suppliers. In applications such as food packaging or waste sorting, explicit models are simply non-existent.

- A scene reconstruction, for example by active stereovision or laser triangulation, is not always feasible either by lack of visual information or because high-throughput applications drastically constrain the time frame for image processing.

- Annotated real-world data is hardly collectable as it is a time-consuming and tedious task, impractical in industrial environments.

Further to these observations, we specifically address the case of high-throughput open-loop bin-picking **applications for which an explicit model of the target objects is not available** and a scene reconstruction is not always feasible. These conditions are met in various application domains such as car assembly, food packaging, order processing or waste sorting. We also consider that collecting real-world data, annotated or not, for offline learning is not possible.

1.2   Objectives

In this application context, we aim at developing a **learning-based solution for locating the most affordable instances of a pile** independently of the object and gripper models (see Figures 1.3 and 1.5). We look for learning a generic notion of *affordable instance*, that enables many line entry points to be automatically supervised by a unique

(1) Data acquisition   (2) Instance detection   (3) Instance extraction
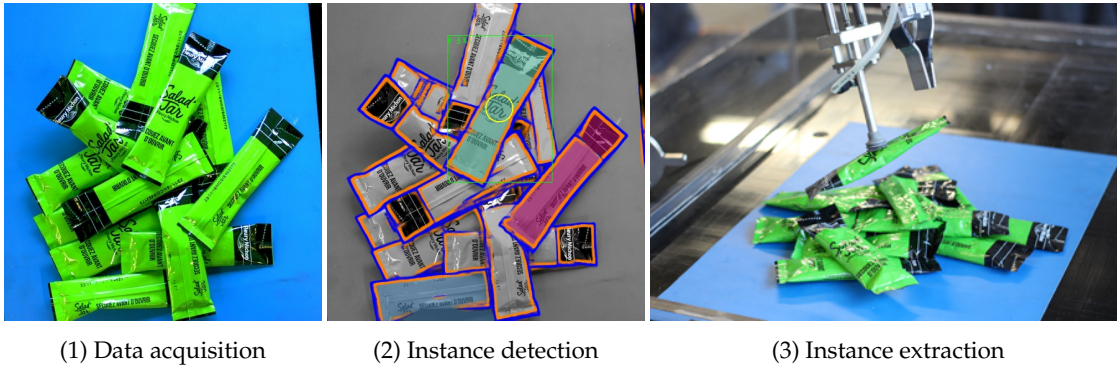
Figure 1.3: Open-loop cycle in the proposed approach, built on an offline supervised synthetic training to detect the most affordable instances of a pile from a single image, independently of the object and gripper models.
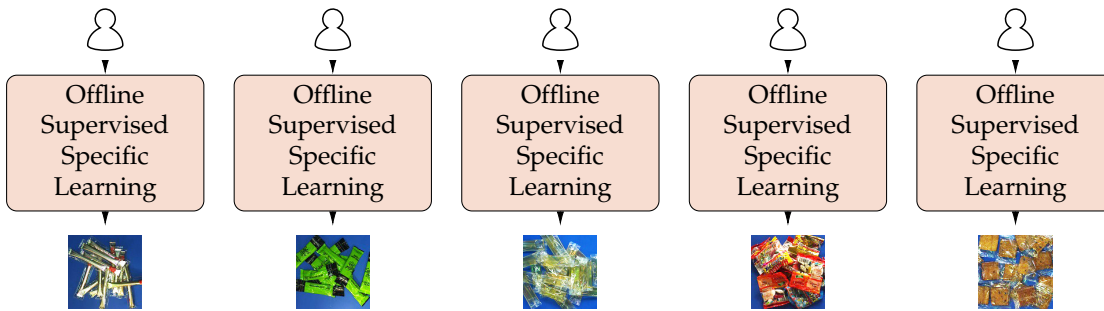
(a) **Current industrial approach**
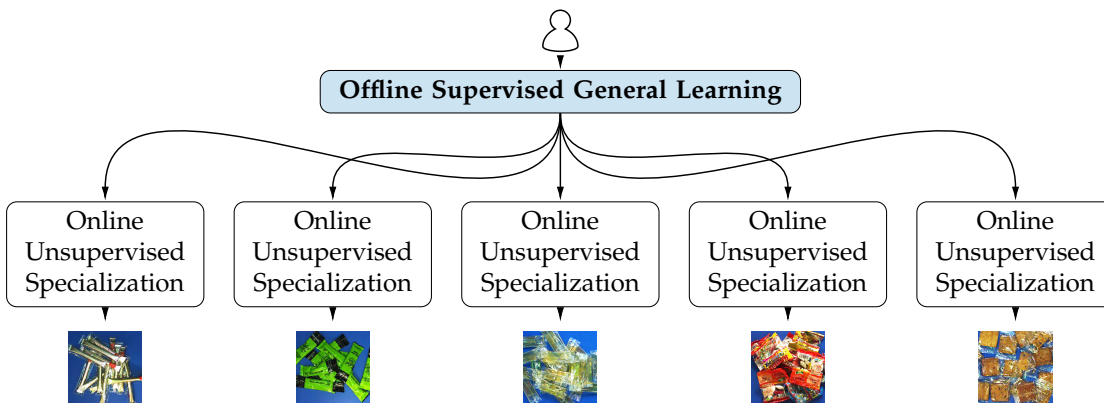
(b) **Our long-term view beyond this work**

Figure 1.4: Our long-term view compared with the current industrial approach. We advocate a unique supervised general learning (this work) that initializes each line entry point to a strong baseline, which is further specialized online.

model (*c.f.* Figure 1.4). Such a global model has many benefits, including reduced maintenance, effective pooling of resources and skills, and scalability. As short-term results are expected from this work in collaboration with an industrial company, we assume the existence of product families for solving practical real-world scenarios. **Specifically, our contributions are driven by the scenario of unpiling sachets in homogeneous bulk, using only RGB images as input data.** Our results nevertheless suggest that the proposed approach can apply to other product families, using alternative input modalities as well.

1.3 Contributions

Our contributions in addressing the problem of object-oriented bin-picking without explicit object models nor real-world data are two-fold:

- a novel deep fully convolutional learning architecture, referred to as *bicameral*, for jointly inferring the instance boundaries and occlusions from a single image;

- a simulation-based scheme for application to real-world robotic setups, referred to as *Mikado*, supported by an extensive experimental evaluation.

Specifically, unlike the state-of-the-art approaches which rely on two independent streams whereas occlusions cause boundaries, the proposed architecture enables a unified representation of the boundaries and occlusions between instances, thus achieving state-of-the-art performances on both synthetic data and public real-world datasets for instance boundary and occlusion detection.

The proposed simulation-based scheme then enables to synthetically train such a bicameral network for real-world bin-picking applications while outperforming the object model-free industrial baseline.

1.3.1 Publications

Most of this work is confidential due to direct industrial applications in collaboration with Siléane, but nevertheless led to a journal paper (in revision) and a workshop paper:

- [67] Grard et al., **Bicameral Structuring and Synthetic Imagery for Jointly Predicting Instance Boundaries and Nearby Occlusions From a Single Image.** Submitted to *International Journal of Computer Vision (IJCV), Special Issue on Deep Learning for Robotic Vision* in July 2018. First revision in January 2019.

Single real-world image

Bicameral Deep Fully Convolutional Network

Boundaries and occlusions

Boundary-Based Clustering & Occlusion-Based Ranking

Most affordable instance

Mikado Simulator

Synthetic training images of piles

Inputs:
Mesh template
Texture images   +

Physics simulation of piles of sachets

Top-view RGB-D camera rendering

Training and test data preparation

Figure 1.5: Overview of the proposed approach. A synthetically trained fully convolutional network jointly learns to detect boundaries and occlusions between instances, independently of the object and gripper models. The coordinates of the most affordable instance centroid is then deduced from the network inference and sent to the robot.

- [66] Grard et al., **Object Segmentation in Depth Maps with One User Click and a Synthetically Trained Fully Convolutional Network.** In *International Workshop on Human-Friendly Robotics (HFR)*, 2017. `https://doi.org/10.1007/978-3-319-89327-3_16`

[67] presents our performance-enhancing fully convolutional encoder-decoder network for oriented instance boundary detection and the benefits of jointly using synthetic data for this task.

[66] presents through a practical interactive application one of our core ideas that boundaries between instances piled up in bulk are better detected by postponing instance localization after instance delineation, and leveraging the duality between boundary detection and instance segmentation.

1.4   Contents

This manuscript is organized as follows:

In Chapter 2, we briefly introduce the mechanisms of deep learning, with a focus on the related mathematical tools that we extensively use in the following chapters.

In Chapter 3, we review the state of the art on the different approaches for bin-picking. We largely focus on the state of the art for generic instance segmentation, *i.e.* category-agnostic instance delineation, which constitutes a core topic of our approach.

In Chapter 4, we lay out our contributions on occlusion-aware generic instance segmentation. Specifically, we present and evaluate the proposed network architecture on the Mikado synthetic data and state-of-the-art real-world datasets. Furthermore, we conduct an ablation study of the bicameral architecture to better characterize its components. We finally describe how to translate the bicameral network inference into grasp coordinates on the most affordable instance of a pile.

In Chapter 5, we present our contributions on applying the proposed occlusion-aware instance segmentation approach to bin-picking. Specifically, we describe a training data generation pipeline to synthetically train a bicameral network for real-world bin-picking applications. We show the plausibility of the proposed synthetic training data, then extensively evaluate our synthetically trained models on a real-world robotic setup to demonstrate the applicability of our method in industrial bin-picking conditions.

In Chapter 6, we summarize our work and contributions, then draw some directions for future work.

Chapter 2

Deep Convolutional Networks

In this chapter, we briefly present deep convolutional networks (DCN) with a focus on the related mathematical tools that we extensively use in the next chapters. More precisely, we describe the state-of-the-art implementation of DCNs, their benefits and limitations. We also shortly introduce some state-of-the-art interpretations to better understand and explain their properties. For more detailed explanations on the general mechanisms of deep learning, we refer the reader to [62, 99].

2.1 Introduction

Weights

$x_1 \circ\!\!\longrightarrow w_1$

Activation function

Output

$x_2 \circ\!\!\longrightarrow w_2 \longrightarrow \Sigma \longrightarrow \boxed{\max(0, \cdot)} \longrightarrow y = \max(0, \sum_{i=1}^{3} w_i x_i + b)$

Inputs

$x_3 \circ\!\!\longrightarrow w_3$

Generally, $y = \max(0, W^\top x)$

$1 \circ\!\!\longrightarrow b$
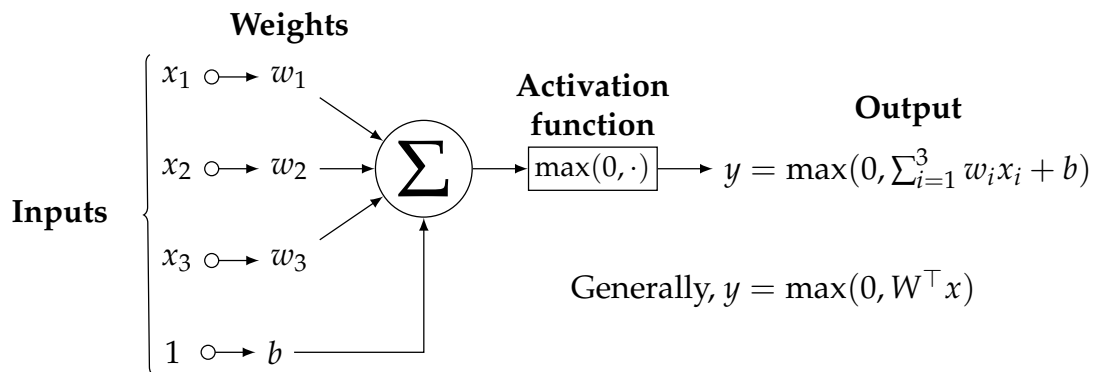
Figure 2.1: A three-input unit of a deep neural network, namely *neuron*, using a rectified linear unit (ReLU) as activation function

2.1.1 Definitions

A *deep neural network* is a computational learning structure of interconnected nodes, called *neurons*, aimed at encoding some input tensor(s),

11

such as images, audio signals, videos, in a discriminative and linearly separable representation space for classification or regression tasks [62, 99]. Most generally, neurons are arranged in *layers*, such that a layer's neuron input is connected to all the previous layer's neuron outputs. In such a configuration, layers are said *fully connected*. For example, in the case of images, each pixel would be a neuron input, and a top layer's neuron would then be connected to each pixel. As illustrated by Figure 2.1, each neuron is a simple two-step operation: an affine transformation of the input values to which a non-linear *activation* function is applied, such as the so-called *rectified linear unit* (ReLU) [98] defined as $\max(0, \cdot)$. The *parameters*, also referred to as *weights*, of these linear operators are determined by non-convex optimization. In state-of-the-art applications, a deep network may contain billions of parameters structured in hundreds of layers [87].

Bottom   Top   Bottom   Top   Bottom   Top

Fully connected   **Locally connected**   **Convolutional**
                              Receptive field   *Receptive field &*
                                                         Parameter sharing
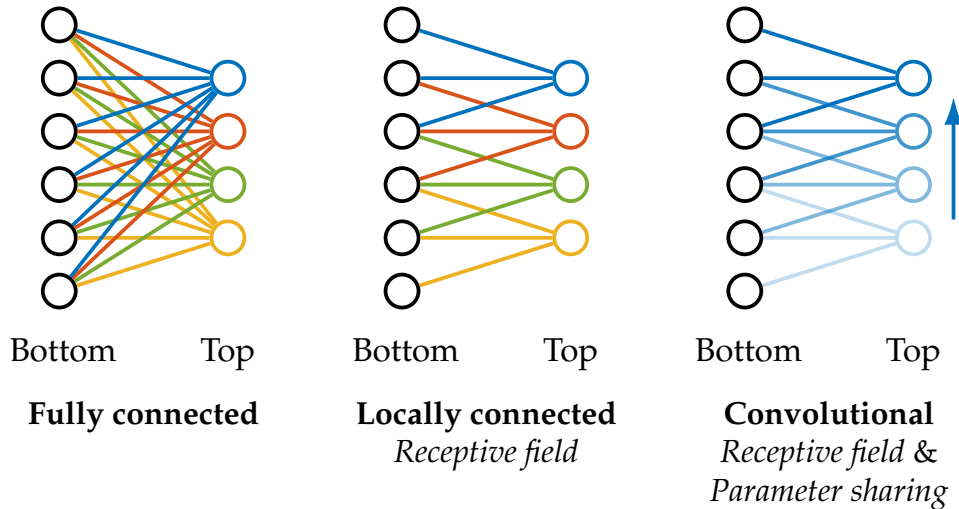
Figure 2.2: Example of a 4-neuron top layer turned into a 1-neuron convolutional layer by two architectural priors: local neuron receptive fields and in-layer parameter sharing. A 1-neuron convolutional layer thus results in a neuron moving across the whole bottom layer. State-of-the-art convolutional layers are sets of moving neurons, each accounting for a filter.

Deep convolutional networks (DCNs) are deep neural networks specific to multi-channel images, introduced in [101]. As images contain millions of pixels and spatially recurrent patterns, using a "fully connected" neural network as it is for encoding images is intractable and inefficient. Deep convolutional networks thus present two key architectural priors: reduced neuron *receptive fields* and in-layer *parameter sharing* (*c.f*. Figure 2.2). First, each neuron is connected to only a few spatially neighboring neurons of the previous layer. Second, in a layer, all the neurons share the same parameters. This equivalently results

in one neuron per layer moving across the whole image. Introducing these priors has many benefits: reduction of the overall number of parameters, better generalization capability, translation invariance. Layers with such priors are then qualified as *convolutional* because they can be interpreted as convolution filters with learnable kernel. In state-of-the-art applications, deep convolutional layers are structured in sets of convolutional filters, interleaved with *max-pooling* layers that aggregate spatially neighboring features by element-wise max operators. As illustrated by Figure 2.3, an image fed-forward through a convolutional encoder thus results in deep three-dimensional *feature maps* or *representations*, that encode the image semantics at different resolutions.
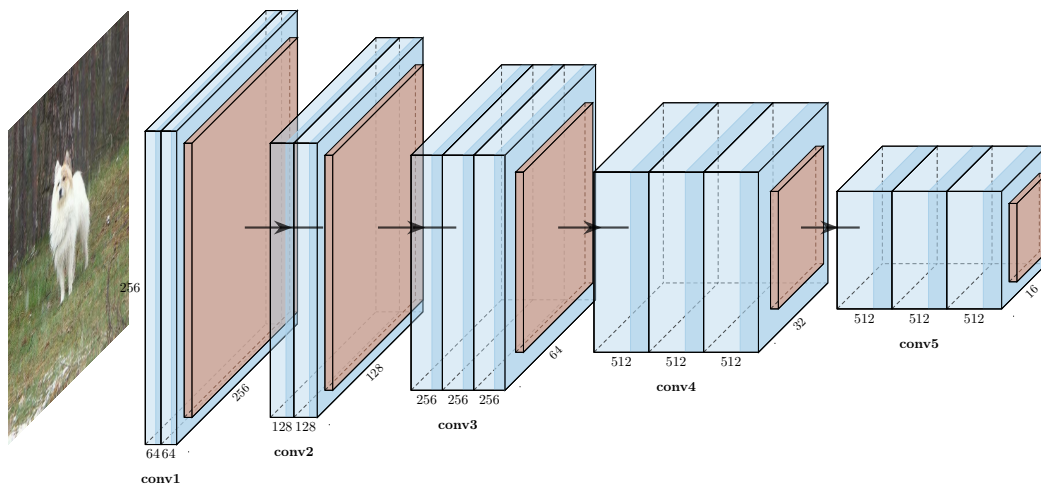
Figure 2.3: Overview of a VGG16-based [168] convolutional encoder (best viewed in color). Convolutional representations are in blue, element-wise activation units are in red. Arrows indicate max-pooling operators in 2×2-neighborhoods.

Fully convolutional networks (FCNs) are deep convolutional networks that contain only convolutional layers, typically aimed at classifying each pixel independently, instead of the whole image with one or a few image-level labels.

Encoder-decoder networks are FCNs designed for recovering high-resolution pixel label maps. As previously illustrated in Figure 3.14, such networks typically introduce a convolutional *decoder* aimed at gradually recovering a full spatial resolution by interleaving convolutional and unpooling layers from the encoder's latent representations.

2.1.2 Learning

A network is aimed at approximating a function $f : \mathcal{X} \to \mathcal{Y}$, $\mathbf{x} \mapsto \mathbf{y}$
from a finite set $\{(\mathbf{x}_n, f(\mathbf{x}_n) \in \mathcal{X} \times \mathcal{Y}\}_{1 \leqslant n \leqslant N}$ **of** $N \in \mathbb{N}^\star$ **annotated**
samples. For example, in binary classification using FCNs, we want
to assign a label $y \in \{0, 1\}$ to a RGB pixel $x \in \mathcal{X} = \mathbb{R}^3$. A trained
network is thus a generalizable function $f_\mathbf{W} : \mathcal{X} \to \mathcal{Y}$, $\mathbf{x} \mapsto \hat{\mathbf{y}}$, using
the learnable parameters $\mathbf{W} \in \mathbb{R}^Q$, where $Q \in \mathbb{N}^\star$ is the overall number
of parameters. $f_\mathbf{W}$ is defined by $\mathbf{W}$, but also a number of architectural
choices: the number and depth of convolutional layers; how the layers
are interconnected; the type of activation functions. As the search space
for determining the optimal architecture is infinite, design choices
are made by intuitions from state-of-the-art network architectures [75,
87, 98, 101, 168, 174] and empirical results. Beyond the scope of this
work, recent works explored tractable optimization-based strategies for
addressing this concern [110, 209], which nevertheless remains an open
research field.

Training a network means determining the network parameters that
best minimize an application-dependent loss function and jointly en-
able a strong generalization. It consists in iteratively updating the
weights by stochastic gradient descent (SGD) using the backpropaga-
tion mechanism [98, 100]. Basically, a training iteration is a two-step
action: an inference-like *forward pass* of a batch of training images; a
backward pass of the corresponding gradients for updating the weights
with respect to the gradient norm and direction, roughly according to
the following update equation:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha \nabla \mathcal{L}(\mathbf{W}_t) \tag{2.1}$$

where $\mathbf{W}_t$ and $\mathbf{W}_{t+1}$ are the network parameters at iterations t and $t + 1$
respectively, $\mathcal{L}(\mathbf{W}_t)$ the loss function to minimize, and α the *learning*
rate that controls how much the weights are adjusted at each iteration.
A low learning rate induces a slow move along the downward slope. In
typical SGD, the learning rate is a global single-valued hyperparameter.
We won't go much further on this topic as it is already extensively
covered in the literature [62, 178]. Note that, in such an optimization,
both the initialization and the optimization path have great importance
for reaching the best performances [194]. Different parameter update
strategies [49, 95, 173, 178, 199] have been proposed for a faster and
more stable convergence. Notably, the *Adam* solver [95] combines
two extensions [49, 178] of SGD: per-parameter learning rates [49] and
weight updates based on moving moments of the recent past gradients
[178]. Additional components have also been commonly adopted to
avoid overfitting during training: the *dropout* mechanism [169], which

consists in randomly shutting off some parameters during training, and the introduction of a regularization term in the loss function, typically the ℓ_2-norm of the weights.

Formally, we look for the network parameters $\mathbf{W}^\star = \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W})$ using the gradient-based Adam method [95]:

$$(\mathbf{W_{t+1}})_i = (\mathbf{W_t})_i - \alpha \frac{\sqrt{1 - (\beta_2)_i^t}}{1 - (\beta_1)_i^t} \frac{(\mathbf{m_t})_i}{\epsilon + \sqrt{(\mathbf{v_t})_i}} \qquad (2.2)$$

where $\epsilon > 0$ avoids division by zero, $\mathbf{m_t}$, $\mathbf{v_t}$ are estimated moments of the recent past gradients, and β_1, β_2 hyper-parameters controlling the decay rates of these moving moments as follows:

$$\begin{aligned}(\mathbf{m_t})_i &= \beta_1(\mathbf{m_{t-1}})_i + (1 - \beta_1)(\nabla\mathcal{L}(\mathbf{W_t}))_i \\ (\mathbf{v_t})_i &= \beta_2(\mathbf{v_{t-1}})_i + (1 - \beta_2)(\nabla\mathcal{L}(\mathbf{W_t}))_i^2\end{aligned} \qquad (2.3)$$

We minimize the following *regularized cross-entropy* loss function:

$$\mathcal{L}_{RCE}(\mathbf{W}) := \underbrace{-\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{C} (\mathbf{y}_n)_i \log(\hat{\mathbf{y}}_n)_i}_{\text{objective}} + \underbrace{\lambda \|\mathbf{W}\|_2^2}_{\text{regularization}} \qquad (2.4)$$

where λ is another hyper-parameter, called *weight decay*, that controls the trade-off between the objective and regularization terms. The objective term aims to minimize the information loss caused by approximating the training data distribution (see Section A.1 for more details). The regularization term enables robustness to noise and prevents from overfitting the training data. Note that the cross-entropy is not a distance because the symmetry condition is not met.

In the case of binary pixel classification, our topic of interest, $\mathcal{Y} = \{0, 1\}^2$. Let $(\mathbf{x}, (y_1, y_2)) \in \mathcal{X} \times \mathcal{Y}$ be a training sample. For variable reduction, we can define $z := y_1$ and $\hat{z} \in [0, 1]$ the corresponding network's estimate. Given that $y_1 + y_2 = 1$, Equation 2.4 can be rewritten as a *regularized binary cross-entropy* loss function:

$$\mathcal{L}_{RBCE}(\mathbf{W}) := \mathcal{L}_{BCE}(\mathbf{W}) + \lambda \|\mathbf{W}\|_2^2 \qquad \text{where}$$
$$\mathcal{L}_{BCE}(\mathbf{W}) := -\frac{1}{N} \sum_{n=1}^{N} (z_n \log(\hat{z}_n) + (1 - z_n) \log(1 - \hat{z}_n)) \qquad (2.5)$$

Bayesian Interpretation   Note that, from a Bayesian perspective, the network inference $\hat{z} = f_{\mathbf{W}}(\mathbf{x}) \in [0, 1]$ from Equation 2.5 can be viewed as a Maximum A Posteriori (MAP) probability estimate with a Gaussian prior on the weights (see Section A.2 for a formal proof).

2.2   Limitations

In Section 2.1, we introduced FCNs and their training for binary pixel classification to later build our solution. For fully understanding the limitations of our method, we need to understand the limitations of FCNs. We therefore lay out their limitations in this section.

2.2.1   Large Training Datasets

Application-driven studies have shown that FCNs capture the invariants of a training dataset. For example, a FCN learns to recognize apples by extracting the attributes shared by training examples of apple. As a consequence, if the training dataset is biased by unbalanced distributions or inconsistent ground-truth annotations [181], then the FCN captures these biases as well [180]. In our example, if the apples depicted in the training examples are all red, then the FCN learns the red color as an invariant. Large training datasets prove therefore more suitable for training FCNs as their data distribution is less likely to be biased. Moreover, as FCNs may contain billions of parameters, a large training dataset partly prevents from overfitting. This is problematic because the training samples must be annotated, which is tedious and time-consuming for learning pixel classification from real images. In Chapter 5, we address this limitation by synthetically training the proposed FCN. Note that many research works also concern the unsupervised training of FCNs but the resulting performances remain lower than those obtained by supervised training.

2.2.2   Limited Internal Representations

Lack of Invariance and Non-Equivariance   Due to their built-in convolutional nature, FCNs are translation invariant. As a result, a context-free pattern is internally represented independently of its image location. For example, in object detection on RGB images, a FCN can detect multiple non-occluded objects in an image by learning only from single-object image patches. A FCN is however not scale-invariant, nor rotation-invariant. Such invariances can be captured only by accordingly augmenting the training images with geometric transformations. More generally, FCN representations are not scale equivariant, nor rotation equivariant. In other words, rotating a pattern representation is not equivalent to representating the rotated pattern. Note that [160] recently addressed this concern by introducing "capsule" networks, which enable to intrinsically learn pose-equivariant representations. Such networks however remain prospective learning architectures.

| (a) Fake images [135] | (b) Adversarial example [63] |
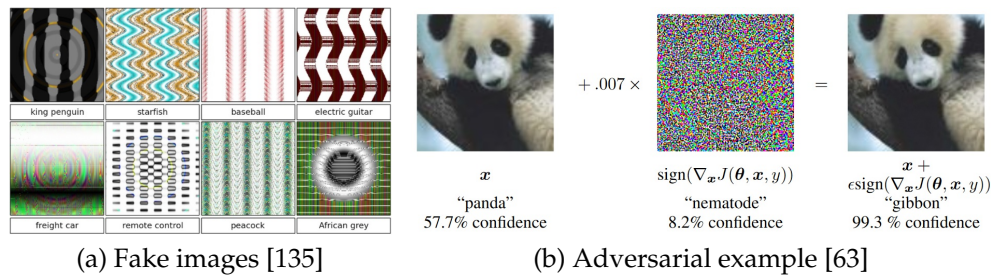
Figure 2.4: DCNs and humans recognize objects differently. (a) Images that are unrecognizable to humans but classified by a DCN as familar objects with $\geqslant 99,6\%$ certainty [135]. (b) Adding an imperceptibly small vector, whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, changes a DCN's classification [63].

Figure 2.5: Can you see the duck? A deep network trained for object boundary detection can be easily fooled by deceptive details.

Semantic Encoding or Not? A deep network inference looks impressive because human-level concepts seem to be "understood". [63, 135] however showed that DCNs can be easily fooled, as illustrated in Figure 2.4. Evolved fake images that are unrecognizable to humans can be classified by a DCN as familiar objects with $\geqslant 99,6\%$ certainty [135]. Adding an imperceptibly small vector, whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, can also change a DCN's classification [63]. Our own experiments on object boundary detection confirm this weakness, as shown by Figure 2.5.

2.2.3   Lack of Explanability

Although DCNs have shown remarkable performances and consequently become state-of-the-art for a plethoric range of applications,

they remain poorly explained because of the complex entanglement between their internal linear operators and non-linearities. Training by non-convex optimization also enables a very convenient versatility, but at the cost of no theoretical guarantees on the success of learning, unless performing resource-intensive experiments. Fundamentally, three basic questions remain unanswered by theory:

- If we use an existing training dataset, what is the best network architecture for the given training data distribution?

- If we use a training data generator, what is the best training data distribution for a given network architecture?

- What are the hyperparameters for best training a given network on a given training data distribution?

Despite these limitations, we argue that using DCNs is a reasonable and effective choice for learning generalizable representations, in light of the state-of-the-art insights and interpretations.

2.3 Interpretations

Deep representations prove the most effective and versatile solution in many applications because they enable to extract high-level concepts from end-to-end training on massive data. Specifically, empirical studies have shown that the representations learned by FCNs are multi-scale and strongly generalizable to unseen samples. In this section, we briefly lay out the state-of-the-art empirical analysis of deep representations, and the recent mathematical frameworks that can explain these observations.

2.3.1 Hierarchical Representations

Visualizing the intermediate feature maps of a DCN reveals that the learned representations are hierarchical [200]. As depicted in Figure 2.6, the top representations of a convolutional encoder convey high-level concepts built from the combination of lower-level features encoded by the bottom layers. For example, in face recognition applications, texture invariants such as the skin and eyes color are first encoded in the first layers, then larger parts like the nose, lips, eyes in the mid-level representations, and finally faces as combination of these parts. Furthermore, such hierarchical features consistently transition from general to specific by the last layers [194]. In particular, the bottom convolutional filters are typically Gabor-like kernels that embed strongly generalizable local representations.
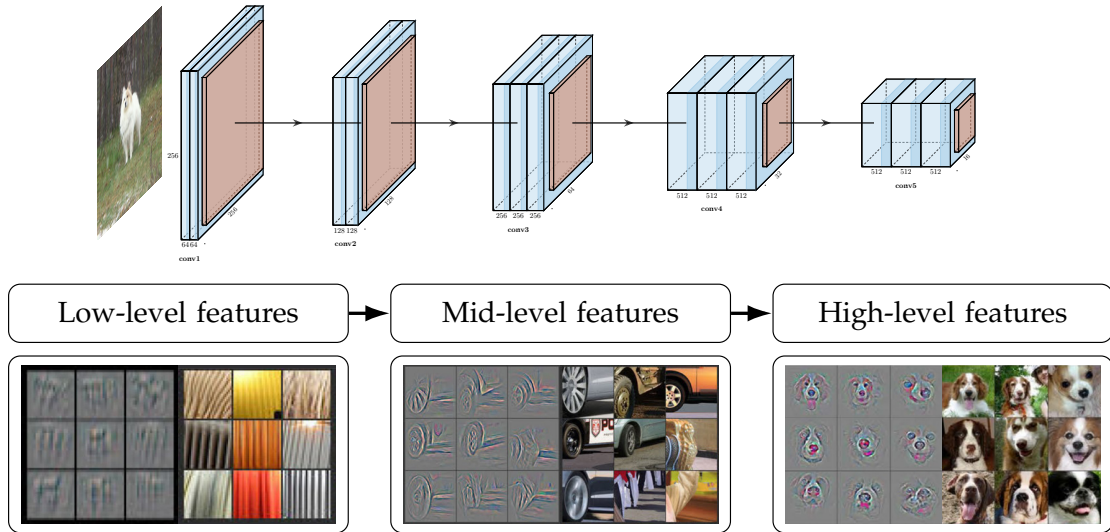
Figure 2.6: Intermediate activations of a DCN trained for object categorization, along with the corresponding image patches. The top representations convey high-level concepts resulting from the combination of lower-level features in bottom layers [200].

2.3.2   Kernel Perspective

DCNs can be viewed as elements of a Reproducing Kernel Hilbert Space (RKHS) of functions [19]. In such a perspective, there exists a positive definite kernel K that defines a RKHS $\mathcal{H}$ of functions from $\mathcal{X}$ to $\mathbb{R}$, along with a mapping $\phi : \mathcal{X} \to \mathcal{H}$. A function $f_{\mathbf{W}}$ in this RKHS can then be written in linear form such that:

$$\forall \mathbf{x} \in \mathcal{X}, \ f_{\mathbf{W}}(\mathbf{x}) = <f_{\mathbf{W}}, \phi(\mathbf{x})>_{\mathcal{H}} \tag{2.6}$$

Consistently with many application-driven studies showing empirically that a DCN captures multiscale invariants, $\phi(\mathbf{x})$ is a data representation that can prove invariant to the action of groups of transformations, such as translations and diffeomorphisms [124]. Interestingly, Equation 2.6 induces that $f_{\mathbf{W}}$ is $\|f_{\mathbf{W}}\|_{\mathcal{H}}$-Lipschitz continuous:

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}, \ |f_{\mathbf{W}}(\mathbf{x}) - f_{\mathbf{W}}(\mathbf{x}')| \leqslant \|f_{\mathbf{W}}\|_{\mathcal{H}} \left\|\phi(\mathbf{x}) - \phi(\mathbf{x}')\right\|_{2} \tag{2.7}$$

thus showing that $\|f_{\mathbf{W}}\|_{\mathcal{H}}$ acts as a regularizer [19]. For example, let $f_{\mathbf{W}}$ be a network trained to distinguish apples from cookies. As illustrated by Figure 2.7, lessening $\|f_{\mathbf{W}}\|_{\mathcal{H}}$ enables to better control the network predictions by the corresponding representations.
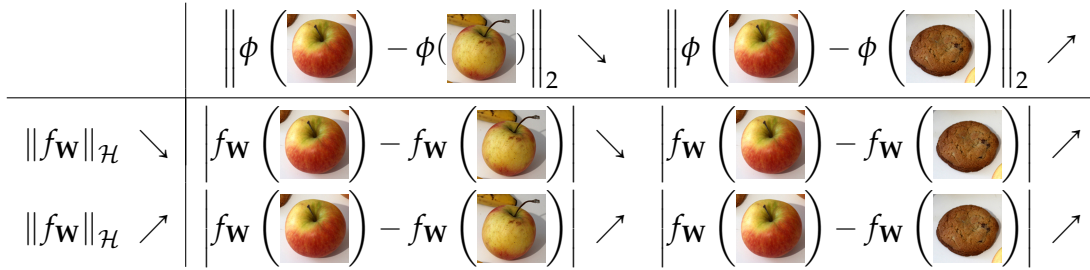
Figure 2.7: Variations of $|f_{\mathbf{W}}(\mathbf{x}) - f_{\mathbf{W}}(\mathbf{x}')|$ with respect to the variations of $\|f_{\mathbf{W}}\|_{\mathcal{H}}$ and $\|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|_2$ (c.f. Equations 2.6 and 2.7), assuming that $f_{\mathbf{W}}$ is trained to distinguish apples from cookies. From a kernel perspective, the norm $\|f_{\mathbf{W}}\|_{\mathcal{H}}$ of a deep function $f_{\mathbf{W}}$ acts as a regularizer [19]: predictions are better subordinated to representations by lessening $\|f_{\mathbf{W}}\|_{\mathcal{H}}$.

Although the quantity $\|f_{\mathbf{W}}\|_{\mathcal{H}}$ is not analytically computable, $\|f_{\mathbf{W}}\|_{\mathcal{H}}$ can be controlled by an upper bound [19]:

$$\|f_{\mathbf{W}}\|_{\mathcal{H}} \leqslant \omega(\|\mathbf{W}_1\|, ..., \|\mathbf{W}_L\|) \tag{2.8}$$

where ω is increasing in all of its arguments and $\|\mathbf{W}_k\|$ is the spectral norm of the kth linear operator $\|\mathbf{W}_k\|$. In addition to providing theoretical insights on deep convolutional models, this mathematical formalization has led to promising direct applications [40, 132, 163]. Specifically, as suggested by Equation 2.8, controlling the spectral norm of the convolutional layers thereby enables a better generalizability [163], a more stable discriminator for generative adversarial networks [132], and a better robustness to adversarial examples [40].

2.4   Conclusion

Deep fully convolutional networks (FCNs) are layered connectionnist systems, entangling linear operators and non-linearities, for **learning generalizable hierarchical image representations**. Specifically, **a FCN captures the invariants of a training dataset**. As a result, if a dataset is biased, then the FCN learns these biases as well. Moreover, the learned representations are not intrinsically invariant, nor equivariant, to rotations and scaling. Nevertheless, FCNs enable to **learn high-level concepts, such as the notions of instance and occlusion, from low-level data, and achieve state-of-the-art performances in many applications**. The FCN parameters are learned by non-convex optimization in a **end-to-end training**, typically driven by a regularized binary cross-entropy loss function for binary pixel classification. There currently exist many deep learning implementations such as [2, 36, 52, 92, 142, 164]. In this work, we use the C++/CUDA library named *Caffe* [92].

Chapter 3

State of the Art

In this chapter, we review the state of the art on open-loop visual-based bin-picking. Specifically, we first describe the different approaches for automating such a task and review the related works. We then review more deeply the state of the art on instance segmentation, from which we design our model-free object-oriented bin-picking approach.

3.1 Bin-Picking

In this section, we review the state-of-the-art approaches for open-loop bin-picking, which is the target application of this work. In such applications, a robot is typically slaved to a vision-based module, in a three-step operating cycle:

1. **Image acquisition:** a visual sensor captures a photometric scene representation. Typical sensors are simple RGB cameras, hyperspectral cameras or more complex 3D sensors that provide a multi-view scene reconstruction as well (*c.f*. Figure 3.1).

| uEye CP, | FX10, | MotionCam 3D, | Ensenso X, | Kinect 2, |
| IDS Imaging | Specim | Photoneo | IDS Imaging | Microsoft |
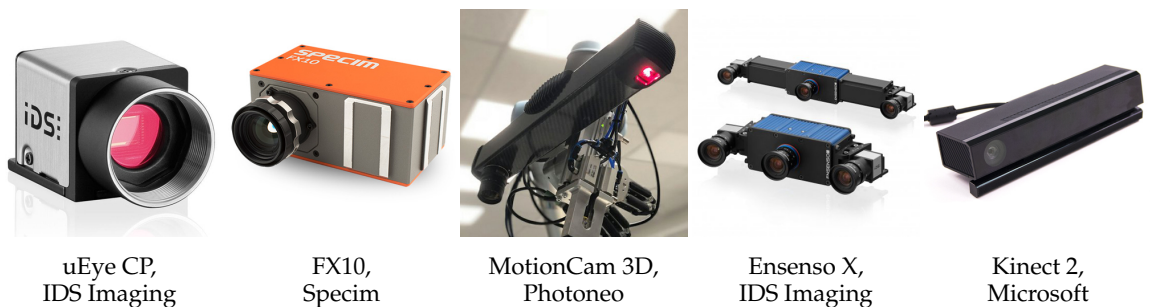
Figure 3.1: Examples of visual sensors. From left to right: RGB camera; hyperspectral camera; laser-based 3D scanner; active binocular system; time-of-flight camera.

2. **Image processing:** coordinates of the next instance to extract are computed from the captured scene representation. The computation time and the relevance of the produced coordinates are key factors to achieve stationnary performances in high-throughput applications. This work aims to play this step.

| LR Mate 200, | Quattro s650H, | TP80, | LBR iiwa, | Sawyer, |
| Fanuc | Adept | Staübli | Kuka | Rethink Robotics |

Figure 3.2: Examples of robotic manipulators

3. **Robot action:** a robotic manuipulator (*c.f*. Figure 3.2) extracts the object at the sent coordinates. The robot end-effector is typically a vacuum suction cup or a parallel jaw gripper, but there exist alternative application-dependent designs (*c.f*. Figure 3.3).

| Vacuum | Parallel jaw | iCub fingered | Soft fingered | Universal |
| suction cup | gripper | gripper [162] | gripper [83] | gripper [28] |

Figure 3.3: Examples of robot end effectors

Vision-based bin-picking can be categorized into two main classes (*c.f*. Figure 3.4): *gripper-oriented* and *object-oriented*.

Gripper-oriented approaches aim to detect grasp hypotheses with respect to the gripper model and physics without any explicit notion of instance. Such approaches prove effective unless the object geometry induces strong occlusions and entanglements between the instances.

Object-oriented alternatively aim to detect object instances independently of the gripper model. Grasp hypotheses are then generated with respect to both the gripper model and the detected most affordable instance, thus considerably reducing the search space. For example, in the common case of flat convex objects such as food packets, the

best grasp hypotheses are likely to be located near the centroid of each unoccluded instance. We advocate this latter approach, that paves the way towards a general framework for instance-aware grasping. Figure 3.5 illustrates how a gripper-oriented search strategy is boosted by our object-oriented approach.
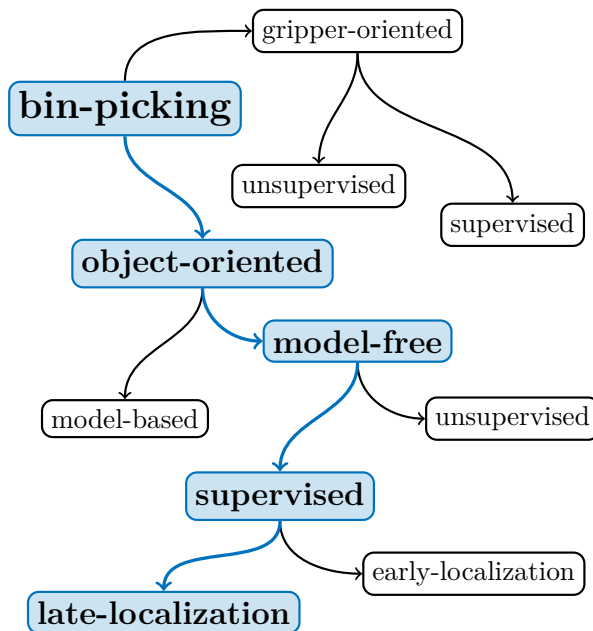
Figure 3.4: Categorization of approaches for vision-based bin-picking. Blue annotations indicate the category of the proposed approach.

(a) Image   (b) Depth map (input)   (c) Gripper-oriented   (d) Object-oriented
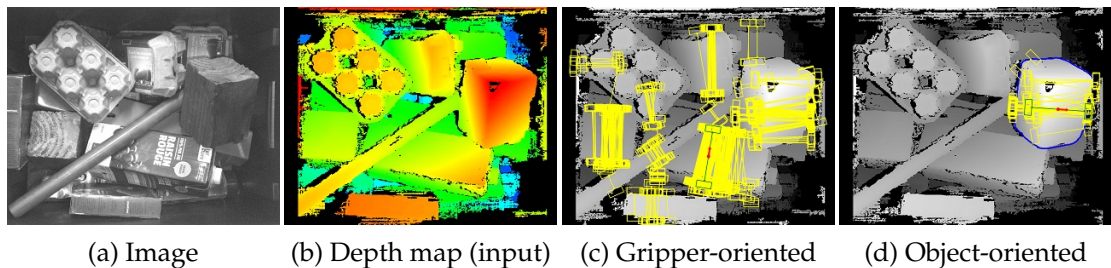
Figure 3.5: Example of a proprietary gripper-oriented algorithm boosted by our object-oriented approach (best viewed in color). Yellow annotations indicate potential grasps for a parallel-jaw gripper. Green annotations indicate the selected "best" grasp. Without the notion of instance, the end effector may be sent to hardly extractable objects (c). Boosting the grasp detection using our object-oriented approach enables instead to focus on the most affordable instance, thus drastically reducing the grasp search space (d).

3.1.1 Gripper-Oriented Bin-Picking

Gripper-oriented bin-picking approaches (see Figure 3.19) consist in detecting grasp opportunities with respect to the robot end-effector physics. Although various grippers have been developed [88], vacuum-suction and parallel-jaw grippers remain most widely used in the industry, and constitute the most studied gripper models in grasp detection. Early grasp detectors employed unsupervised heuristics on depth images to detect and rank either locations where parallel jaws can be best inserted or locally planar areas for a vacuum suction cup [47]. Jointly with more complex analytic gripper models for detection, ranking heuristics-based grasp candidates was boosted by deep convolutional networks (DCNs) [119, 120].

Supervised end-to-end training using DCNs was first explored for single-grasp prediction on single-object images [103]. Inspired by object detection techniques [152, 155], end-to-end training for parallel-jaw grippers was later extended to multi-grasp detection [93, 151], then to multi-object settings by joint classification and regression of predefined grasp templates [39, 205]. Fully convolutional grasp detection was generalized to vacuum suction grippers by inferring instead pixel-wise affordance maps [8, 134, 201], by analogy with semantic segmentation techniques [11, 35]. Training data for these networks was however made from sparse manual annotations [39, 103, 134] or heuristics-based labels [201], whereas the notion of grasp affordance for a robotic gripper may fundamentally differ from a human perspective.
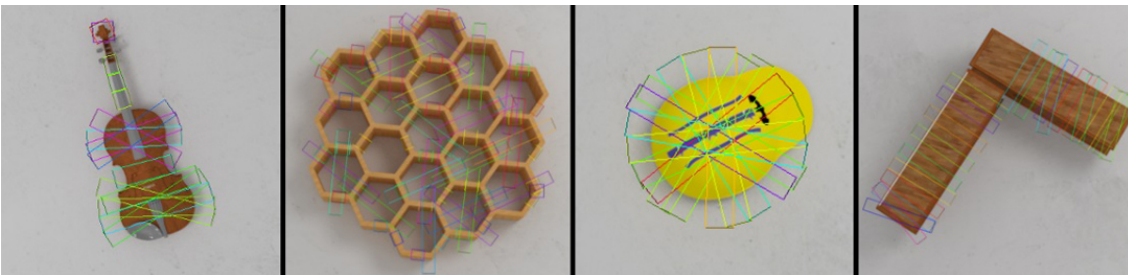
Figure 3.6: Examples of parallel-jaw grasp detections in simulation [44]

As illustrated by Figure 3.6, simulation was thus introduced to generate unbiased training data for parallel jaw grippers [44, 118, 179] and multi-gripper settings as well [130]. Nevertheless, synthetically trained **state-of-the-art grasp detectors lack an explicit notion of object instance**, which is critical for handling occlusions in dense piles and reducing the grasp search space in high-throughput applications.

(a) **Gripper-oriented bin-picking**

(b) **Model-based object-oriented bin-picking**

(c) **Model-free object-oriented bin-picking (Ours)**
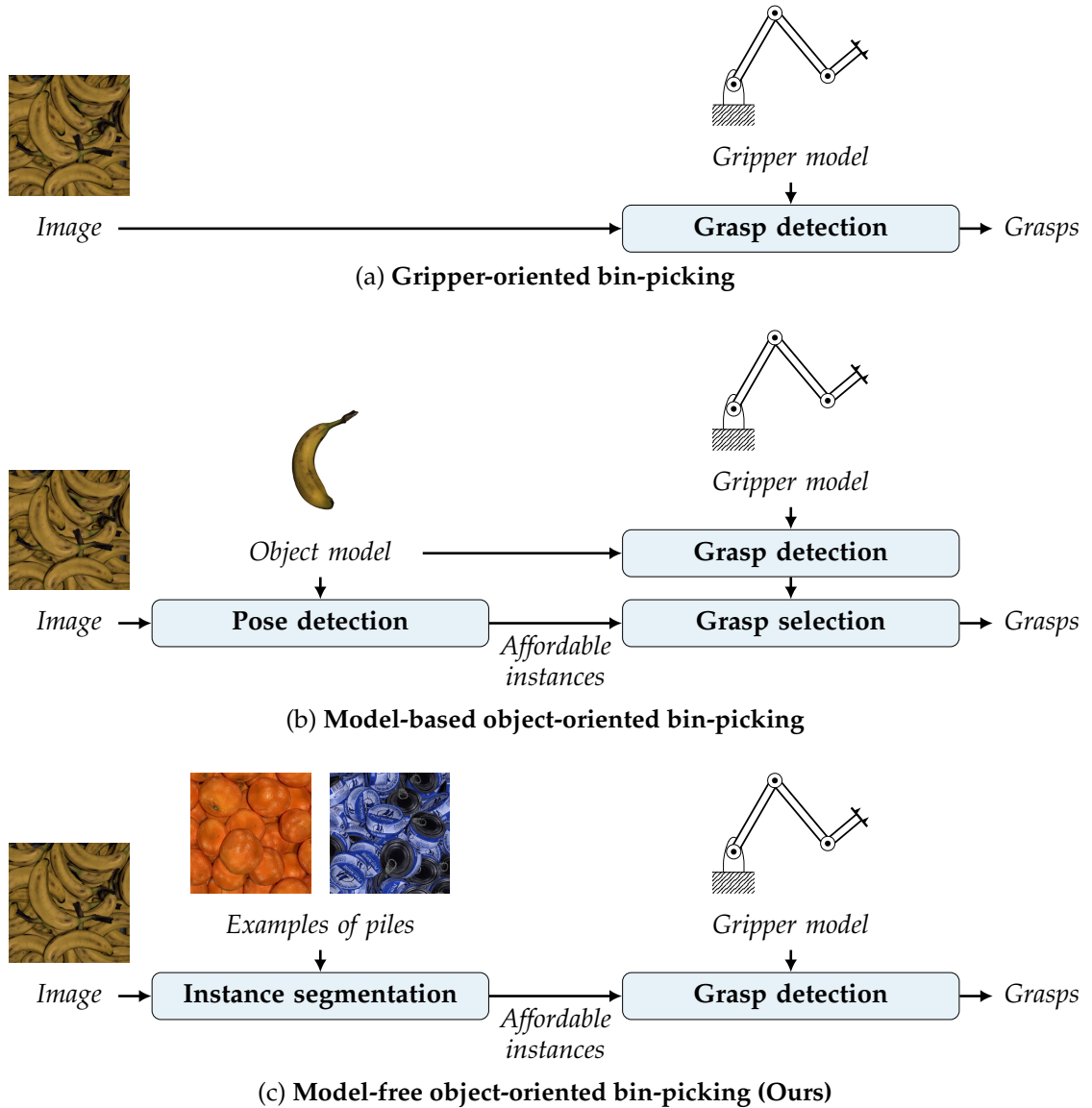
| | Gripper-oriented | Object-oriented | |
		Model-based	**Model-free (Ours)**
Provides 6D instance poses		✓	
Handles occlusions between instances		✓	✓
Does not require explicit object models	✓		✓
Handles intra-class variations	✓		✓

Figure 3.7: Comparison between bin-picking approaches. Gripper-oriented approaches define the notion of grasp affordance relatively to friction forces and torques without explicit notion of object instance, which is problematic in the case of strong occlusions and entanglements between instances. Alternatively, model-based object-oriented approaches rely on the notion of *pose of an object*, *i.e.* a distinguishable static state of this object [27]. The notion of pose is however limited to rigid objects and requires explicit models, which are not always available. In contrast with these approaches, we envision an object-oriented model-free approach based on the notion of *generic instance*. Such an approach aims to first delineate affordable instances independently of the object and gripper models. Grasp detection can then be oriented on the detected affordable instances.

3.1.2   Object-Oriented Bin-Picking

Object-oriented bin-picking aims to locate affordable instances independently of the gripper model(s). In such a perspective, **the notion of affordance is strongly related to the perception of occlusions**, while gripper-oriented approaches define affordance relatively to friction forces and torques. Object-oriented approaches are divided into two categories: *model-based* and *model-free*. Model-based methods assume an explicit model of the target object, while model-free algorithms are driven by image segmentation techniques. Figure 3.7 illustrates the differences between these approaches.

Model-Based Object-Oriented Bin-Picking   Model-based methods (see Figure Figure 3.19) rely on the notion of *pose of a rigid object*, *i.e.* a distinguishable static state of this object [27] (see Figure 3.8).

(a) Object CAD model   (b) Depth map   (c) Detected object poses
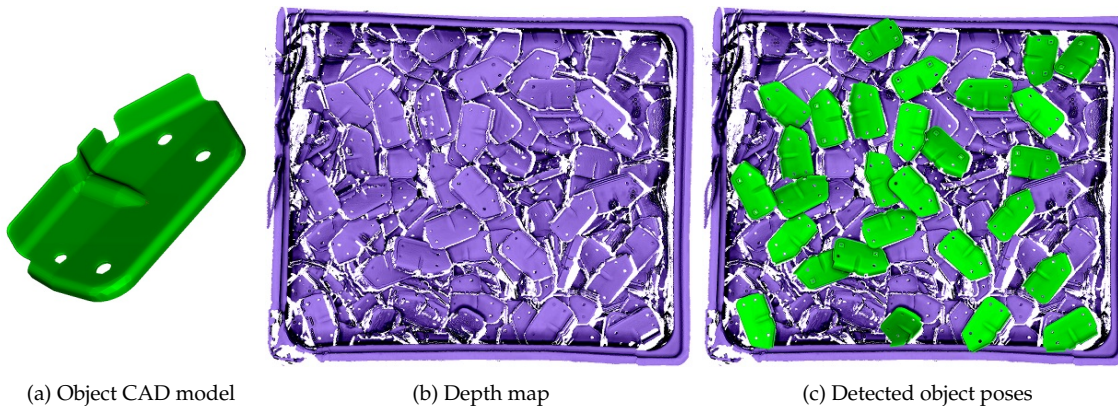
Figure 3.8: Example of depth-based object pose detection and estimation [27]. Such an approach requires an explicit object model, which is not always available in bin-picking applications.

The pose of an instance is commonly represented as a rotation and a translation that map a point in the object frame into the camera frame. Early approaches for pose detection in RGB images employed template matching from a discrete set of perspective projections [79, 80, 81, 202]. Covering a wide pose variability however requires a prohibitive number of templates. Alternative approaches resorted to heuristics for fitting predefined geometric primitives in point clouds, such as planar surfaces [149] or cylinders [69, 136]. Template-free approaches generalized to complex object models emerged using voting-based learning strategies based on depth-based hand-crafted data representations and feature clustering in the pose space [3, 20, 24, 38]. State-of-the-art pose detectors now leverage synthetically trained deep convolutional networks for representation learning [94, 102, 150]. Due to quantization

of the pose space or the selection of a pose cluster representative, a refinement of the pose estimation remains a necessary final step for accurate results [3, 24, 94, 102], typically using the iterative closest point registration method [18]. Generally, model-based approaches face two limitations for large-scale bin-picking applications. First, they **require explicit object models that are not always available**. Assuming an object model also excludes the case of instances with texture or geometric intra-class variations, unless prohibitively considering each variation as a new model. Second, **the notion of pose is defined for rigid objects**, thereby hardly appropriate for handling elastic deformations (see Figure 3.9).

(a) Template   (b) Input   (c) Detected template instances
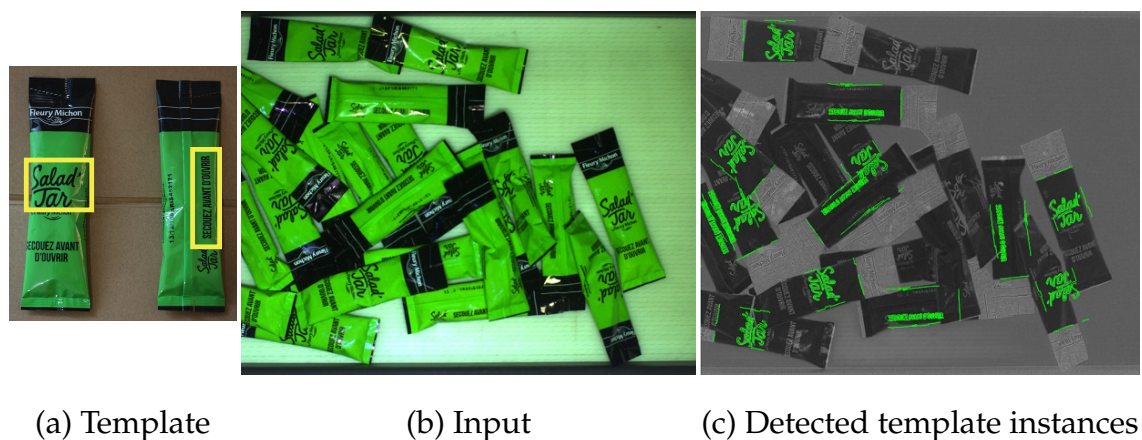
Figure 3.9: Example of shape-based matching using the Halcon software [1]. While strong priors on the target object texture are required, this approach fails to handle geometric intra-class variations such as elastic deformations.

Model-Free Object-Oriented Bin-Picking   Model-free approaches are related to the task of *instance segmentation*, which consists in delineating object instances without explicit object models (see Figure 3.10). Two main paradigms have emerged in the literature, localizing instances before and after delineation respectively.

Early-localization algorithms are built on the assumption that instances can be approximated as rectangles. They typically consist in first predicting rectangle region proposals that might contain an object [155], for further learning-based grasp detection [176], object function prediction [45], or amodal segmentation [184, 206], *i.e.* inferring both the visible and occluded object parts. While box proposal-based segmentation techniques certainly reduce the complexity in images sparsely populated with instances, such approaches prove inadequate for the case of dense piles (see later Figure 3.13) as a rectangle region may be shared by several instances.

| Urban scene understanding [73] | Vessel segmentation [126] | Plant analysis [161] |
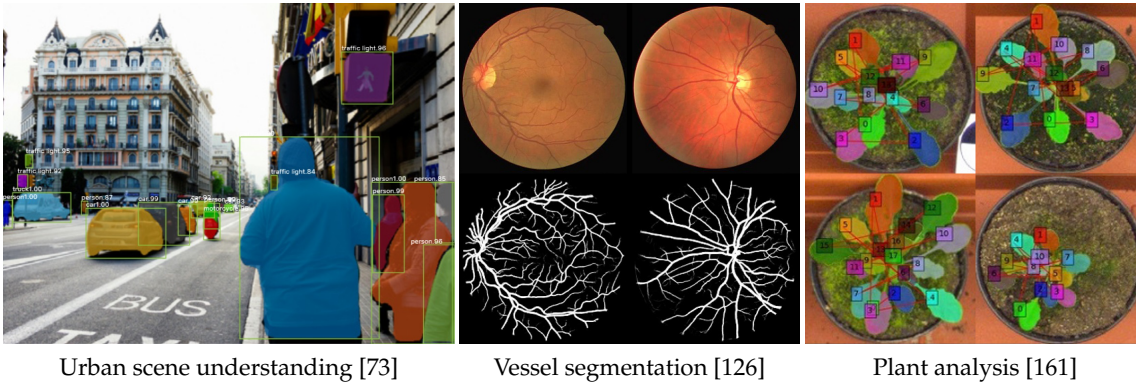
Figure 3.10: Examples of deep learning-based instance segmentation in various applications, illustrating the versatility of model-free approaches

Alternatively, late-localization approaches were first built on perceptual grouping heuristics from the Gestalt principles. They typically relied on heuristics-based bottom-up strategies for merging pixels into object parts, starting from a graph of superpixels [7, 182]. These graph-based models were further boosted by learning-based edge detection using structured random forests [46], and Pareto optimization for multi-scale combinatorial grouping [146]. Towards end-to-end learning, state-of-the-art algorithms for delineating instances leverage deep learning for instance boundary detection [188], *i.e.* classifying each pixel as a boundary or not, or semantic segmentation [35], *i.e.* assigning a category to each pixel. **Given our application context and objectives, we advocate the late-localization model-free object-oriented approach**. A deeper analysis of the state of the art on instance segmentation is thereby provided in Section 3.2.

3.2   Instance Segmentation

In this section, we review the state of the art on *instance segmentation*, which is a core topic of interest in model-free object-oriented bin-picking, and thereby in our approach. Instance segmentation is aimed at finely **delineating object instances in images, without explicit object models**.

Figure 3.11 illustrates the differences between types and granularities of object-level scene understanding in the literature. In this work, we are mostly interested in the task of *generic instance segmentation*, *i.e.* category-agnostic instance segmentation. Specifically, instance segmentation differs from the tasks of *image classification* and *semantic segmentation* in terms of objective: these two tasks aim instead to assign image-level and pixel-level object categories respectively, without

Image classification
Image-level categorization

Instance detection
Coarse localization

Instance segmentation
Fine localization

Semantic segmentation
Pixel-level categorization
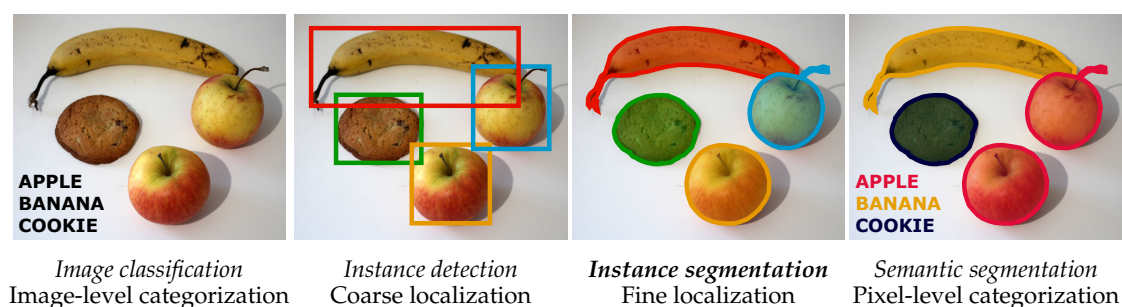
Figure 3.11: Types and granularities of object-level scene understanding
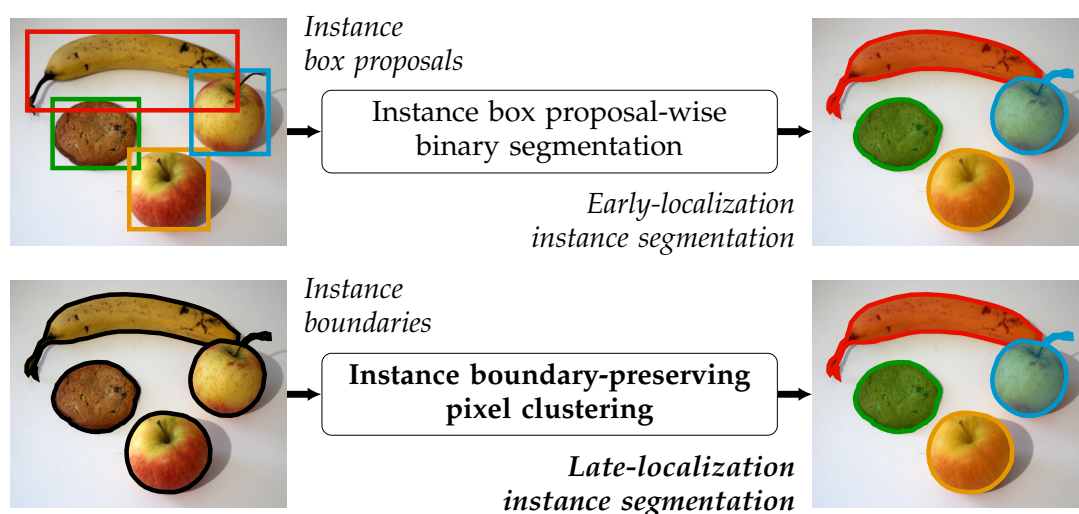
Figure 3.12: Approaches for instance segmentation. In the case of dense piles of objects, box proposals may contain multiple instances, thereby inducing ambiguous binary segmentations. We thus advocate a late-localization approach for first separating instances independently of their bounding box.

notion of instance. Note that instance segmentation augmented with pixel-level categorization thus defines the task known as *semantic instance segmentation*. Instance segmentation also differs from the task of *instance detection*, but in terms of granularity: instance detection consists in coarsely localizing instances by approximating them as rectangles.

As illustrated by Figure 3.12, instance segmentation techniques are divided into two categories: the early-localization and late-localization strategies, that consider instance localization as a task preceding or following instance delineation respectively. Early-localization approaches consist in first detecting the image region framing each instance. Late-localization approaches aim instead to first detect boundaries between instances to further cluster the pixels into instance segments.

Joint Boundary and Occlusion Detection   In both instance segmentation paradigms, a step further to separating instances consists in learning the notion of occlusion as well. Similarly to two-stream network architectures aimed at fusing different modalities [53, 167], **state-of-the-art approaches for this task rely on two independent streams** that predict boundaries and occlusions separately. Concretely, in the late-localization paradigm, it consists in detecting respectively the instance boundaries and their orientation [186]. In the early-localization paradigm, it consists in coloring respectively the visible instance mask and the mask including both the visible and invisible instance parts [206], namely the *modal* and *amodal* masks.

However, occlusions are a major source of instance boundaries. Considering occlusions jointly with boundaries could thus provide much richer information for scene understanding. Humans indeed leverage shadows and partially occluded patterns to instantly detect object boundaries and guess simultaneously the spatial relations between instances. Moreover, in state-of-the-art solutions, an instance-wise orientation is assigned to object boundaries in natural scenes for mostly describing which side is foreground, and which side is background [56, 186].

In contrast with previous works, we aim at addressing scenes composed of many instances occluding each other, in which the background is often hidden. In such configurations, the network should learn to answer instead the more general question: which side is above and which one is below? We now more deeply review the state of the art on both instance segmentation approaches, how the notion of occlusions is learned, and finally the training datasets for these tasks.

3.2.1   Early-Localization Instance Segmentation

Box proposal-based instance segmentation relies on the assumption that instances can be isolated in a rectangle. In such a perspective, the problem of segmentation is shattered into simpler multiple binary segmentations by first locating the instance bounding boxes [84].

Proposal-Based Segmentation   Early approaches consisted in merging regions based on hand-crafted region features [183, 187], or counting learning-based edges [76, 207]. Selecting rectangle regions containing objects was further boosted by deep region features [59, 60, 70, 116, 189], but still using heuristics for box proposal generation.

[155] thus introduced the concept of "region proposal network", which consists in using a deep convolutional encoder trained for locally assigning and regressing box templates from a predefined set

(i) Urban scenes   (ii) Dense piles of objects

(a) Unlike images sparsely populated with instances (i), rectangle fitting is unsuitable for the case of dense piles, as multiple instances often share the same rectangle region (ii).
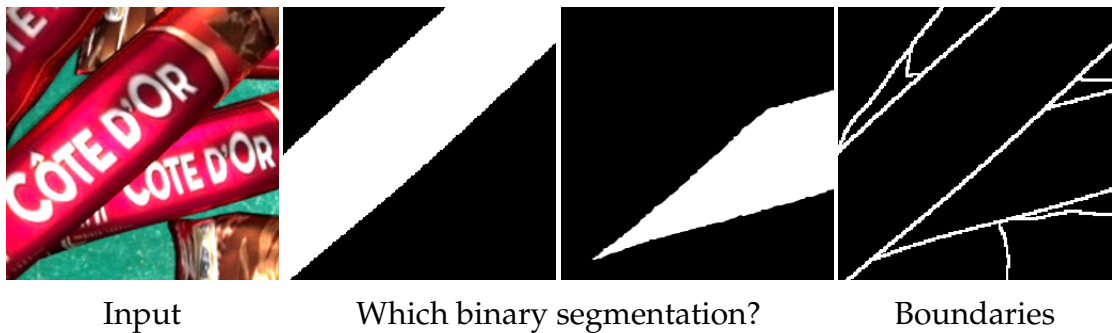
Input   Which binary segmentation?   Boundaries

(b) Unlike boundary detection, coloring an instance may result in classifying differently similar patterns inconsistently with the translation invariance property of convolutional layers.

(c) As a result, a box proposal-based segmentation approach [45] gives poor results on dense piles of sachets. First row: input; second row: best-scored detected box proposals (yellow rectangles) and corresponding binary segmentations (colored areas) after training on our synthetic training data. Best viewed in color

Figure 3.13: A box proposal-based segmentation approach is unsuitable for bin-picking scenes, as the rectangle fitting assumption becomes invalid.

of different scales and ratios. The trade-off between speed and ac-
curacy of such a network was further balanced by position-sensitive
score maps [41], by dividing the image into blocks instead of using
box anchors [152, 153], or by introducing multiscale nested connections
[198] and spatial pyramid pooling [74]. Inferring a pixel-wise instance
delineation in each box proposal was later obtained using a second
network trained for binary segmentation [143], whose spatial accuracy
was further improved by top-down refinement modules [144].

To finally obtain a convolutional structure combining the two steps,
i.e. box detection and binary segmentation, trainable end to end, [73]
introduced bilinear interpolation to pool the deep features of each
box proposal, thus enabling the recovery of higher-resolution instance
binary masks from low-resolution localization feature maps in one
feedforward pass. This two-network architecture was later enhanced
by bottom-up path augmentation for a better accuracy [112], seman-
tic segmentation for pixel-wise object categorization [33, 45], and a
loss function that alleviates the performance-penalizing foreground-
background class imbalance [108].

Limitations   However, coloring all the pixels of an instance becomes
an **ambiguous task if multiple instances occluding each other share
the same region proposal**, as in the case of bin-picking scenes (see Fig-
ure 3.13). Unlike people or cars in natural scenes, an **instance in a pile
does not often fit a rectangle** because instances piled up in bulk can
remain at rest in any pose. Furthermore, when a manufactured object is
instantiated multiple times like often in robotic setups, it may mean as
a consequence **classifying differently similar patterns, inconsistently
with the translation invariance property of convolutional layers**. An
instance in a dense pile of objects occluding each other may also occlude
other instances and be partially occluded at the same time.

Consequently, applying a box proposal-based instance segmenta-
tion approach on dense piles gives poor results. **In this work, we
thus consider deep convolutional networks for classifying pixels as
instance boundary or not, independently of their bounding box.** We
review the state of the art on this alternative instance segmentation
approach in Section 3.2.2.

3.2.2   Late-Localization Instance Segmentation

Late-localization approaches are based on learning similarities and
dissimilarities between pixels, for further clustering them into instance
segments independently of their bounding box.

Graph-Based Segmentation   Early strategies represented images as graphs of superpixels obtained by heuristics-based oversegmentation techniques [4, 42, 54, 141, 175] and consisted in greedily merging superpixels using hand-crafted features [7, 72, 129, 148, 171, 182]. The complexity of superpixel merging was later reduced by various learning-based algorithms [147] such as conditional random fields [6, 105], Pareto optimization on hierarchical region trees [37, 68, 146], tree-like structure of binary classifiers [185], deep convolutional networks for recovering contextual information among superpixels [77], parametric min-cuts over different seed locations [89, 90, 97], or level-set methods from geodesic distance transforms [96].

These learning-based merging techniques importantly relied on edge detections, as such cues constitute the premises of boundaries between instances. [46, 107] notably introduced structured random forests (SRFs) for learning to efficiently assign a contour patch to each pixel. SRFs were later outperformed by deep convolutional networks similarly trained to classify image patches [16, 165].

End-to-end training of instance boundary detectors finally emerged with fully convolutional encoder-decoder networks [17, 114, 125, 157, 188, 192, 193], as they enable to capture multiscale relations between pixels and recover high-resolution contour maps in one feedforward pass. **In this work, we thus consider fully convolutional networks, specifically encoder-decoder architectures, for classifying pixels as instance boundary or not, and for inferring their nearby spatial layouts as well.**

Encoder-Decoder Networks   Inspired by auto-encoders for unsupervised representation learning, encoder-decoder networks have been firstly introduced for single-task setups, such as semantic segmentation [11] and instance contour detection [193], in order to **recover high-resolution boundaries despite the resolution loss when encoding object-level semantics**.

The encoder produces deep hierarchical features, and the decoder gradually outputs a binary or category map using symmetric unpooling stages ($c.f.$ Figure 3.14a). To keep the upsampling efficient, the decoder typically uses max-unpooling layers that take the pooling indices from the encoder max-pooling layers. However, such an architecture requires the network to restore accurate boundaries only from the last encoder activation maps, where information is the most spatially compressed. Instead of a progressive decoding, [114, 192] introduced holistically-nested connections for a late fusion of all the encoder feature maps upsampled to the image scale, thus giving a multiscale view to the decoder (Figure 3.14b).
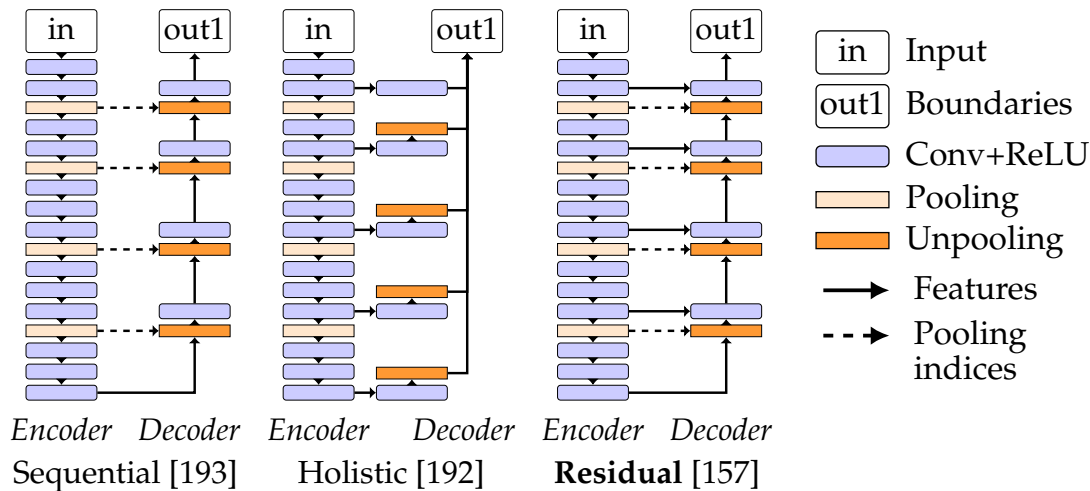
Figure 3.14: Single-stream VGG16-based [168] encoder-decoder architectures

Similarly to two-stream designs aimed at fusing different modalities [53, 167], the two-stream baseline for oriented boundary detection [186] employed independent encoder-decoder streams with holistically-nested connections. This architecture however hardly allows for learning a joint feature representation of boundaries and occlusions due to direct connections between each intermediate feature map and the output layer from which starts the backpropagation. In the context of semantic segmentation, [117] proposed to merge local and global semantics through a dual-task training, consisting in jointly decoding pixel labels and inferring image labels after the encoder. Image-level classification is however unfeasible in our object category-agnostic problem, although detecting instance boundaries and inter-instance occlusions require global cues as well.

Combining progressive upsampling with connections to the latent feature representations at each scale can be achieved alternatively by residual-like connections [75] between the encoder and decoder (*c.f*. Figure 3.14c), as proposed in single-task networks [43, 157, 188]. Residual-like connections notably proved to be superior to holistically-nested ones for single-stream encoder-decoder networks [188]. Indeed, by giving each decoder stage access to both the upsampled previous one and the corresponding encoder activation maps, the network can gradually merge the higher-level semantics of the previous scale with the spatial information lost during encoding at the current scale. Performing such a combination besides reduces the checkerboard artifacts inherent to unpooling [138]. In contrast with two independent multiscale encoder-decoder streams, the proposed bicameral design, which can viewed as an "encoder-bidecoder" structure, employs skip

connections to combine local and higher-level cues from a single feature space for detecting both boundaries and nearby occlusions.

3.2.3   Occlusion Detection from a Single Image

Finding occlusion relations has mostly been studied jointly with depth estimation in multiview contexts [57, 64, 208] and motion sequences [9, 10, 78, 91, 170, 172, 190], as occlusions often translate into missing pixel correspondences in different points of view or consecutive frames. Some recent works have more ambitiously focused on predicting a dense depth map from a monocular image [50, 104, 111], but the results are still less accurate than standard multi-view 3D reconstruction algorithms, and these techniques require ground-truth depth maps difficult to obtain.

However, considering a single point of view for inferring occlusions instead of distances from the camera seems more prone to success, as occlusions consist in binarized differences of depth at object boundaries. Such a binarization then conveys a relative depth ordering independent of the point of view and the distance between the camera and the scene. [156] firstly proposed a two-stage approach consisting in using an edge detector [128] to extract gradient-based features for a conditional random field (CRF) that performs local foreground/background classifications. Because local gradient-based features are limited for understanding occlusions, [82] introduced 3D cues within a similar procedure, by making assumptions on the global 3D structure of the scene (sky, ground). Observing that detecting objects and foreground/background occlusions are actually coupled tasks, object part segmentation and figure/ground organization were later recovered in a single step using angular embedding [122]. However, as angular embedding and CRFs both require expensive computational time at large scales, [177] suggested a faster simultaneous edge and foreground/background detection by leveraging structured random forests [46], but still using hand-crafted features derived from a limited set of contour token clusters. In order to avoid human biases when defining features, a convolutional neural network (CNN) was instead employed to produce contextual feature representations [56] or to learn pixel-centric pairwise relations for affinity and figure/ground embedding [123]. Towards end-to-end training, and in the footsteps of fully convolutional networks (FCNs) for pixel-wise classification, two approaches have lately emerged (see Figure 3.15).

Box proposal   *Box proposal*   *Image*   *Image*

Modal mask   *Amodal mask*   *Boundaries*   *Orientations*
(a) **Amodal segmentation** [206]   (b) **Oriented boundaries** [186]
    (Early localization)   (Late localization)
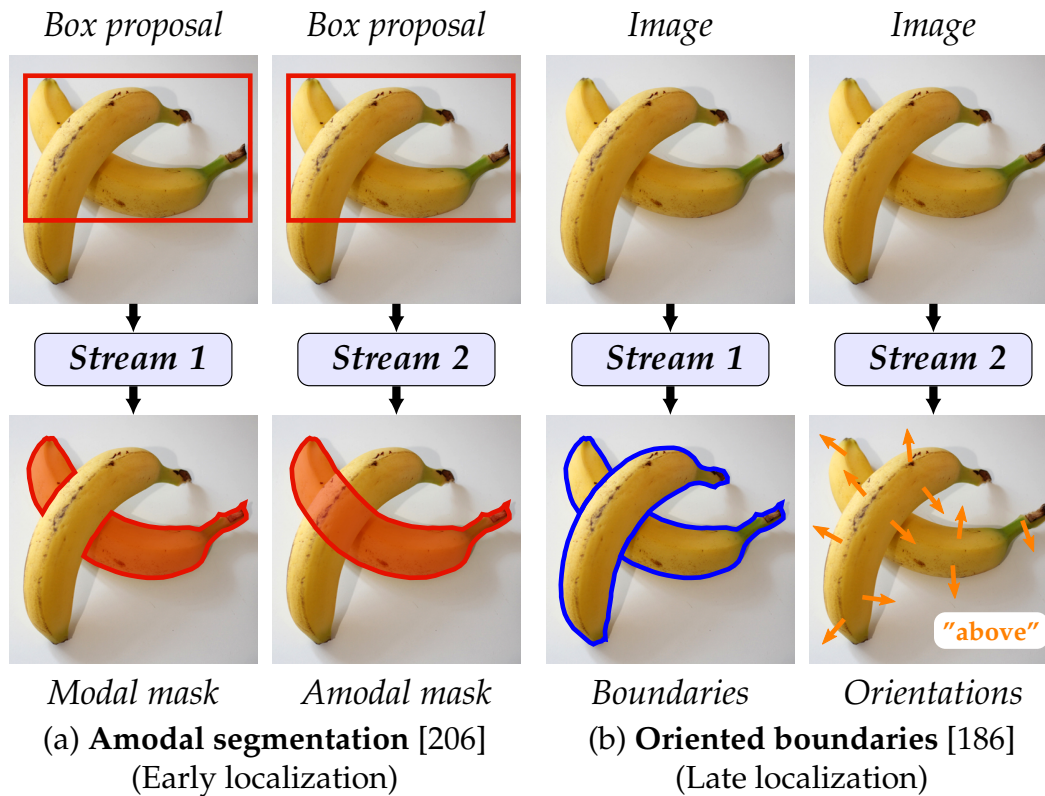
Figure 3.15: State-of-the-art approaches for learning instance boundaries and occlusions rely on two independent streams. In the early-localization paradigm (a), the streams predict the modal and amodal masks respectively, *i.e.* the mask of the visible parts and the mask including both the visible and occluded parts. In the late-localization paradigm (b), the streams predict respectively the boundaries and their orientations indicating which side is above and which side is below. However, in bin-picking scenes, boundaries are mostly caused by occlusions. We thus advocate a joint representation learning instead of two independent streams.

Amodal Segmentation   The first approach [206], namely amodal instance segmentation, follows the two-step early-localization paradigm of region proposal-based instance segmentation, but aims instead at predicting for each instance the mask including both the visible and the non-visible instance parts. An estimation of the instance occlusion rate can then be obtained by comparing the predicted *modal* and *amodal* masks, *i.e.* the mask of the visible parts and the mask including both the visible and occluded parts. **This approach however cumulates the drawbacks of region-based instance segmentation, discussed in Section 3.2.1, and the difficulty of coloring something invisible**, thus resulting in low instance boundary accuracy.

Oriented Boundary Detection   The second approach [186] follows the late-localization paradigm and consists in a two-stream FCN that predicts independently boundaries and their occlusion-based orientation in one forward pass. More precisely, [186] set up one stream of the network to predict the raw orientation of a local unit vector specifying the occlusion relations by a left-hand rule, and used a logistic loss function that strongly penalizes wrong directions but only weakly tangent directions. However, to ensure a local continuity, the orientation predictions have to be further "adjusted" using the local tangent vectors of the predicted boundaries as the network may not predict similar orientations for neighbourhood pixels. There is indeed no constraints ensuring a local continuity of the network prediction, all the more as the ground-truth orientation map is noisy itself.

To overcome this issue and remove any post-processing step, we propose instead to reformulate the occlusion prediction as a local binary segmentation problem near boundaries. Both modelled as binary maps, boundaries and occlusions can then be detected using a single fully convolutional encoder-decoder structure equipped with residual-like connections, *i.e.* the proposed bicameral design, thus efficiently sharing features instead of using two independent encoders.

3.2.4   Datasets for Boundary and Occlusion Detection

Joint boundary and occlusion detection from a single image raised interest with the BSDS Border Ownership dataset (BSDS-BOW) [156], derived from the BSDS500 dataset [127] for object contour detection, which contains 200 natural images manually annotated with object part-level oriented contours. As state-of-the-art FCNs require more training data, [186] presented a dataset larger than BSDS-BOW, namely the PASCAL Instance Occlusion Dataset (PIOD), comprising about 10,000 manually annotated natural images from the PASCAL VOC Segmentation dataset [51]. Similarly for amodal instance segmentation, [55, 206] have proposed real-world datasets, namely the Densely Segmented Supermarket Amodal dataset (D2SA) and the COCOA Amodal dataset (COCOA). These latter datasets are subsets of much larger datasets for instance segmentation in the early-localization paradigm, COCO [109] and D2S [55] respectively, but augmented with the ground-truth amodal annotations, that can be derived for oriented boundary detection. Despite their challenging instance intra-class variability, the support images contain few instances and are limited in terms of inter-instance occlusions.

PIOD [186]

COCOA [206]

D2SA [55]

Mikado (Ours)

	Dataset	PIOD [186]	COCOA[1] [206]	D2SA[1] [55]	Mikado (Ours)
Overall	Average resolution	469×386	578×483	1962×1569	640×512
	Number of images	10,100	3,823	5,600	2,400
	Number of instances	24,797	34,884	28,703	**48,184**
	Ground-truth annotations	Human-made			Computer-generated
Per image	Number of instances	2.5	9.1	5.1	**20.1**
	Inter-instance occlusions	1.3	13.5	2.8	**52.9**
	Background pixels	69%	33%	79%	24%

[1] The statistics on COCOA and D2SA are only on the train and validation subsets as the test subset is not provided.

Figure 3.16: Samples and characteristics of the state-of-the-art datasets for oriented boundary detection [186] and amodal instance segmentation [55, 206] compared with our synthetic data. Unlike Mikado, occlusions in these datasets are mostly due to objects occluding the background, thereby unsuitable for bin-picking scenes in which occlusions are between instances.

However, in our robotic setups, scenes are composed of many instances occluding each other. Applying on such scenes a model trained on PIOD or COCOA would give poor results since **these datasets provide mostly foreground/background boundary examples** for training. As extending hand-labeled real-world datasets is a time-consuming task, D2SA partly alleviates this concern by artificially overlaying manually delineated instances for creating fake images with more instances, but at the cost of lighting inconsistencies at instance boundaries and limited pose variations (see Figure 3.17). The images from these datasets suffer besides from missing or ambiguous ground-truth annotations, thereby introducing human biases during training and test.

(a)   (b)
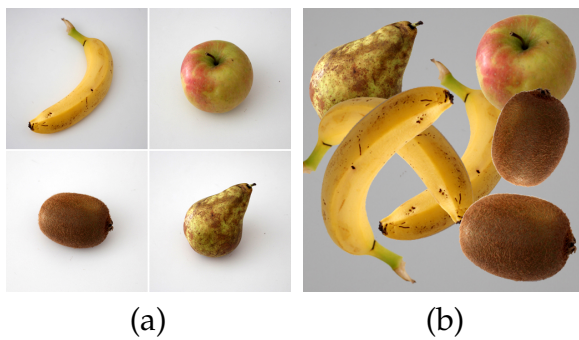
Figure 3.17: Illustration of the augmentation strategy of [55], later referred to as D2SA+. Fake training images of piles (b) are created by overlaying manually isolated instances (a) from real images.

To address these issues, synthetic datasets [26, 131, 158] have emerged during the course of this work for learning and evaluation as they offer a fully controlled environment and a perfect ground truth. For benchmarking the tasks of pose detection and estimation, [26] generated depth images of many rigid instances piled up in bulk. For the tasls of object detection, semantic segmentation and instance segmentation, [131] and [158] proposed synthetic urban and indoor scenes respectively.

In this work, we address bin-picking scenes of deformable objects in large-scale applications. **As in such scenarios real-world datasets are not available and annotating real-world data is prohibitive in industrial processes, we propose to make and use synthetic data for oriented boundary detection from a single image**. As depicted by Figure 3.16, **our synthetic data, namely Mikado, emphasizes the inter-instance occlusions that are under-represented in the state-of-the-art real-world datasets for the same task**.
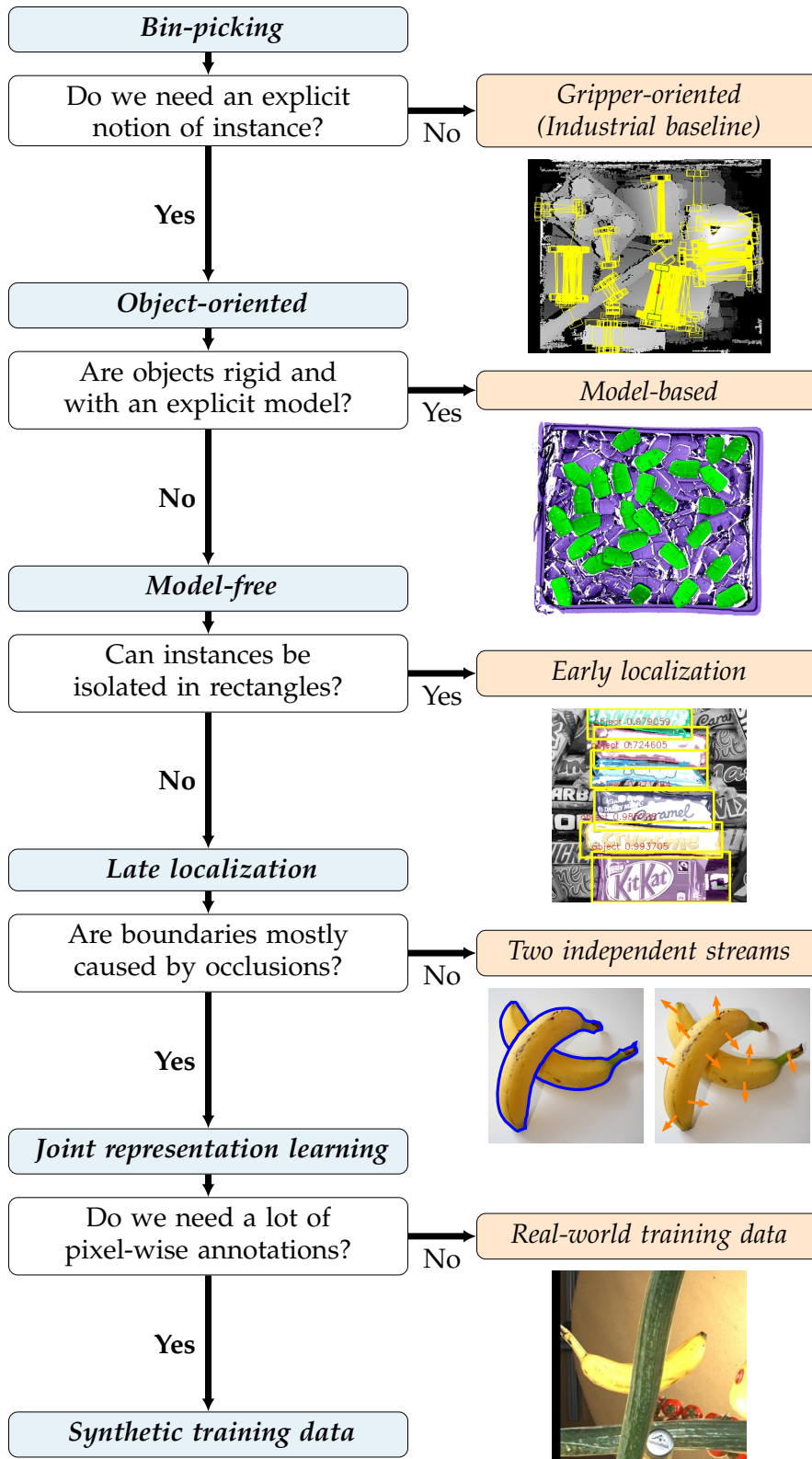
Figure 3.18: Reasoning pipeline for building our solution

3.3   Conclusion

Bin-picking approaches are divided into two categories: gripper-oriented and object-oriented (*c.f*. Figure 3.19). Unlike object-oriented approaches, gripper-oriented techniques lack the notion of instance, which is important for handling occlusions in dense piles of instances. Object-oriented bin-picking relies either on the notion of pose, thereby requiring an explicit rigid object model, or on generic instance segmentation techniques (*c.f*. Figure 3.20) which can handle deformable objects as well. The task of instance segmentation can be fulfilled either by first coarsely locating instances before delineation, but under the assumption that instances can be isolated in a rectangle, or by learning boundaries between instances independently of their bounding box. Given our application context and objective, **we thus advocate a model-free late-localization object-oriented approach** (*c.f*. Figure 3.18).

Specifically, **we propose a synthetically trained bicameral convolutional structuring for jointly learn boundaries and occlusions from a single image**. In contrast with state-of-the-art approaches, which rely on two independent streams for boundaries and occlusions respectively, we explore in Chapter 4 a fully convolutional architecture for joint representation learning, as occlusions strongly cause boundaries in bin-picking scenes. We then present and discuss through extensive experiments in Chapter 5 a synthetic training data generation pipeline for applying such a network to real-world bin-picking.
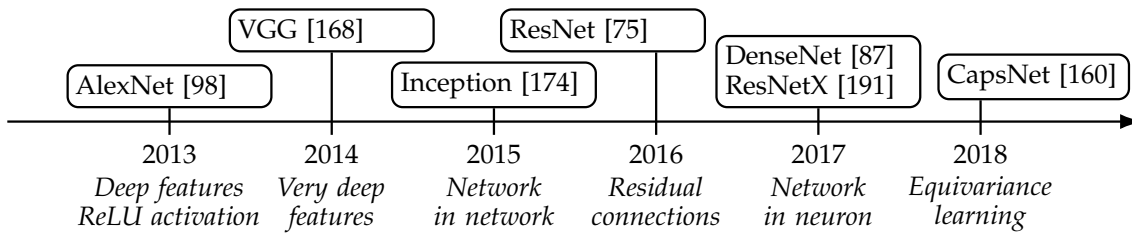
(a) **Deep convolutional encoders**
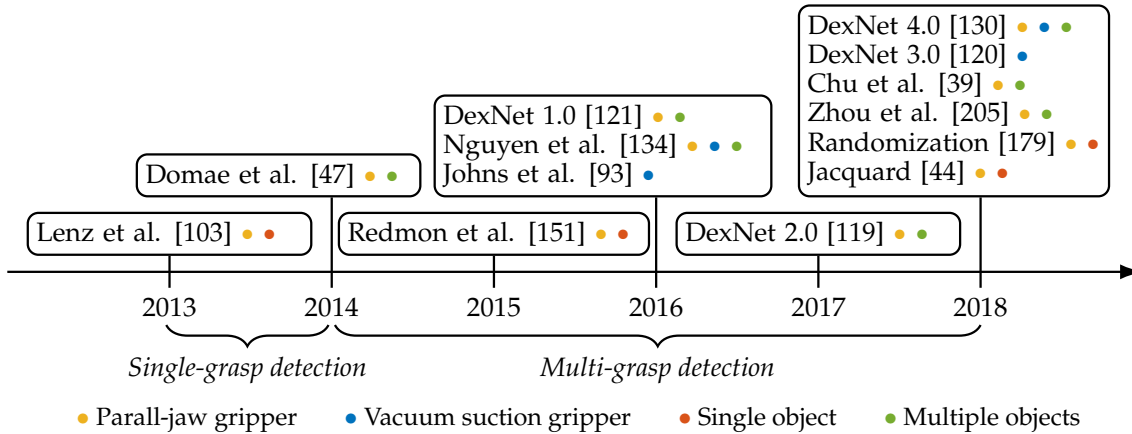
(b) **Gripper-oriented bin-picking**
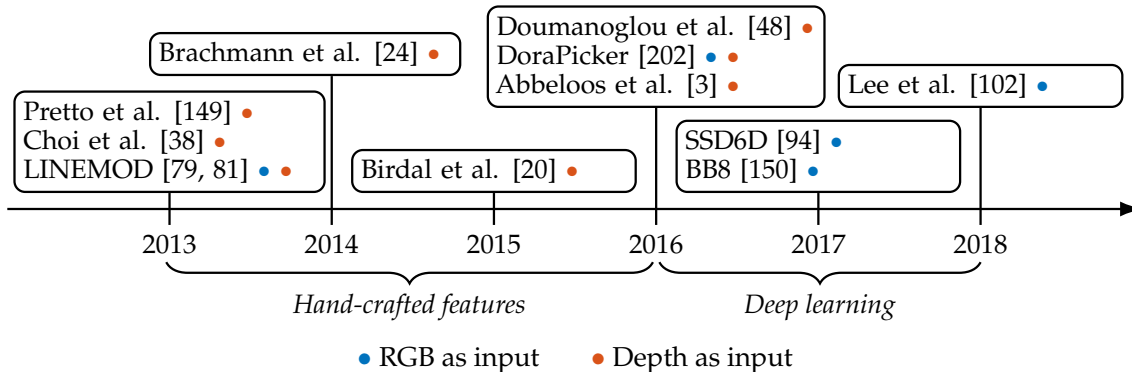
(c) **Model-based object-oriented bin-picking**
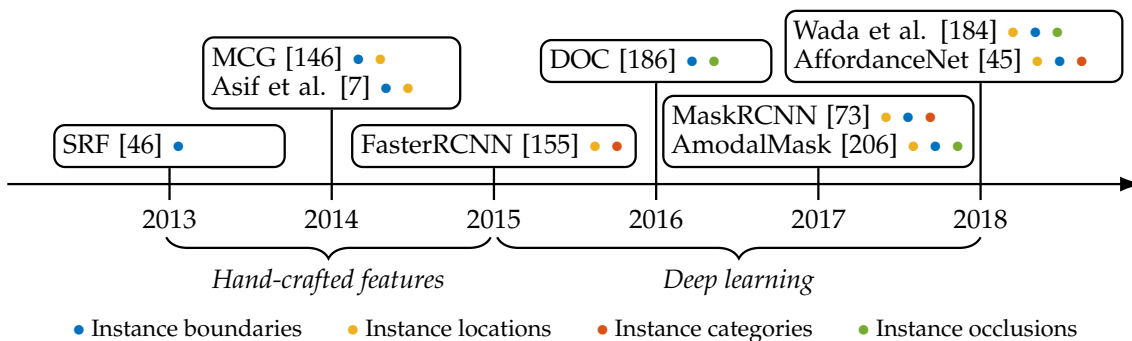
(d) **Model-free object-oriented bin-picking**

Figure 3.19: Evolution of the state of the art for gripper-oriented (b) and model-based object-oriented (c) bin-picking approaches
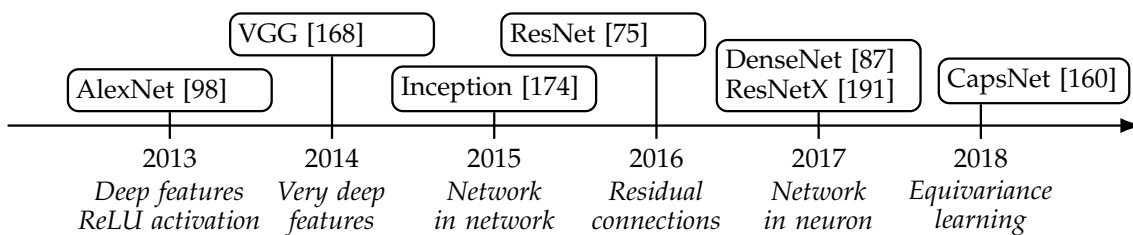
(a) **Deep convolutional encoders**
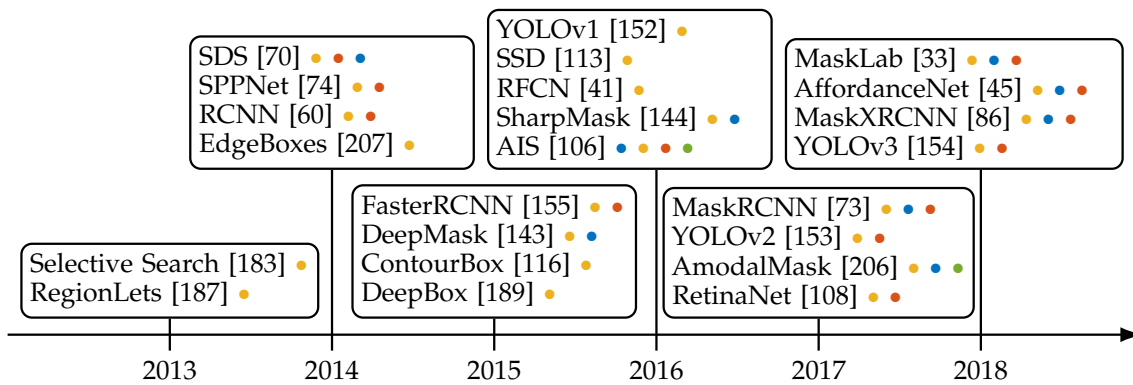
(b) **Early-localization instance segmentation**
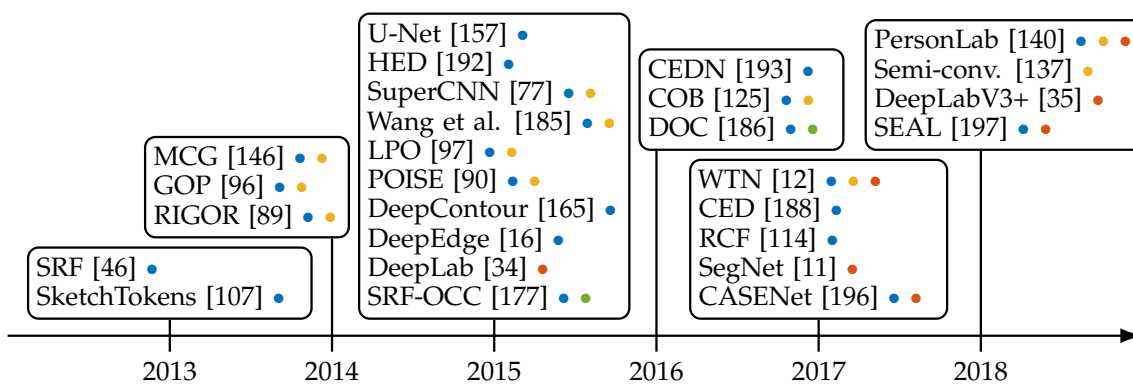
(c) **Late-localization instance segmentation**
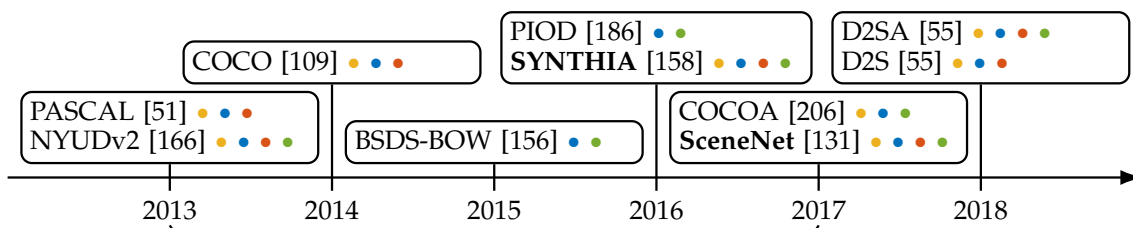
(d) **Datasets**

● Instance boundaries ● Instance locations ● Instance categories ● Instance occlusions

Figure 3.20: Evolution of the state of the art for instance segmentation

Chapter 4

Occlusion-Aware Instance Segmentation

In this chapter, we present our contributions on inferring instance boundaries and occlusions from a single image, using the deep learning tools presented in Chapter 2, in the ultimate goal of inferring the most affordable instance of a pile in bin-picking applications.

4.1 Bicameral Structuring

In bin-picking scenes, the boundaries between instances are mostly due to their mutual occlusion. Unlike the state-of-the-art two-stream approaches, we thus argue that boundaries and occlusions should be learned from a joint feature space (see Figure 4.1).

Two streams (Baseline) **Bicameral structuring (Ours)**
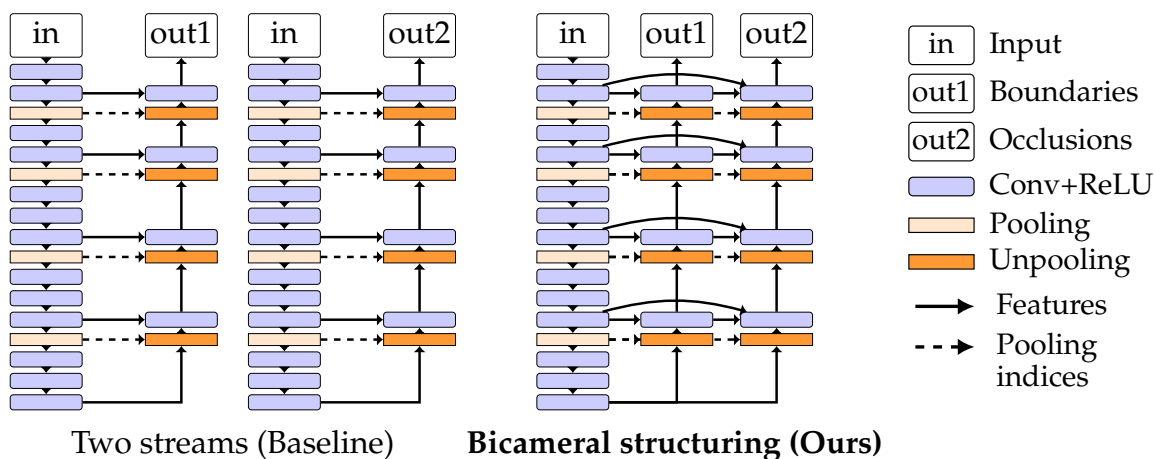
Figure 4.1: The proposed bicameral structuring compared with the two-stream baseline built from [186] and [157].

Specifically, we propose a *bicameral* fully convolutional structure, consisting in one encoder shared by two cascaded decoders through skip connections, for jointly inferring the instance boundaries and their unoccluded side from a single RGB image.

4.1.1   Bicameral Architecture

In our experiments, we implement a bicameral FCN from a VGG16-based [168] encoder backbone, as depicted by Figure 4.2. Alternative backbones such as [75, 87] can be employed as well; this point is addressed in our ablation study. The two cascaded decoders have the same structure: four convolutional layers whose numbers of filters are, from bottom to top, 256, 128, 64, 32. The convolutional layers of each decoder are interleaved with unpooling layers that take as input the pooling indices from the encoder max-pooling layers. As illustrated in Figure 4.3, such an unpooling mechanism enables to reuse the most salient pixel locations fired by the encoder during the feedforward, instead of arbitrarily padding with zeros. The first decoder is trained to recover boundaries, while the second decoder handles occlusions, modelled as boundary orientations. The boundary decoder is linked to the encoder by residual-like connections [75] to the top of each encoder convolutional block. The occlusion decoder is similarly connected to the encoder, but also to each boundary decoder convolutional layer. Skip connections are implemented by concatenating feature maps. We investigate alternative implementations of skip connection in our ablation study, but show that using concatenation is the best default choice.

4.1.2   Bicameral Learning

Boundaries   Instance boundaries are modelled as binary maps such that activated pixels are boundaries, *i.e.* pixels in contact with two or more different instance labels. For such a binary classification, we thus use a regularized cross-entropy loss function (*c.f.* Chapter 2). However, boundary pixels generally represent a small fraction of a whole image. To alleviate this class imbalance between boundary and non-boundary pixels, we employ a variant of cross-entropy, referred to as a *balanced cross-entropy* loss function [114, 157, 188, 192, 193]. Specifically, the two terms of the loss function are weighted in order to counterbalance the low number of boundary pixels against non-boundary pixels. In our experiments, we set these weights such that the "contour pixel" penalty is 10 times more important than the "non-contour" term, regardless of the dataset. We are aware of recently proposed more advanced loss functions [43, 108, 197] but we leave the introduction of these loss functions for future work.

Layer name	Number of filters	Kernel size
conv1X	64	
conv2X	128	
conv3X	256	3×3
conv4X	512	
conv5X	512	
deconv4x	256	
deconv3x	128	5×5
deconv2x	64	
deconv1x	32	
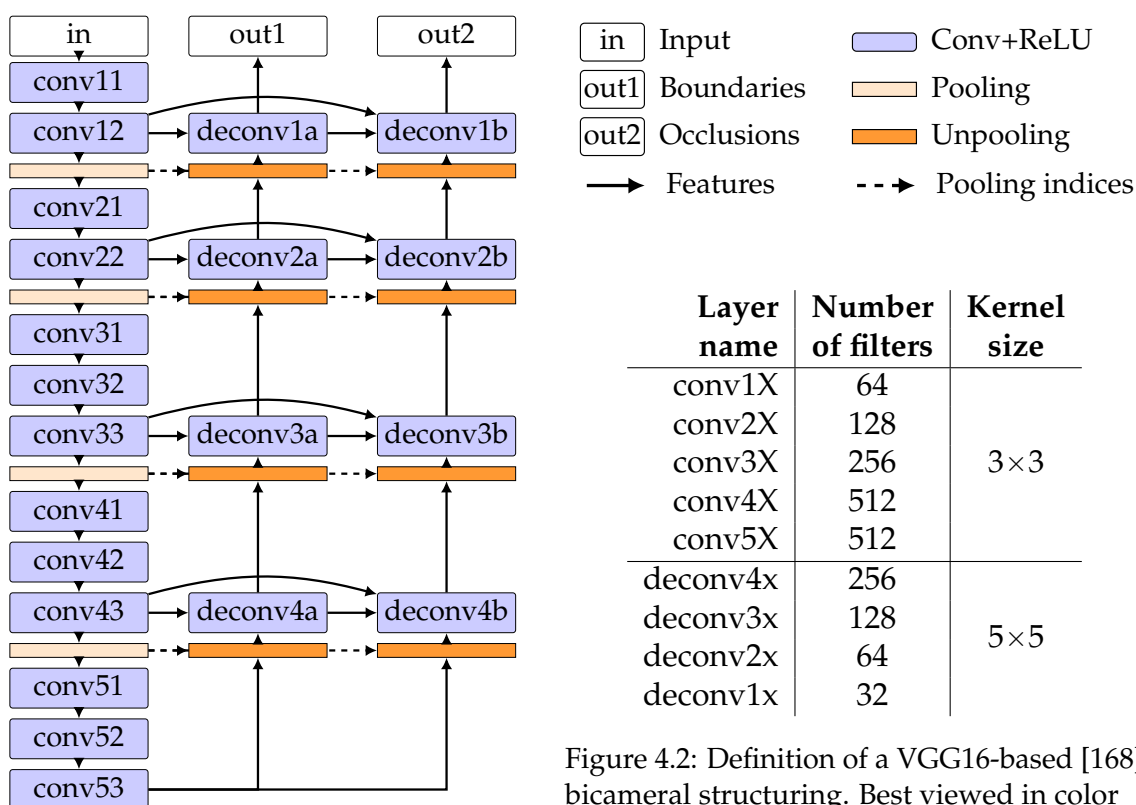
Figure 4.2: Definition of a VGG16-based [168] bicameral structuring. Best viewed in color
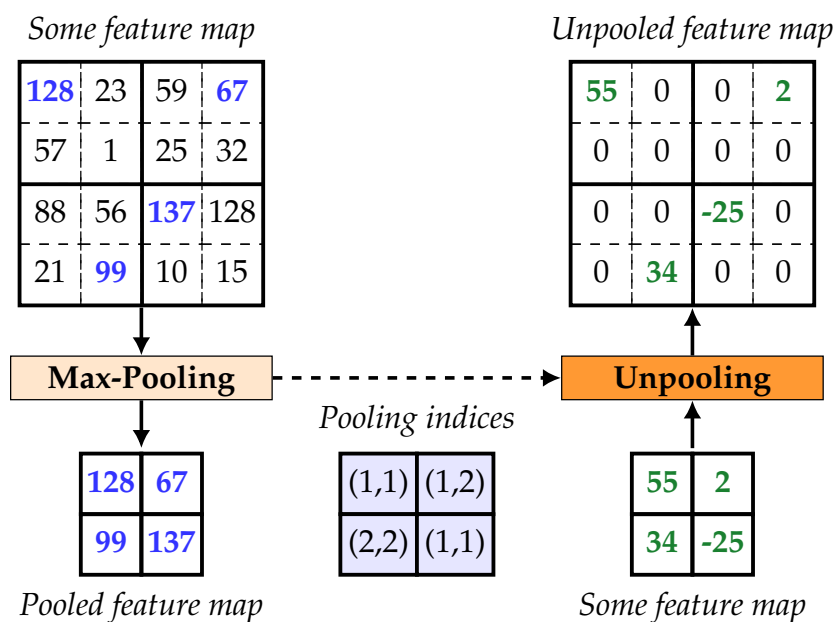
Figure 4.3: Unpooling mechanism, also introduced in [11, 193]. When a feature map is pooled in the encoder by a max-pooling operator, the pooling indices are stored for the corresponding decoder's unpooling layer. This enables to reuse the most salient pixel locations when unpooling, instead of arbitrarily padding with zeros.

Occlusions   In the two-stream baseline [186], the stream for occlusions consists in inferring the raw orientation $\theta \in (-\pi, \pi]$ of a local unit vector specifying the occlusion relation by a left-hand rule, independently of the stream for boundaries. Their orientation learning is driven by a logistic loss function that strongly penalizes wrong directions but only weakly tangent directions. However, a consistency check between boundaries and orientations is required after each forward pass, by using the local tangent vectors of the predicted boundaries for adjusting the predicted orientations, since by construction, there is no mechanism enforcing a local continuity of the network inference.

Input   Ground truth   Input   Ground truth
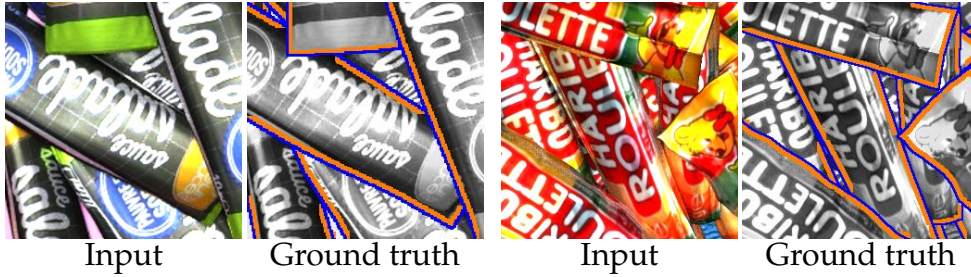
Figure 4.4: Our representation of boundaries (blue) and occlusions (orange). Near a boundary segment, an occlusion is locally represented as a binary segmentation highlighting the occluding side of this boundary segment.

By using instead a single encoder-decoder structure, and reformulating the occlusion prediction as a local binary segmentation problem close to instance boundary detection, we can overcome this limitation. In practice, near each boundary pixel, we propose to set the side which is above the other one to "1", and the side below to "0". More precisely, for generating the ground-truth occlusion binary maps, we sweep all the ground-truth instance boundaries, and for each boundary pixel, we binarize the centered local region by computing the mean Z-offset in each segment of the region. In the end, a ground-truth occlusion map is a binary map whose positive pixels are the instance boundaries slightly translated to one side or another, according to the relative depth difference of the boundary sides, as illustrated by the final ground-truth image in Figure 4.4. Note that boundary pixels are set to 0 in the occlusion map. Occlusions can then be learned using a balanced cross-entropy loss function as well.

Bicameral Loss Function   Formally, let $\mathbf{p} \in \mathcal{P}$ be a pixel location; typically $\mathcal{P} = \{1, .., W\} \times \{1, .., H\}$ for an image of width $W \in \mathbb{N}^*$ and height $H \in \mathbb{N}^*$. We note $\mathcal{N} = \{1, .., N\}$ where $N \in \mathbb{N}^*$ is the number of training images, and $\mathbf{M_p} \in \mathcal{V}$ the value at location $\mathbf{p} \in \mathcal{P}$ in a matrix $\mathbf{M} \in \mathcal{V}^R$, with $R = W \times H$. With the proposed formulation for

occlusions, the network jointly minimizes two balanced cross-entropy loss functions $\mathcal{L}_{\text{boundaries}}$ (Eq. 4.1) and $\mathcal{L}_{\text{occlusions}}$ (Eq. 4.2), respectively for instance boundaries and occlusions, defined as follows:

$$\mathcal{L}_{\text{boundaries}}(\mathbf{W}) = -\frac{1}{|\mathcal{N}||\mathcal{P}|} \sum_{n \in \mathcal{N}} \sum_{\mathbf{p} \in \mathcal{P}} \mu \mathbf{Y}_{n\mathbf{p}} \log(\hat{\mathbf{Y}}_{n\mathbf{p}}) \\ + (1 - \mathbf{Y}_{n\mathbf{p}}) \log(1 - \hat{\mathbf{Y}}_{n\mathbf{p}}) \tag{4.1}$$

$$\mathcal{L}_{\text{occlusions}}(\mathbf{W}) = -\frac{1}{|\mathcal{N}||\mathcal{P}|} \sum_{n \in \mathcal{N}} \sum_{\mathbf{p} \in \mathcal{P}} \mu \mathbf{Z}_{n\mathbf{p}} \log(\hat{\mathbf{Z}}_{n\mathbf{p}}) \\ + ((\mu - 1)\mathbf{Y}_{n\mathbf{p}} + 1)(1 - \mathbf{Z}_{n\mathbf{p}}) \log(1 - \hat{\mathbf{Z}}_{n\mathbf{p}}) \tag{4.2}$$

where $\{(\mathbf{X}_n, \mathbf{Y}_n, \mathbf{Z}_n) \in (\mathbb{R}^3)^R \times \{0,1\}^R \times \{0,1\}^R\}_{n \in \mathcal{N}}$ is the training dataset of RGB images $\mathbf{X}_n$, associated with their ground-truth binary maps $\mathbf{Y}_n$ and $\mathbf{Z}_n$ for boundaries and occlusions respectively. If a pixel $\mathbf{p} \in \mathcal{P}$ of the image $\mathbf{X}_n$ is an instance boundary then $\mathbf{Y}_{n\mathbf{p}} = 1$, else $\mathbf{Y}_{n\mathbf{p}} = 0$. If $\mathbf{p}$ is the unoccluded side of an instance boundary then $\mathbf{Z}_{n\mathbf{p}} = 1$, else $\mathbf{Z}_{n\mathbf{p}} = 0$. $(\hat{\mathbf{Y}}_n, \hat{\mathbf{Z}}_n) = f_{\mathbf{W}}(\mathbf{X}_n) \in [0,1]^R \times [0,1]^R$ designates the network inference for boundaries and occlusions respectively, using the parameters $\mathbf{W}$. In practice, $\mu = 10$. The factor $((\mu - 1)\mathbf{Y}_{n\mathbf{p}} + 1)$ in Eq. 4.2 ensures consistency with Eq. 4.1, as we want the intersection between the boundary and occlusion binary maps to be empty. Basically, this factor enables to give the "is-not-unoccluded-side" penalty as much importance as the "is-unoccluded-side" term when a pixel in the occlusion map is a boundary, *i.e.* $\mathbf{Y}_{n\mathbf{p}} = 1$ and $\mathbf{Z}_{n\mathbf{p}} = 0$.

Implementation Details

- In practice, to ensure probability-like inferences, we train the network to infer $(\hat{\mathbf{Y}}_n, \hat{\mathbf{Z}}_n) \in \mathbb{R}^R \times \mathbb{R}^R$, then we apply the sigmoid function $\sigma : \mathbb{R} \to [0,1]$, $x \mapsto (1 + \exp(x))^{-1}$.

- In addition to Equations 4.1 and 4.2, we perform a ℓ_2-regularization of the parameters to avoid overfitting, but the regularization term is here omitted for brevity (*c.f.* Chapter 2).

- When generating the ground-truth occlusion map, local patches that contain more than two segments are fully set to 0 as they cannot be binarized. This proves to be a reasonable limitation as in practice an overwhelming majority of boundary pixels are between only two instances or between an instance and the background (*e.g.*, 97.1% of the boundary pixels in our synthetic dataset, and 99.4% in PIOD [186]). We leave for future work the study of the minority of pixels at the junction of more than two instances.

4.1.3 Experimental Setup

We now describe our experimental protocol to evaluate the proposed network architecture on real-world and synthetic data. We are interested in finding answers to the two following questions:

1. **Is the proposed bicameral structuring the best architecture for oriented boundary detection?**

2. **How does the late-localization paradigm compare with the early-localization paradigm (see Figure 4.5)?**

To investigate each of these questions, we conduct two sets of experiments that consist in respectively comparing:

1. the proposed bicameral design with our two-stream baseline and alternative architectures, on PIOD [186] and Mikado;

2. the proposed bicameral design with the box proposal-based approach on Mikado, then the amodal segmentation approach on COCOA [206].

More precisely, each set of experiments is respectively composed of comparisons between:

1. (a) the bicameral design and our two-stream baseline for oriented boundary detection, built from [186] and [157];

    (b) the bicameral design and alternative architectures;

    (c) bicameral designs with different partial feature sharing between the bicameral decoders;

    (d) bicameral designs with and without skip connections;

    (e) bicameral designs with different type of skip connections;

    (f) bicameral designs with different encoder backbone;

2. (a) the bicameral design with the box proposal-based segmentation approach [45] for instance boundary detection;

    (b) the bicameral design with a two-stream network for amodal instance segmentation [206].

As reported by Table 4.1, the remaining of this section focus on experiments 1a and 2a–b. Experiments 1b–f constitute our ablation study and are discussed in Section 4.3. The generation and plausibility of Mikado are discussed in Chapter 5.
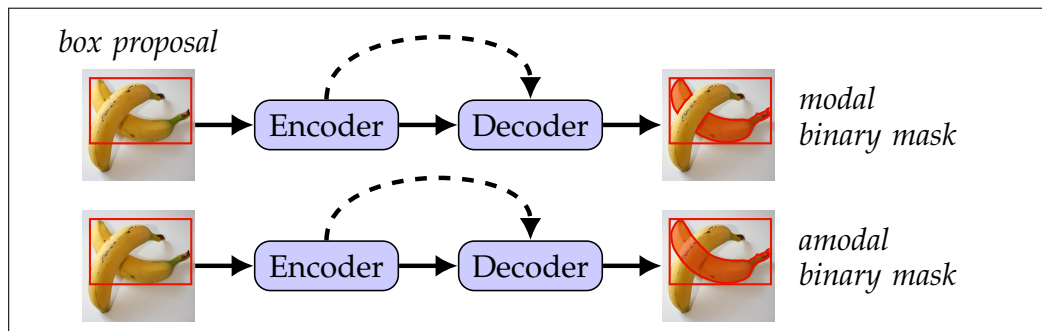
Amodal segmentation [206] – *Early localization*
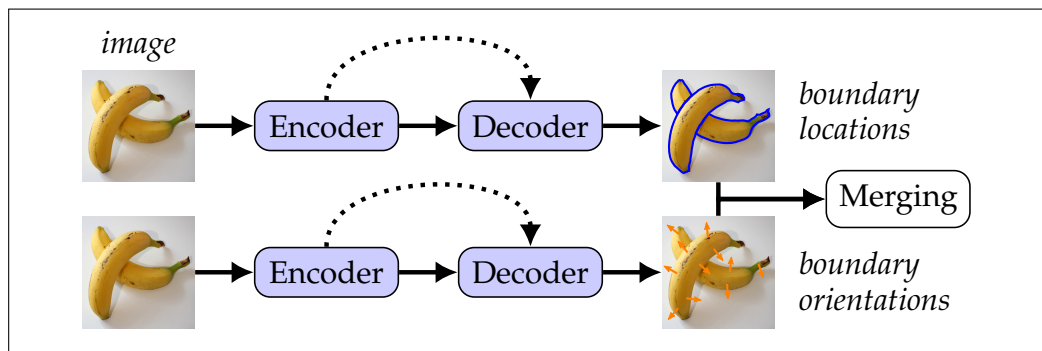
Oriented boundary detection [186] – *Late localization*
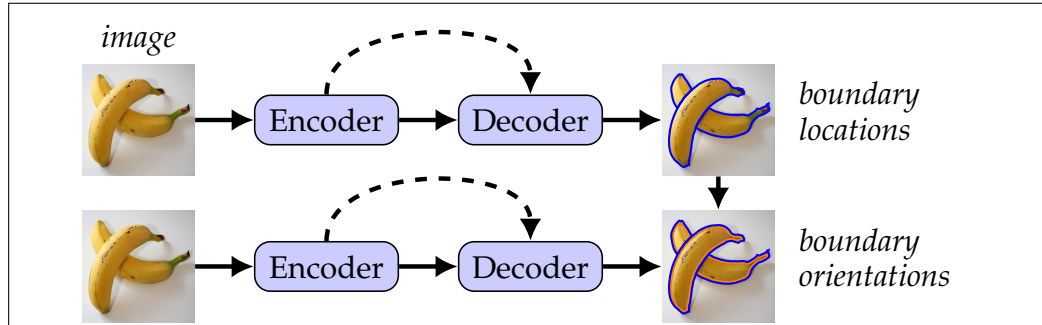
Our ablation study's baseline – *Late localization*
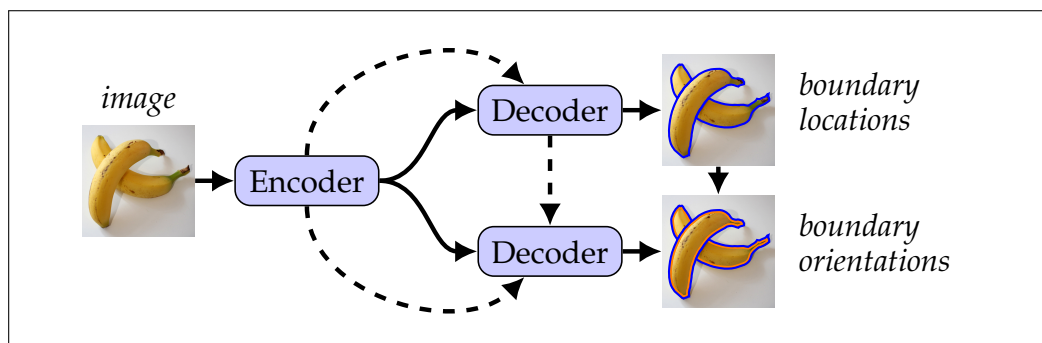
Bicameral (Ours) – *Late localization*

⟶ / ·····▸ / --▸ Sequential / holistically-nested / skip connections

Figure 4.5: Comparison with the state-of-the-art approaches, which rely on two independent streams for boundaries and occlusions respectively.

Experiment		Section	Figure(s)	Table(s)
Comparison with the two-stream baselines				
Late localization	1a	4.2.1	4.1, 4.6, 4.7	4.3
Early localization	2a	4.2.2	4.9	4.4
	2b		4.10, 4.12	4.5
Ablation study of the bicameral design				
Alternative architectures	1b	4.3.1	4.13, 4.14	4.6
	1c	4.3.2	4.15	4.7
Skip connections	1d	4.3.3	4.16, 4.17, 4.18, 4.19	4.8
	1e			
Encoder backbone	1f	4.3.4	4.21	4.9, 4.10

Table 4.1: Sections, figures and tables related to each experiment

Let us finally address some concerns that the reader may have regarding our experimental setup:

- We leave out the BSDS Border Ownership dataset [156] as it contains only 200 images, and mostly because the ground truth does not define instance boundaries but object part-level edges.

- We compare the late and early-localization paradigms for boundary and occlusion detection only on COCOA because amodal segmentation requires the amodal instance masks, which are not available in PIOD and not generated in Mikado. For training the proposed network on COCOA and comparing with amodal segmentation, we turn both the COCOA ground truth and the precomputed results of [206] into oriented instance boundaries.

Data Preparation   To robustly assess the generalizability of each model, each experiment is cross-validated using three folds, except for the amodal segmentation results as we use the precomputed binary outputs made publicly available by the authors. To present more significative results when comparing architectures, curves and scores are averaged on the three folds. For training, the networks are not directly fed with the original images but several sub-images randomly extracted from each original image, and augmented offline with random geometric transformations (flipping, scaling and rotation). Note that performances are not impacted by cropping given that the networks are fully convolutional. Table 4.2 details the folds for each dataset and the related experiments.
    We also point out some experiment-dependent details:

- Folds of Mikado are defined such that a texture appears only in one of the three subsets.

	PIOD [186]	**Mikado**	**COCOA** [206]
Training images	9,600	13,600	12,800
Validation images	800	800	1,424
Test images	800	4,800	1,323
Training iterations	18,000	34,000	24,000
Training epochs	15	20	15
In experiments	1a–f		2a–b

Table 4.2: Per-dataset folds for our cross-validation experiments after offline data augmentation (rotation, flipping, scaling)

- Folds of PIOD and COCOA are defined with respect to the initial split proposed by their authors. Specifically, the original training images are used for training or validation in our folds, and the original validation images for test. The original test images are never used as they are not publicly available.

- For fairly comparing with box proposal-based segmentation on Mikado, we use the Caffe implementation of [45], which made a Mask R-CNN-like [73] network publicly available. Just like ours, their network's encoder is based on VGG16 [168].

- For comparing with amodal segmentation, we use the precomputed binary outputs made publicly available by the authors. We derive the oriented boundaries from both the COCOA ground truth and their precomputed results alike: after intersecting the modal and amodal masks of an instance, amodal pixels that don't belong to the intersection are considered closer to the camera than the pixels of the intersection. This gives an orientation to the instance boundaries, *i.e.* the boundaries of the modal mask.

Training Settings For each dataset and each experiment, each network is trained and tested using Caffe [92], and the exact same settings (including fixed random seeds). At training time, we use the Adam solver [95] with $\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, a fixed learning rate of 10^{-4}, a weight decay of 10^{-4}, a ℓ_2 regularization, and a batch size of eight 256×256 images. The training images are randomly permuted at each epoch. As we solve a non-convex optimization problem, without

theoretical convergence guarantees, the number of training iterations is chosen for each dataset from an empiric analysis on training and validation subsets. As generally adopted, the optimization is stopped when the validation error stagnates or increases while the training error keeps decreasing. Please note that although the chosen stopping criterion may not be optimal for reaching the best performances on each dataset, it is however sufficient for significative comparative performances on a given dataset since each network in a comparison is trained under the exact same conditions. For all experiments, each network has its encoder initialized with weights pretrained on ImageNet [159], and its decoder(s) with the Xavier method [61]. The decoders are also equipped with dropout layers (with a ratio of 0.5) after each convolutional block at training time, to avoid overfitting.

Evaluation Metrics   As commonly adopted [5, 16, 17, 46, 68, 107, 125, 165, 192, 193] since [128], we compute at test time the precision and recall, and typical derived metrics: the best F-score on dataset scale (ODS) and the average precision (AP). Whereas ODS highlights one binarization threshold that gives the best compromise between recall and precision, AP conveys the area under the precision-recall curve over the full recall interval. For some experiments, we also consider the average precision in high-recall regime (AP_{60}), that is the precision averaged on the recall interval $[0.6, 1]$. As matching tolerance, *i.e.* the maximum ℓ_2-distance to the closest ground-truth pixel for a pixel predicted positive to be considered as a good hit, we consider a hard value of 0 pixels for Mikado (which contains perfect ground-truth boundaries) but a state-of-the-art value of $0.0075\sqrt{W^2 + H^2} (\simeq 2.7$ pixels in our evaluations) for PIOD and COCOA that contain approximate hand-made annotations, where $W \in \mathbb{N}^\star$ and $H \in \mathbb{N}^\star$ are respectively the image width and height. We perform evaluation without non-maximum suppression, which may artificially improve precision.

4.2   Comparison with the State of the Art

We now evaluate the proposed bicameral network by comparing, first, with the two-stream baseline for oriented boundary detection and, second, with the two-stream baseline for amodal instance segmentation.

4.2.1   Oriented Boundary Detection

We compare with the proposed bicameral structuring the two-stream baseline for oriented boundary detection [186] on PIOD [186] and

Figure 4.6: Comparative results for instance boundary (blue) and unoccluded side (orange) detection on PIOD (above) and Mikado (on the left). From top to bottom: input (a), ground truth (b), prediction using the two-stream baseline (c), using the proposed architecture (d). The proposed architecture improves the detection of fine details. Red rectangles highlight misdetections from the baseline corrected by our network.

Mikado (more details later in Chapter 5). Table 4.3 reports the comparative performances on these two datasets, and Figure 4.6 some qualitative comparative results.

| Architecture | PIOD [186] | | | | Mikado (Ours) | | | |
| | Boundaries | | Occlusions | | Boundaries | | Occlusions | |
	ODS	AP	ODS	AP	ODS	AP	ODS	AP
Two-stream baseline	.673	.708	.681	.733	.755	.832	.788	.872
Bicameral (Ours)	**.697**	**.738**	**.692**	**.747**	**.769**	**.847**	**.801**	**.884**

Table 4.3: Comparative best F-score on dataset scale (ODS) and average precision (AP) for instance boundary and occlusion detection on two datasets. The proposed bicameral architecture, which combines a shared encoder and cascaded decoders, outperforms the two-stream baseline.
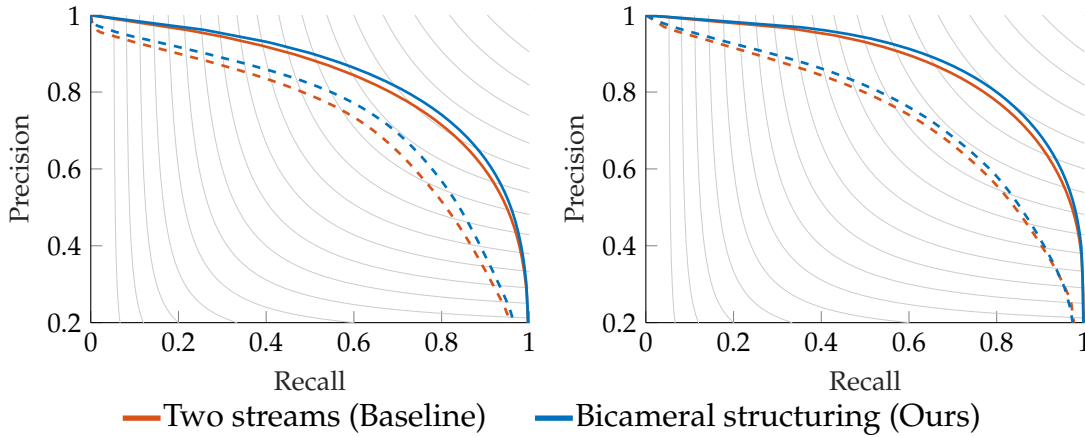
Figure 4.7: Comparative precision-recall curves for instance boundary (left) and unoccluded side (right) detection on PIOD (dashed lines) and Mikado (solid lines). Best viewed in color

Joint Representation Learning and Hierarchical Decoders   As shown by Figure 4.6, a bicameral network enables to expectedly eliminate a number of false positive predicted by the two-stream baseline. On PIOD, one can observe that some undetected boundaries near detected occlusions in the baseline's results are recovered using our network, as the notion of occlusions is learned jointly with the notion of instance. Inversely, the results on Mikado show that some boundaries misdetected by the baseline are removed where non-occlusions are instead detected by the proposed bicameral design. Figure 4.7 and Table 4.3 corroborate these observations. A gain of 3 and 1.4 points in AP for boundaries and occlusions respectively is achieved by the bicameral
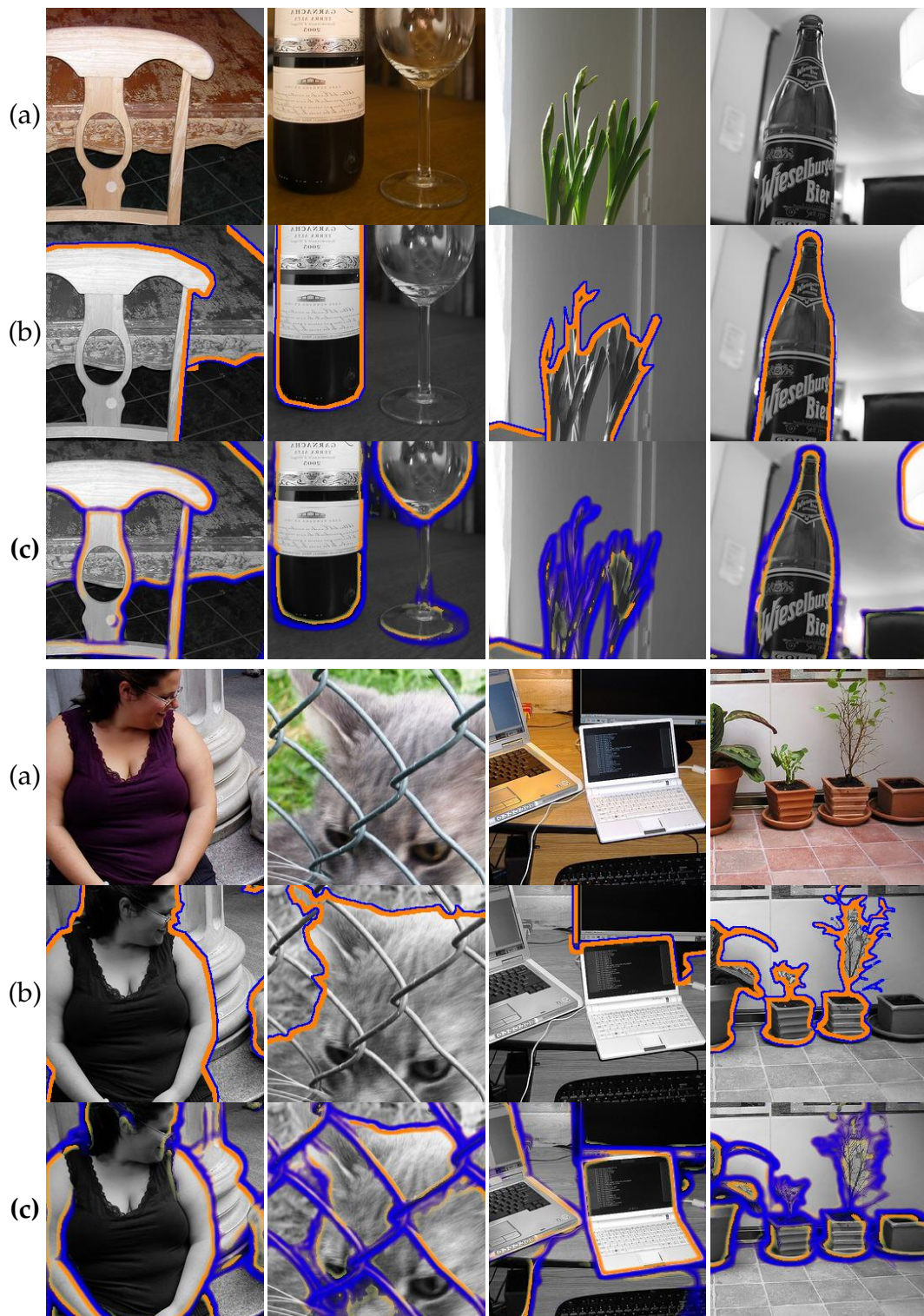
Figure 4.8: Instance boundaries (blue) and their unoccluded side (orange) detected using our architecture (c) on PIOD images (a), annotated by humans (b). The proposed network is able to fairly predict non-annotated boundaries and better delineate instances coarsely annotated by humans.

design over the baseline on PIOD, and more than 1 point on Mikado. These results confirm that **learning boundaries and occlusions from a joint feature space defines a consistent and performance-enhancing task**, while reducing the number of parameters by more than 25%.

Biases in the Hand-Made Annotations   One can also notice that the hand-made annotations hinder the training and evaluation due to their inaccuracy and incompleteness. As illustrated by Figure 4.8, **the bi-cameral network is able to fairly predict non-annotated boundaries**, *e.g.* internal boundaries of instances with holes, missing instances, or instances ambiguously considered as part of the background. Further-more, objects with complex shape, such as houseplants, which are often coarsely annotated by humans, are finely delineated by the proposed network. This can be explained by the regulatization term in the loss function, which enables to smooth the mapping learned from the noisy ground truth. As we will see in Chapter 5, resorting instead to syn-thetic training data for real-world applications proves more effective for learning generalizable invariants.

4.2.2   Amodal Instance Segmentation

We now take a step back to compare our late-localization approach with the early-localization paradigm. Specifically, we first compare on Mikado the boundary branch of the proposed bicameral network with a Mask R-CNN-like network [45] to evaluate each paradigm in the context of bin-picking scenes. We then compare the proposed bicameral structuring with the two-stream baseline for amodal instance segmentation [206] on COCOA [206].

Table 4.4 reports the comparative performances on Mikado, and Figure 4.9 some qualitative comparative results. Table 4.5 reports the comparative performances on COCOA, and Figure 4.10 some qualita-tive comparative results.

Late Instance Localization instead of Box Proposals   Box proposal-based segmentation approaches consist in first detecting scored rectan-gle regions that might contain an instance, then coloring the instance in each box proposal by binary segmentation. Such approaches prove effective for maximizing the detection recall in scenes where instances can be isolated in rectangles.

However, in bin-picking scenes, the bounding box of an instance is likely to be shared by the neighboring instances. This becomes problematic if a rectangle region contains multiple instances of the same object. Indeed, the binary segmentation of such a region implies
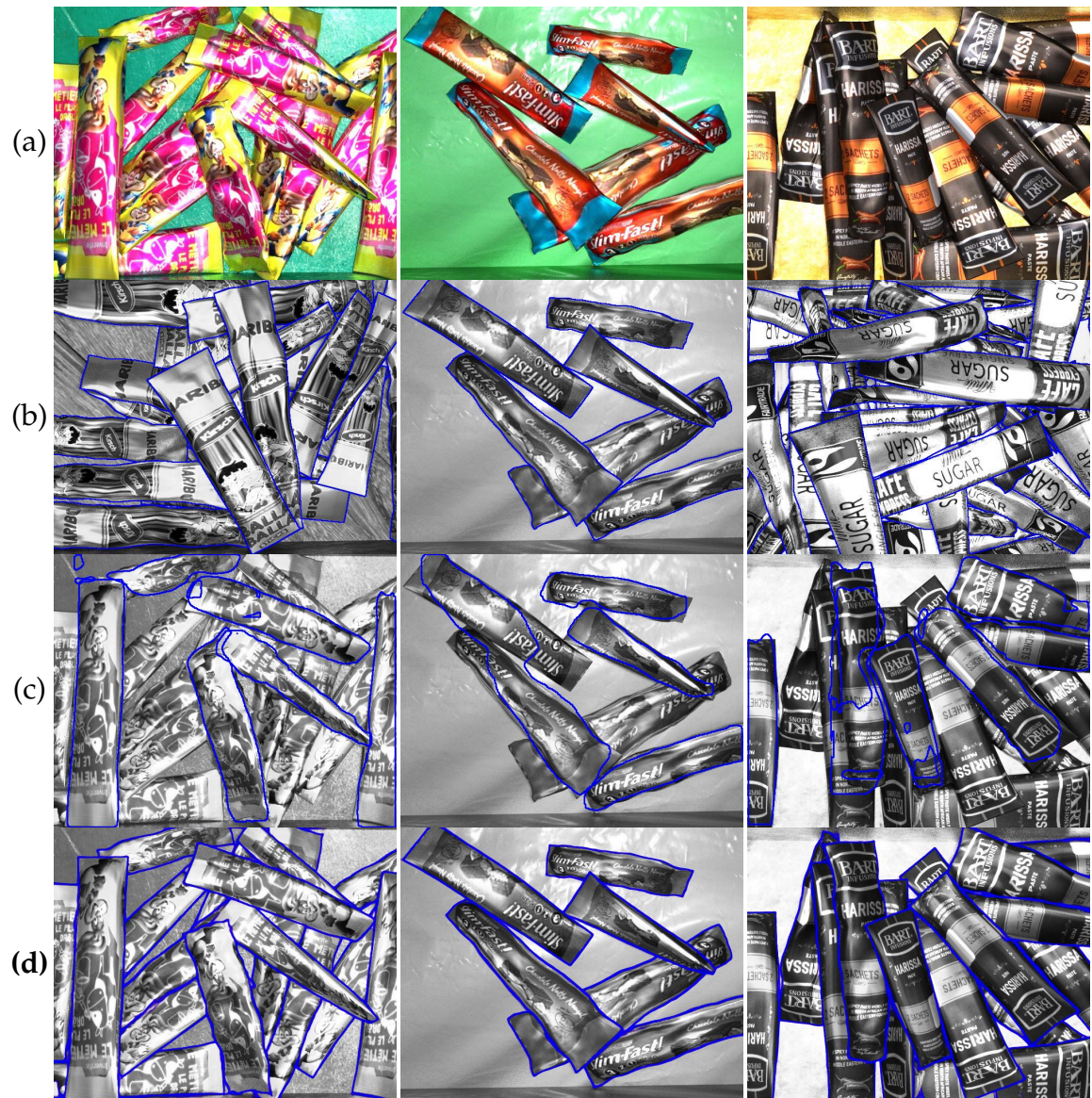
Figure 4.9: Comparative instance boundary detections on synthetic piles of sachets. From top to bottom: (a) input; (b) ground truth; (c) instance boundaries using a box proposal-based approach [45]; (d) using our approach. Best viewed in color

On Mikado (Ours)	Score threshold	Boundaries	
		ODS	AP
Box proposal-based segmentation [45]	0.25	.216	–
	0.50	.212	–
	0.75	.207	–
Bicameral (Ours)	–	**.809**	.885

Table 4.4: Unlike ours, a box proposal-based segmentation approach [45] gives inaccurate instance boundaries on dense piles of sachets.
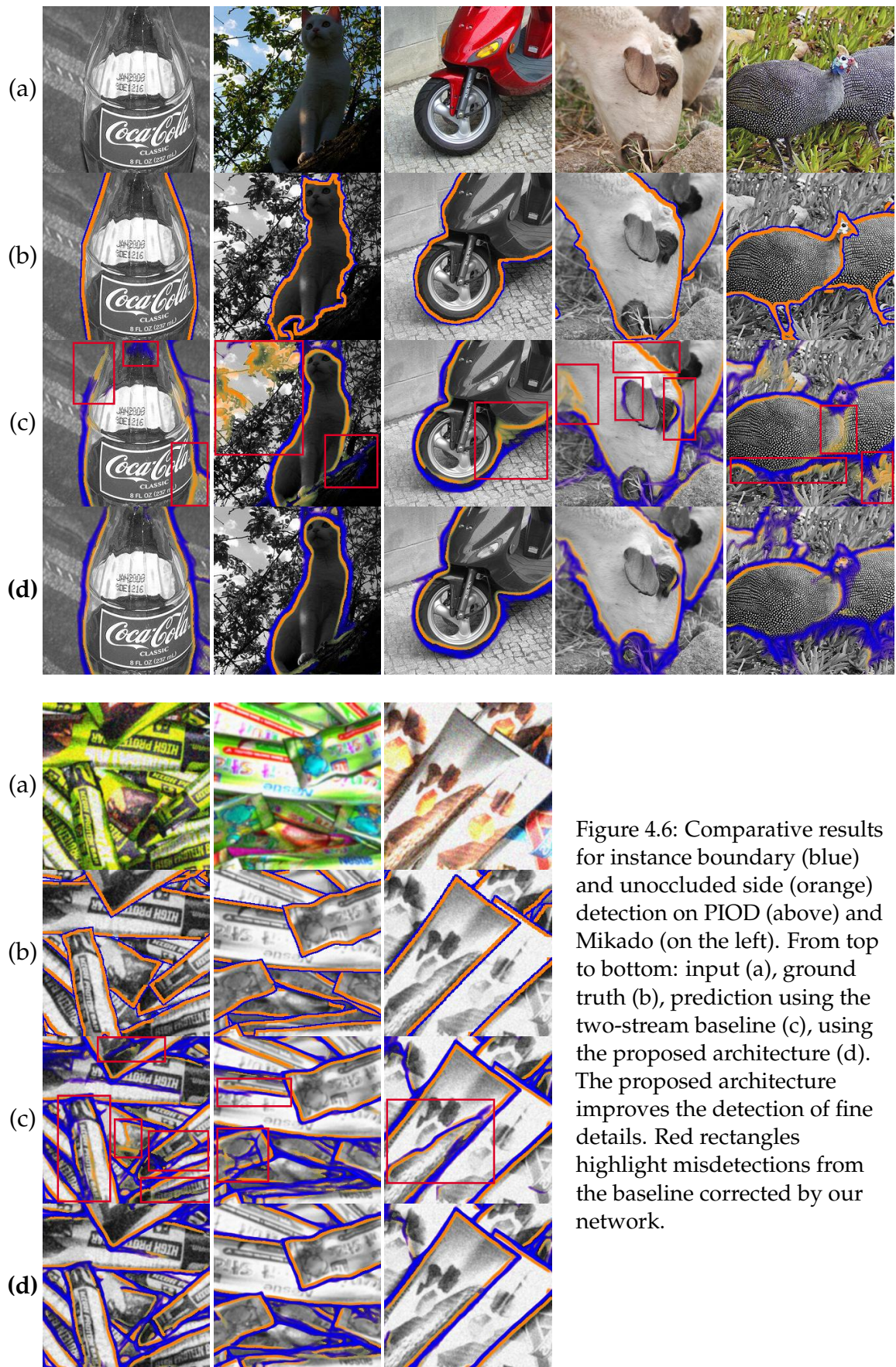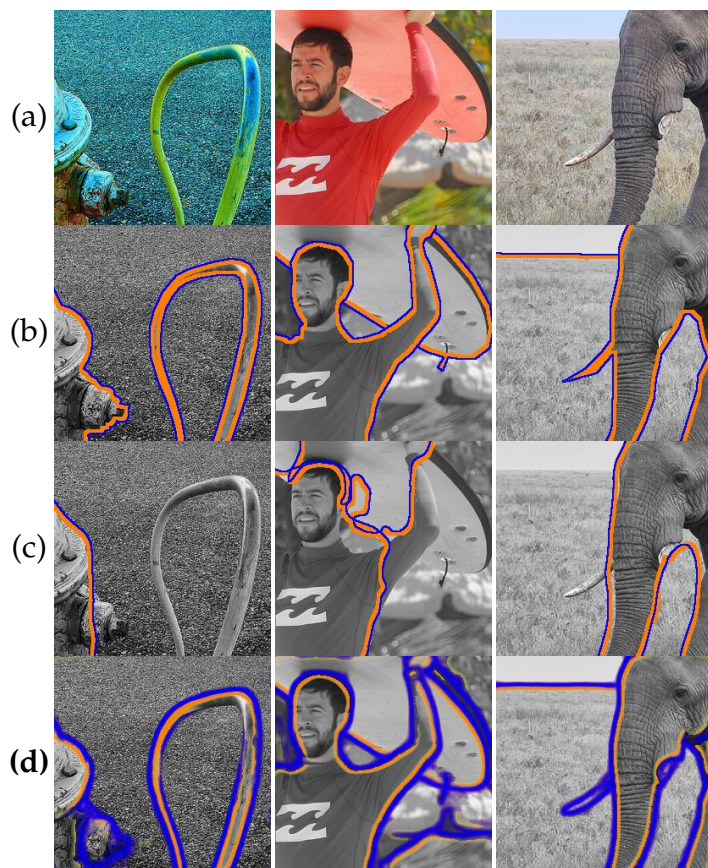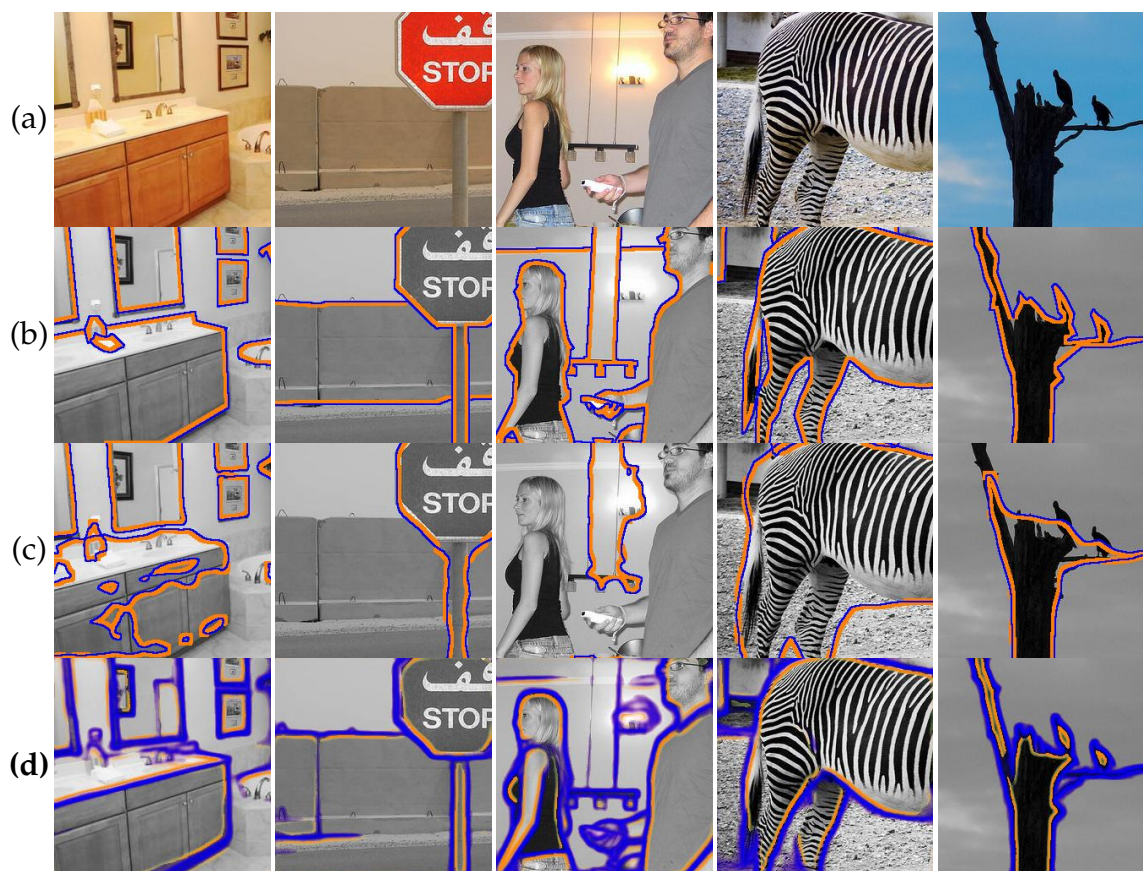
Figure 4.10: Comparative results for instance boundary (blue) and unoccluded side (orange) detection on COCOA (best viewed in color). From top to bottom: input (a), ground truth (b), prediction by amodal segmentation [206] (c), ours (d). Unlike the proposed approach, using a box proposal-based detection qualitatively leads to coarse segmentations and non-detected instances.

| Input | Things only | Things and stuff |
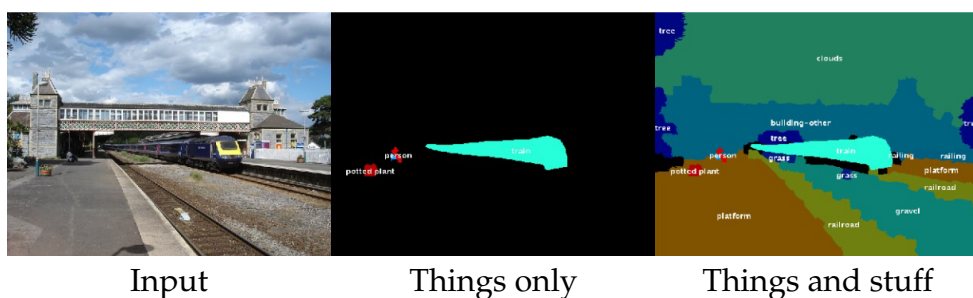
Figure 4.11: Example of thing and stuff annotations [29]

On COCOA [206]		**Boundaries**		**Occlusions**	
		ODS	AP	ODS	AP
All regions	Amodal segmentation [206][2]	.492	–	.529	–
	Bicameral (Ours)	**.666**	.694	**.637**	.673
Things[1] only	Amodal segmentation [206][2]	.536	–	.608	–
	Bicameral (Ours)	**.666**	.690	**.640**	.674
Stuff[1] only	Amodal segmentation [206][2]	.489	–	.397	–
	Bicameral (Ours)	**.687**	.727	**.648**	.693

[1] As llustrated in Figure 4.11, things are objects with well-defined shape (*e.g.* car, person) and stuff instances amorphous regions (*e.g.* grass, sky) [29].
[2] The evaluation is performed on the binary segment proposals made available by the authors.

Table 4.5: Comparative performances for instance boundary and unoccluded side detection on COCOA [206]. Whereas the proposed network equally performs on things and stuff, oriented boundary detection by amodal instance segmentation tends to focus on things and miss stuff instances.

—/- - -/······· Using the proposed network (all/thing/stuff)
●/■/▲ Using amodal instance segmentation [206] (all/thing/stuff)
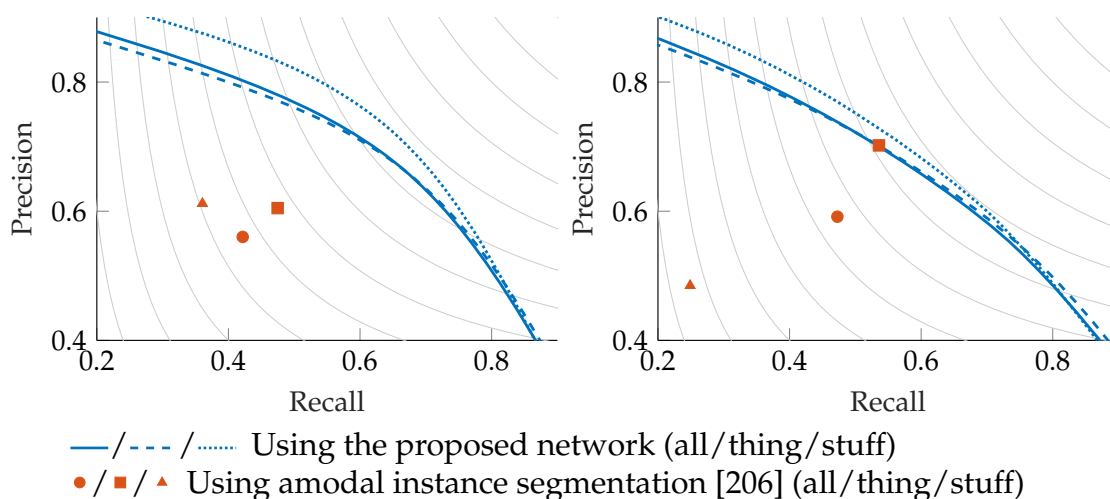
Figure 4.12: Precision-recall curves for instance boundary (left) and unoccluded side (right) detection on COCOA comparing bicameral structuring against amodal instance segmentation [206]. Best viewed in color

that similar patterns are classified differently inconsistently with the translation invariance property of deep convolutional networks. In this paradigm, the ambiguity of the binary segmentation can then only be broken by focusing on the global appearance of each object. Furthermore, the overall image segmentation is conditionned upon the region proposal network's performances.

As shown by Figure 4.9, this results in coarse or missing instance boundaries. By contrast, **the proposed approach enables to finely delineate each instance**, as our pixel-wise learning is free from any ambiguity. In accordance with these observations, Table 4.4 reports that our method outperforms a Mask R-CNN-like network [45] by near 60 points for detecting boundaries on Mikado, independently of the score threshold for filtering the proposals.

Oriented Boundaries instead of Amodal Masks   Amodal segmentation can be viewed as an overlay on the box proposal-based segmentation approach, aimed at introducing the notion of occlusion in addition to the notion of instance. Specifically, amodal instance segmentation [206] consists in two indepedent box proposal-based streams for respectively inferring the modal and amodal masks of each proposal, *i.e.* the mask of the visible instance parts and the mask including both the visible and occluded parts. An estimation of the occlusion rate of an instance can then be obtained by considering the intersection between the modal and amodal masks.

However, amodal segmentation approaches cumulate the drawbacks of box proposal-based segmentation discussed above, and the difficulty of coloring something invisible. Indeed, the ambiguity of binary segmentation raised when a rectangle region contains multiple instances of the same object is worsened when amodally coloring an instance, as the network must learn to activate "hidden" instance pixels which visibly belongs to another instance. Furthermore, when comparing the modal and amodal masks for estimating the occlusion rate of an instance, mismatches between the two masks are very likely to occur, thus inducing false positive in the boundary and occlusion maps.

As shown by 4.10, this results in inaccurate or missing boundaries and misdetected occlusions. By contrast, the proposed network overcomes these limitations by postponing instance localization: each pixel is jointly classified as boundary or not and assigned with an occlusion-based orientation independently of the instance bounding box it belongs to. Amodal segmentation also tends to favour "things", *i.e.* objects with well-defined shape, over "stuff" instances, *i.e.* amorphous regions such as grass or sky (see examples in Figure 4.11), while our method is agnostic to this categorization. Indeed, the region proposal network is

trained to coarsely highlight objects from the background. As a result, stuff instances are confused with parts of the background. These observations are corroborated by Table 4.5 and Figure 4.12 which report a gain in ODS of 17.4 points for boundaries and 10.8 points for occlusions over all instance types on COCOA using our approach. On things only, the proposed bicameral network outperforms the two-stream baseline by 13 and 3.2 points for boundaries and occlusions respectively. On stuff instances, the performance gains reach 19.8 and 25.1 points respectively.

In contrast with the proposal-based approach augmented with amodal segmentation **the late-localization approach proves more effective to recover comprehensive and accurate boundary and occlusion maps**, consistently with our design choices.

4.2.3   Conclusion

A bicameral FCN is defined as two cascaded decoders sharing a single encoder, with skip connections between the encoder and each decoder, and between the decoders as well, thus **enabling a fine delineation of each instance in both general scenes** (PIOD and COCOA) **and bin-picking scenarios** (Mikado).

The proposed network, which follows a late-localization paradigm, outperforms the two-stream baselines in both instance segmentation paradigms, *i.e.* oriented boundary detection [186] and amodal segmentation [206], on both real-world and synthetic data. Specifically, the performance gain reaches 2 points and near 20 points over the state-of-the-art late-localization and early-localization approaches respectively.

Trained on real-world data with biased hand-made annotations, a bicameral network is able to fairly detect non-annotated boundaries, thus suggesting that using instead synthetic training data with unbiased ground truth may enhance the performances.

4.3   Ablation Study

In Section 4.2, we showed that the bicameral structuring introduced in Section 4.1 outperforms the state-of-the-art two-stream approaches. In this section, we conduct a deeper analysis of the proposed bicameral architecture. Specifically, we explore alternative network designs and architectural components to better characterize the components of a bicameral network.

4.3.1   Alternative Architectures

We first compare the proposed bicameral structuring with alternative network architectures (see Figure 4.13) in order to better expose the benefits of sharing encoder representations and cascading decoders.

Shared encoder   *Cascaded* decoders   *Multi-task* decoder
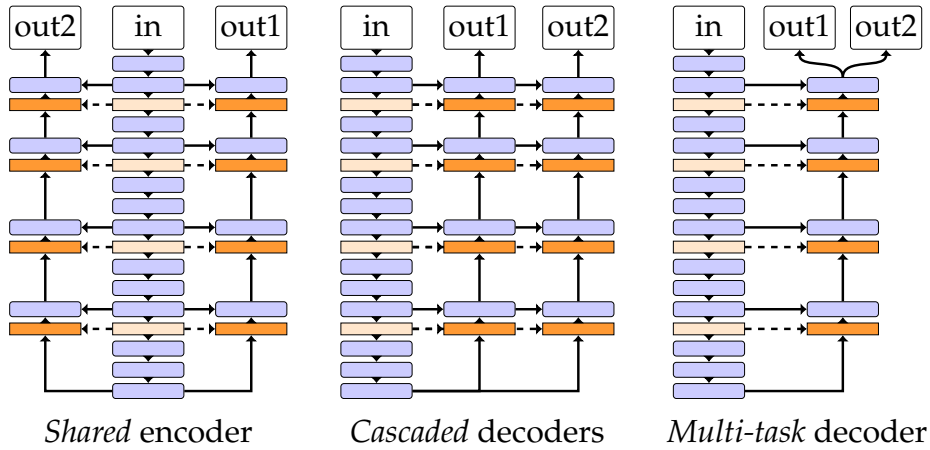
Architecture	Number of parameters
Baseline	46,839,938 ($\times 1.0$)
Shared	32,125,250 ($\times .69$)
Cascaded	29,949,250 ($\times .64$)
Multi-task	23,420,770 ($\times .50$)
Bicameral	34,301,250 ($\times .73$)

Bicameral
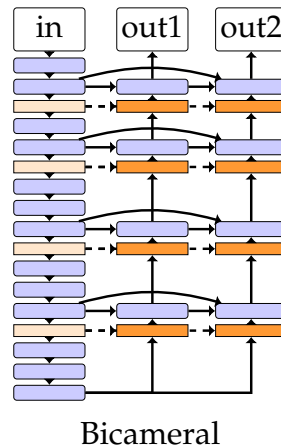
Figure 4.13: Alternative architectures also considered for jointly predicting boundaries and occlusions (best viewed in color), using a VGG16-based [168] encoder and a residual-like structure as decoder template. Legend is the same as Figure 4.2

Specifically, we compare the proposed bicameral architecture with three alternative architectures (*c.f.* Figure 4.13):

- a network with two independent decoders sharing the same encoder, referred to as *shared*;

- two decoders in cascade after one encoder, referred to as *cascaded*;

- a multi-task decoder that infers both boundaries and occlusions, referred to as *multi-task*.

Implementation For fair comparison, each network has the same VGG16-based [168] encoder, and is equipped with residual-like [75] connections, also referred to as skip connections, between the encoder and the decoder(s) for progressively combining local and global features when decoding. All the compared networks also share the same convolutional hyperparameters for their decoder(s). Specifically, for each decoder, the kernel of each convolutional layer is a 5×5 square, and the four convolutional blocks have, respectively from bottom to top, 256, 128, 64 and 32 filters. Note that we arbitrarily choose VGG16 as default backbone for our experiments, but any backbone is virtually suitable (*c.f.* Section 4.3.4).

Shared Encoding Features Instead of Independent Streams Our comparative experiments between single encoder-based designs and independent streams confirm that separating instances and inferring their spatial layout can be done with a single scene representation. Figure 4.14 shows that using a shared encoder, multi-task, cascaded or bicameral design instead of the two-stream baseline results in reaching lower boundary and occlusion test errors on both PIOD and Mikado. This is corroborated by Table 4.6 where the proposed bicameral and alternative architectures outperform the baseline by more than 2 points in ODS and AP, on both PIOD and Mikado for boundary detection.

Bicameral Structuring Instead of Alternative Designs A closer look at Figure 4.14 shows that the cascaded design reaches the lowest occlusion test error on both PIOD and Mikado. This suggests that chaining occlusion to boundary detection eases the backpropagation for occlusion prediction, as the decoder for occlusions may leverage a hierarchical feature representation of flat instance boundaries instead of undecoded image features. A gain in AP is achieved by cascaded decoders over the baseline (1.5 point up on PIOD, 1 point up on Mikado) but also the shared encoder design (1 point up on PIOD) for which decoders are independent. However, Table 4.6 shows that cascaded decoders are slightly less efficient for detecting boundaries on Mikado than a multi-task decoder or two independent decoders sharing the same encoder. This is explained by the impossibility of the occlusion
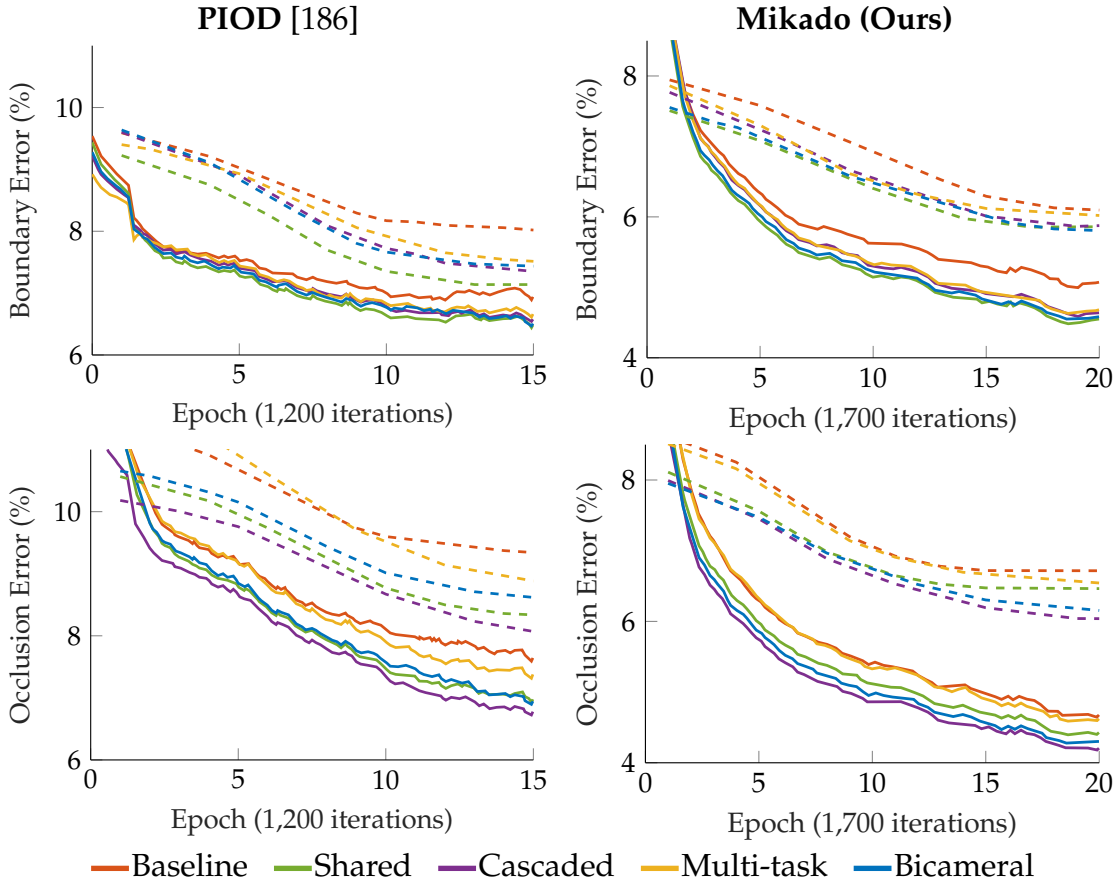
Figure 4.14: Comparative training (solid lines) and test (dashed lines) errors for instance boundary (top) and occlusion (bottom) detection on PIOD (left) and Mikado (right). Lower boundary and occlusion errors are reached when jointly predicting boundaries and occlusions (green, blue, yellow, purple) rather than independently (red).

| Architecture | PIOD [186] | | | | Mikado (Ours) | | | |
| | Boundaries | | Occlusions | | Boundaries | | Occlusions | |
	ODS	AP	ODS	AP	ODS	AP	ODS	AP
Baseline	.673	.708	.681	.733	.755	.832	.788	.872
Shared	.692	.732	.686	.738	.769	.847	.792	.876
Cascaded	.694	.735	.689	.748	.766	.844	.795	.880
Multi-task	.691	.731	.679	.731	.767	.845	.795	.880
Bicameral	**.697**	**.738**	**.692**	**.747**	**.769**	**.847**	**.801**	**.884**

Table 4.6: Best F-score on dataset scale (ODS) and average precision (AP) for instance boundary and occlusion detection on two datasets using different architectures. The bicameral decoder, which combines a shared encoder and cascaded decoders, outperforms the two-stream baseline and a multi-task decoder as well.

decoder to influence directly the encoder blocks. This trade-off is overcame by the bicameral design, which combines cascaded decoders both directly linked to the single encoder and has consequently the largest area under the precision-recall curve in Figure 4.7. The proposed bicameral structuring also outperforms a multi-task design. Table 4.6 notably reports that merging decoders limits the expressive power in favor of boundaries on PIOD. The obtained scores are well illustrated by the comparative predictions in Figure 4.6 where one can observe more closed boundaries and many false positive, mostly occlusions, predicted by the baseline and removed when instead decoding in cascade from a joint feature space.

4.3.2   Decoders Feature Sharing

We now assess whether feature sharing can apply to the branches of a bicameral decoder by comparing three hybrid architectures, namely *M3-B1*, *M2-B2*, *M1-B3*, that merge fully multi-task and bicameral decoder designs by sharing the three, two, one bottom-most layer(s) of their decoders respectively, as depicted by Figure 4.15. Intuitively, this results in sharing some task-dependent layers, as the two decoders are specialized in detecting boundaries and occlusions respectively.

Partially Shared or Independent Decoding Features?   A bicameral structuring outperfoms a multi-task design (*c.f*. Section 4.3.1) but one may also wonder whether partial feature sharing of the bottom-most bicameral decoder layers enables even better performances. Table 4.7 presents the performances obtained with three hybrid architectural variations between multi-task and bicameral designs (Fig. 4.15), each one introducing feature sharing at different levels of decoding. On PIOD, a bicameral decoder remains superior to all hybrid decoders, notably by about 1 point higher in ODS and AP for occlusions. On Mikado, sharing the first bottom decoder stage (M1-B3 design), which conveys object-level semantics, slightly improves performances. This suggests that the task specialization, for boundaries and occlusions respectively, may occur at a more local scale in decoding, either because of the higher density of inter-instance occlusions in Mikado, or due to the shape similarity of the Mikado instances. Unlike any of the hybrid designs, the bicameral decoder nevertheless achieves strong ODS and AP on both PIOD and Mikado. We thus advise to consider boundaries and occlusions separately by default after unpooling the encoder feature maps with lowest resolution.
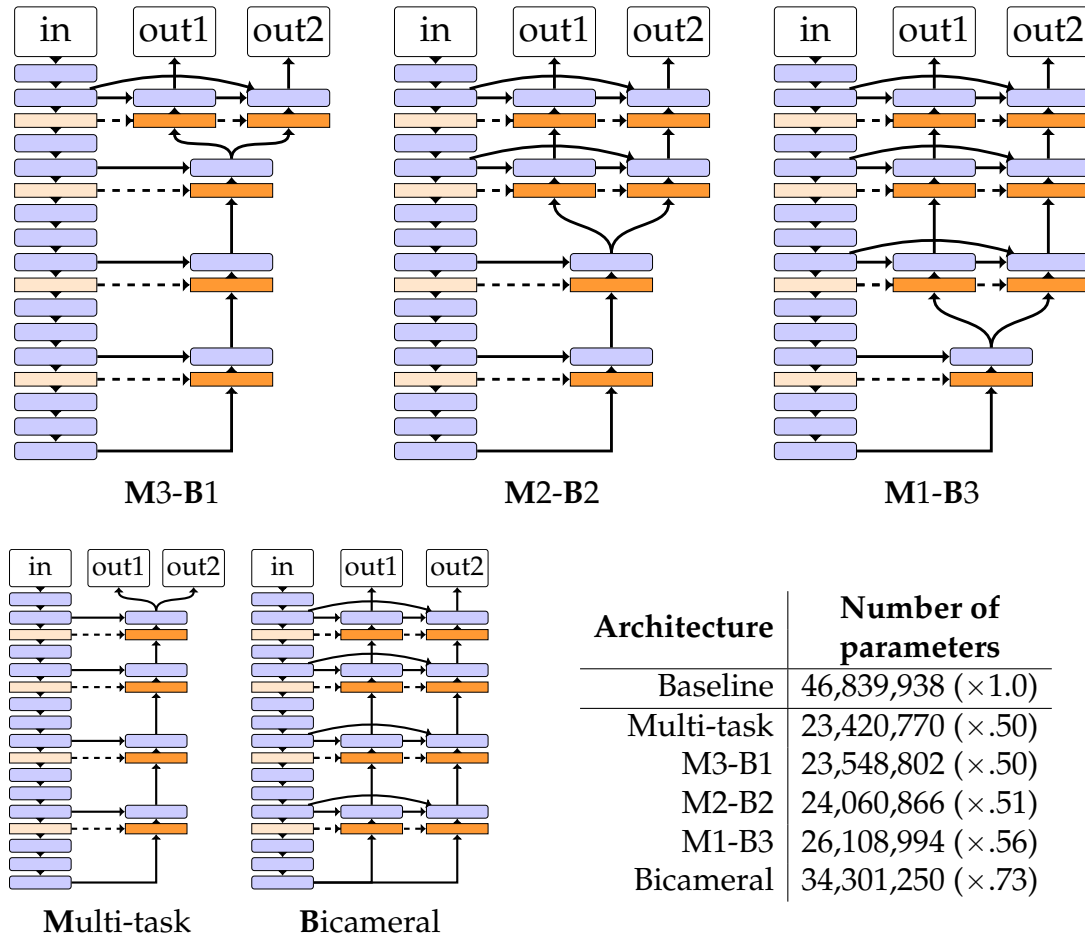
Figure 4.15: Hybrid architectures that merge multi-task ("M") and bicameral ("B") designs at different decoding stages, using a VGG16-based [168] encoder (best viewed in color). Legend is the same as Figure 4.2

Architecture	Number of parameters
Baseline	46,839,938 ($\times$1.0)
Multi-task	23,420,770 ($\times$.50)
M3-B1	23,548,802 ($\times$.50)
M2-B2	24,060,866 ($\times$.51)
M1-B3	26,108,994 ($\times$.56)
Bicameral	34,301,250 ($\times$.73)

Architecture	PIOD [186]				Mikado (Ours)			
	Boundaries		Occlusions		Boundaries		Occlusions	
	ODS	AP	ODS	AP	ODS	AP	ODS	AP
Baseline	.673	.708	.681	.733	.755	.832	.788	.872
Multi-task	.691	.731	.679	.731	.767	.845	.795	.880
M3-B1	.691	.735	.683	.734	.767	.845	.796	.879
M2-B2	.692	.738	.685	.740	.769	.848	.797	.881
M1-B3	.693	.737	.685	.739	**.771**	**.848**	**.802**	**.885**
Bicameral	**.697**	**.738**	**.692**	**.747**	.769	.847	.801	.884

Table 4.7: Best F-score on dataset scale (ODS) and average precision (AP) for instance boundary and occlusion detection on two datasets using different levels of layer sharing between the branches of a bicameral decoder.

4.3.3 Skip Connections

We now study the benefits of skip connections (SC) between the encoder and the decoders in a bicameral structuring.
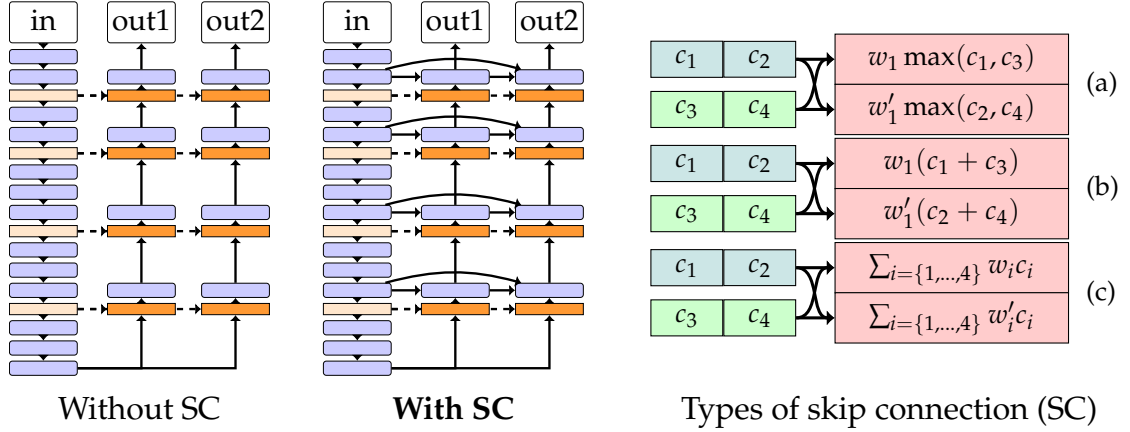
Figure 4.16: A bicameral structuring with and without skip connections (SC), and different types of skip connections: (a) using element-wise max operators, (b) using element-wise sum operators, (c) by concatenation.

As depicted by Figure 4.16, we compare bicameral designs with and without skip connections, and try out different types of skip connection: concatenation (our default choice for all the other experiments); element-wise max; element-wise sum. We choose concatenation by default because one can formally expect element-wise max and sum operations to be obtained using concatenation.

Indeed, let $N \in \mathbb{N}^\star$ be the depth of two layers to merge, and $e, d, f \in \mathbb{R}^N$ feature vectors respectively for the encoder, the decoder, and the resulting fusion. Let $w, w' \in \mathbb{R}^{N \times N}$ be trainable parameters. Using element-wise max operators, $\forall k \in \{1, ..., N\}, f_k = \sum_{i=1}^{N} w_{ik} \max(e_{ik}, d_{ik})$. Using element-wise sum operators, $\forall k \in \{1, ..., N\}, f_k = \sum_{i=1}^{N} w_{ik}(e_{ik} + d_{ik})$. Using concatenation, $\forall k \in \{1, ..., N\}, f_k = \sum_{i=1}^{N} (w_{ik}e_{ik} + w'_{ik}d_{ik})$. If needed, an element-wise sum operator can then be modelled by setting $w = w'$. Similarly, an element-wise max operator can be obtained by setting $w_{ik} = 0$ or $w'_{ik} = 0$ depending on which of the ith encoder or decoder channel has greater importance.

Skip Connections for Combining Local and Global Cues Partially hidden patterns are a major source of boundaries and occlusions. A perception at both local and global scales is however required to understand that an instance is partially occluded. By construction, an encoder-decoder network combines local and global cues by stacking convolutional and pooling/unpooling layers. This combination is

PIOD [186]

Mikado (Ours)
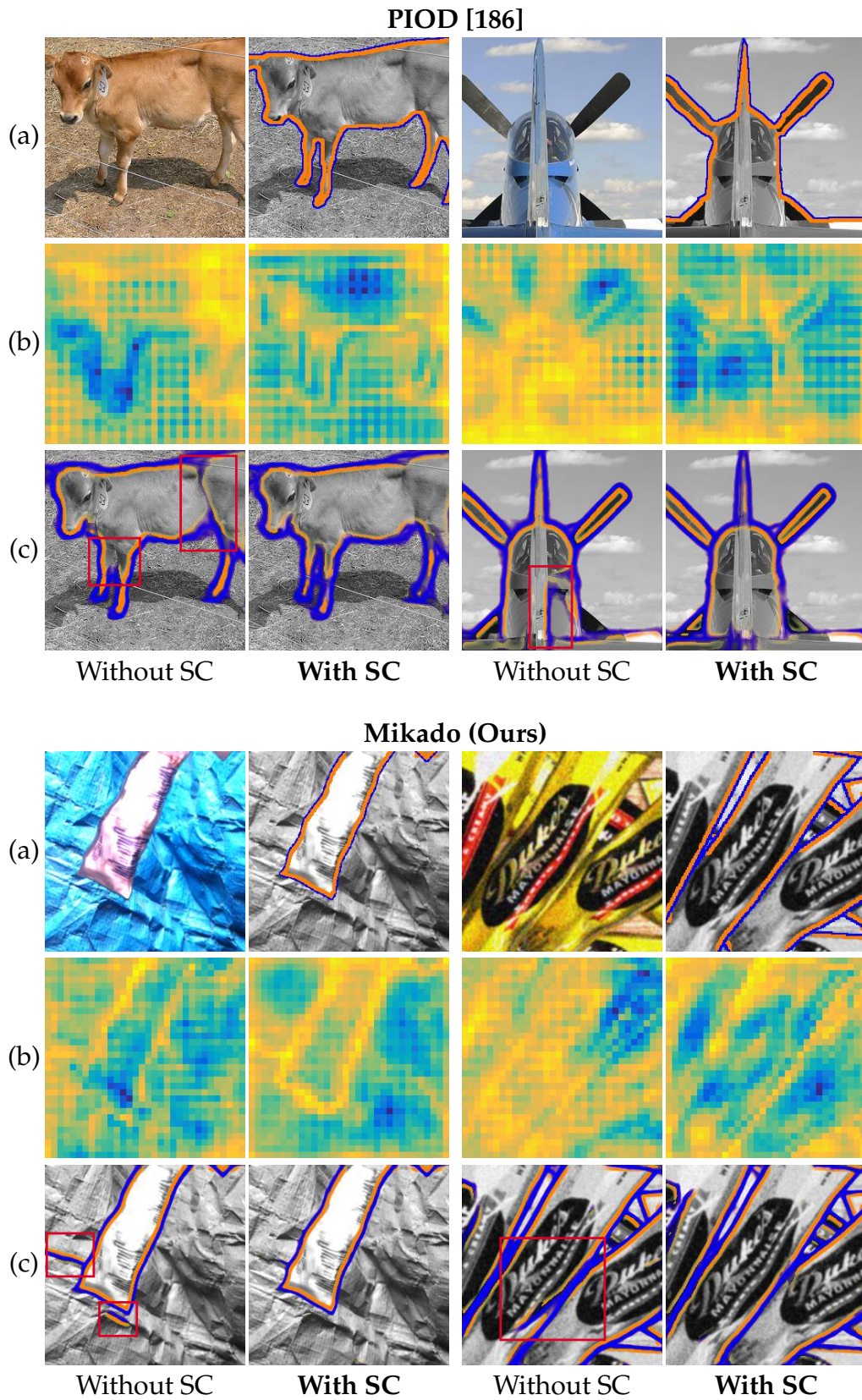
Figure 4.17: From top to bottom: (a) input and ground truth, (b) activation map after deconv4a (see Fig. 4.2), (c) final detection. Combining spatial information and higher-level semantics using skip connections (SC) between the encoder and decoders enables to detect instance boundaries earlier when decoding.

	PIOD [186]					
Skip connections?	**Boundaries**			**Occlusions**		
(Type)	ODS	AP	AP_{60}	ODS	AP	AP_{60}
No	.693	**.744**	.495	.692	**.749**	.520
Yes (Element-wise max)	.685	.729	.512	.676	.731	.522
Yes (Element-wise sum)	.687	.730	.505	.678	.731	.514
Yes (Concatenation)	**.697**	.738	**.517**	**.692**	.747	**.532**

	Mikado (Ours)					
Skip connections?	**Boundaries**			**Occlusions**		
(Type)	ODS	AP	AP_{60}	ODS	AP	AP_{60}
No	.759	.834	.686	.793	.878	.748
Yes (Element-wise max)	.755	.830	.676	.786	.871	.735
Yes (Element-wise sum)	.761	.838	.685	.791	.876	.743
Yes (Concatenation)	**.769**	**.847**	**.698**	**.801**	**.884**	**.758**

Table 4.8: Best F-score on dataset scale (ODS), average precision (AP) and average precision in high-recall regime (AP_{60}) for instance boundary and occlusion detection on two datasets using a bicameral FCN with and without skip connnections. Residual-like connections by concatenation between the encoder and the decoder(s) enable to better detect boundaries and occlusions as local and global cues are combined at each scale when decoding.
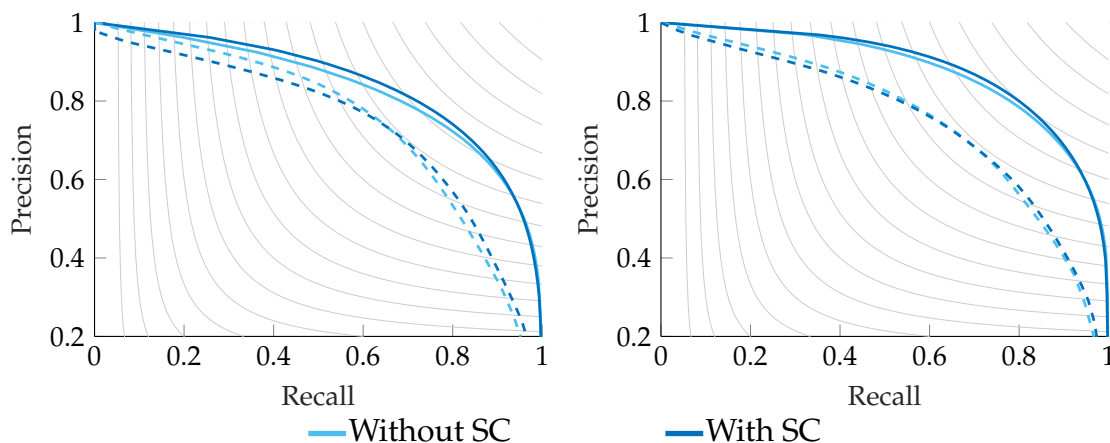
Figure 4.18: Comparative precision-recall curves for instance boundary (left) and unoccluded side (right) detection on PIOD (dashed lines) and Mikado (solid lines)
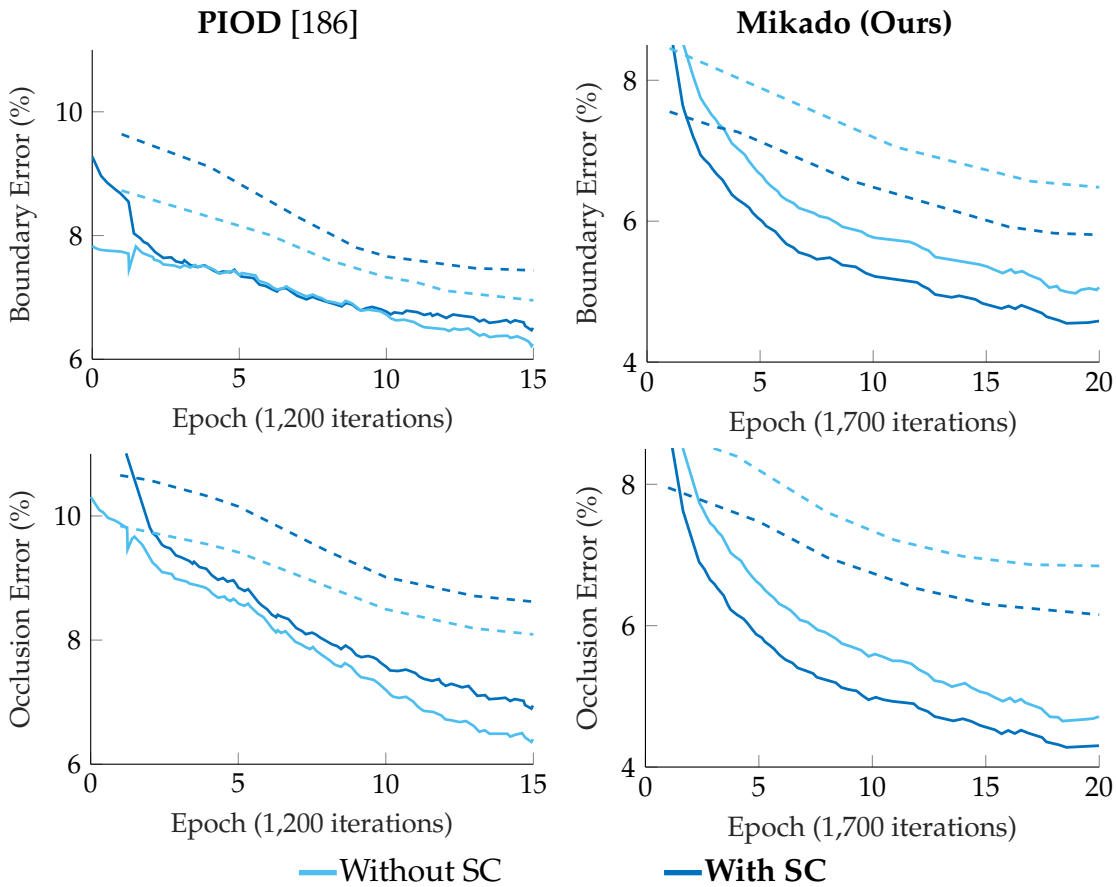
Figure 4.19: Comparative training (solid lines) and test (dashed lines) errors for instance boundary (top) and occlusion (bottom) detection on PIOD (left) and Mikado (right). Relatively to the initial error, a better error minimization is achieved when using skip connections between the encoder and decoder(s).
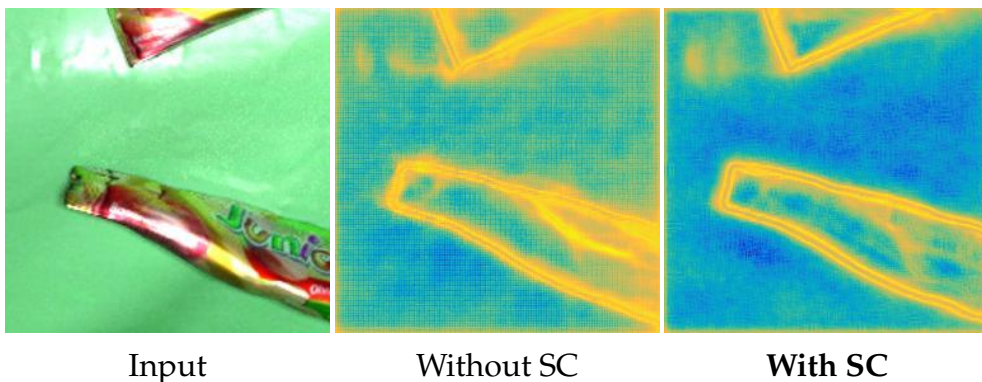
Figure 4.20: Feature maps after deconv1a (see Figure 4.2). Skip connections between the encoder and decoder strongly reduces the "checkerboard" artifacts [138] due to unpooling.

enhanced by residual-like connections at each scale between the encoder and decoder(s), as it enables to gradually combine the unpooled higher-level semantics with the spatial information lost after pooling. Figure 4.17 qualitatively shows what such connections bring: instance boundaries are detected earlier, thus giving the network more flexibility to adjust the following transformations and activations towards the desired output. These observations are corroborated by the scores in Table 4.8 and the precision-recall curves in Figure 4.18. A bicameral structuring with residual-like connections (*c.f.* Figure 4.16) outperforms a bicameral design without such connections by 1 point in ODS and AP on Mikado.

The obtained scores on PIOD are here impacted by the encoder initialization, which was obtained using a skip-connection free architecture. As a result, the backpropagation flow along the skip connections drastically reshapes the encoder from the first iteration, whereas the encoder of the skip connection-free design is only slighty affected by the backpropagation signals coming from the decoders. This mostly impacts the scores on PIOD because the encoders are initialized with weights pretrained on ImageNet, whose object type and context distributions are much closer to PIOD than Mikado. The skip connection-free design thereby starts to train on PIOD with already meaningful image features, unlike the bicameral design with skip connections on PIOD and both designs on Mikado. Figure 4.19 indeed reports that the bicameral network without skip connections starts with a lower training error on PIOD. Despite this disadvantage at training time, a bicameral design with residual-like connections shows a better precision in high-recall regime on PIOD, as shown by Figure 4.18. Table 4.8 confirms a gain of more than 1 point in AP_{60} when adding skip connections.

Concatenation Instead of Alternative Merging Operators   In all of our experiments, we consider skip connections by concatenation instead of alternative operators (Fig. 4.21), because we formally expect less constraints from concatenation. Table 4.8 confirms our expectation: concatenation produces better experimental results than element-wise sum or max operators. Enforcing sum or max operations indeed introduces arbitrary correspondences between the feature channels to merge. As a result, the low-level encoder activations may be overconsidered in the decoder, thus generating more false positives. Concatenation, as proposed, leaves more degrees of freedom for merging the channels, as each weight for their linear combination before activation is learned during backpropagation. Skip connections that turn out irrelevant can thus be switched off by the decoder, with near-zero weights.

4.3.4 Encoder Backbone

We finally investigate the use of a different encoder backbone. Specifically, we consider a deeper encoder chaining blocks of densely connected convolutional layers [87], referred to as *dense blocks*. As illustrated by Figure 4.21, a dense block is a generalization of a residual block [75], which introduces residual connections to give top layers access to earlier bottom layers. Introducing such skip connections between encoder layers enables to build and learn deeper representations as singularities due to zero incoming weights, overlapping incoming weights, or linearly dependent units are eliminated [139], while reducing the number of parameters (see Table 4.9).

Sequential [168]   Residual [75]   Dense [87]
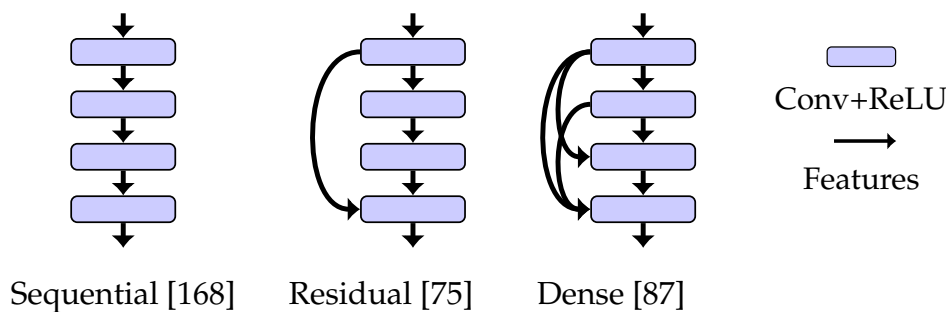
Figure 4.21: Types of convolutional block

Architecture	Encoder backbone	Number of parameters
Two streams (Baseline)	VGG16 [168]	46,839,938 ($\times 1.0$)
Bicameral (Ours)		34,301,250 ($\times .73$)
	DenseNet121 [87]	33,009,846 ($\times .70$)

Table 4.9: Number of parameters for a bicameral design with different encoder backbones

In our experiments, we choose VGG16 as default backbone because it limits the required memory, but any backbone is virtually suitable. To prove this claim, we compare bicameral designs with a VGG16-based and DenseNet121-based encoder respectively.

"Dense Grid" Interpretation   Following our network representation in Fig. 4.2, we point out that, if we describe skip connections in a dense block as "vertical" dense connections, then bicameral connections, *i.e.* skip connections between the decoders and the encoder, can then be interpreted as "horizontal" dense connections. A DensetNet121-based bicameral FCN thus results in a "grid" of densely connected blocks.

On PIOD [186]	Boundaries			Occlusions		
	ODS	AP	AP_{60}	ODS	AP	AP_{60}
Baseline (VGG16)	.673	.708	.476	.681	.733	.518
Bicameral (VGG16)	.697	.738	.517	.692	.747	.532
Bicameral (DenseNet121)	**.712**	**.761**	**.529**	**.714**	**.778**	**.556**

Table 4.10: Best F-score on dataset scale (ODS) and average precision (AP) for instance boundary and occlusion detection on PIOD [186] comparing the use of VGG16-based and DenseNet121-based encoder backbones. Plugging a bicameral decoder to a deeper encoder with dense blocks [87] enables to capture a finer representation of the object boundaries and nearby occlusions.

A Backbone-Agnostic Structuring   Using a deeper encoder composed of dense blocks, instead of sequential ones (Figure 4.21), the joint feature representation in a bicameral design reaches a higher expressive power. Table 4.10 reports a gain in AP of more than 6 points for boundaries and more than 4 points for occlusions over the two-stream baseline when building the bicameral encoder on DensetNet121 [87] instead of VGG16 [168] on PIOD. These results also suggest that a bicameral structuring may apply to any encoder backbone, whatever the depth and the type of convolutional blocks.

4.3.5   Conclusion

In this section, we conducted an ablation study of the proposed performance-enhancing fully convolutional structure. Our experimental analysis exposed the importance of the bicameral characteristics, *i.e.* single shared encoder, decoders in cascade, skip connections between the encoder and the decoders, and backbone-agnostic encoding.

- With any decoder configuration, sharing a single encoder representation for boundaries and occlusions outperforms a two-stream FCN for the same task.

- Additionnally cascading decoders, both linked to the encoder using skip connections, leads to the best trade-off between boundary and occlusion errors.

- Using skip connections between the encoder and decoders enables to recover boundaries earlier than without such connections. Skip connections by concatenation outperfom alternative types as they leave the largest number of degrees of freedom during training.

- The encoder's representative power can be enhanced by using densely connected convolutional layers.

4.4 Localizing Affordable Instances

Our bicameral network $f_{\mathbf{W}}$ is trained to infer boundaries and occlusions from a single image. However, in the context of robotic applications, **we need grasp coordinates on affordable instances to further interact with the real world**. In this section, we thus describe our procedure to localize the most affordable instance from the network inference.

4.4.1 Approach

Our approach is two-fold (see Figure 4.22): first, a boundary-based pixel clustering for generating a set of instance candidates; second, an occlusion-based ranking of the set of candidates.
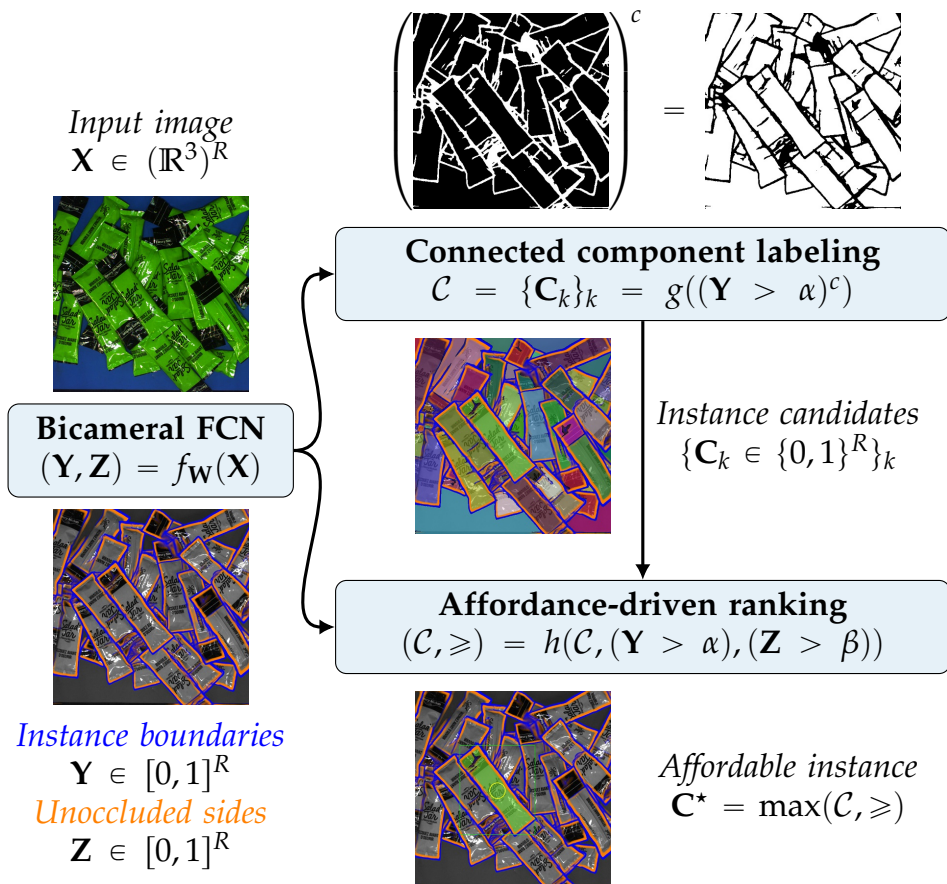
Figure 4.22: Our pipeline for inferring grasp coordinates centered on affordable instances from the bicameral network inference. First, a set of instance candidates is generated from the inferred boundaries $\mathbf{Y}$ by connected component labeling. Second, the set of candidates is ranked by estimating an occlusion-based affordance from the inferred unoccluded sides $\mathbf{Z}$.

Formally, let R be the image resolution, typically $R = W \times H$ for an image of width $W \in \mathbb{N}^{\star}$ and height $H \in \mathbb{N}^{\star}$, and $\mathbf{X} \in \mathbb{R}^{3 \times R}$ an image. Let g and h be the two parameter-free functions for localizing and ranking instances respectively, from the network inference $f_{\mathbf{W}}(\mathbf{X})$. We look for a discrete set $\mathcal{C} = \{\mathbf{C}_k \in \{0,1\}^R\}_k$ of instance candidates $\mathbf{C}_k$ ranked by occlusion-based affordance such that:

$$(\mathcal{C}, \geqslant) = h(g \circ f_{\mathbf{W}}(\mathbf{X}), f_{\mathbf{W}}(\mathbf{X})) \tag{4.3}$$

For two candidates $(\mathbf{C}, \mathbf{C}') \in \mathcal{C} \times \mathcal{C}$, $\mathbf{C} \geqslant \mathbf{C}'$ means that $\mathbf{C}$ is more affordable than $\mathbf{C}'$. We send to the robot the centroid coordinates of the most affordable instance $\mathbf{C}^{\star}$, defined as follows:

$$\mathbf{C}^{\star} = \max(\mathcal{C}, \geqslant) \tag{4.4}$$

4.4.2 Implementation

We now detail the definition and implementation of the two functions g and h introduced in Section 4.4.1.

Instance Localization   Let $(\mathbf{Y} > \alpha) \in \{0,1\}^R$ be the inferred boundary map $\mathbf{Y} \in [0,1]^R$ binarized using the threshold $\alpha \in [0,1]$. The function g takes the complementary of $\mathbf{Y} > \alpha$ as input and returns a set of binary maps $\mathbf{C}_k \in \{0,1\}^R$ corresponding to instance candidates such that:

$$\mathcal{C} = \{\mathbf{C}_k\}_k = g((\mathbf{Y} > \alpha)^c) \tag{4.5}$$

Our function g implements an off-the-shelf connected component labeling algorithm [65]. Each resulting connected component is considered as an instance candidate. In practice, for ensuring stable real-time performances, we nevertheless add a filtering step for removing overly small or large connected components, as the inferred network boundaries are sometimes noisy. An example of instance candidate generation is provided in Figure 4.23.

Instance Ranking   Let $(\mathbf{Z} > \beta) \in \{0,1\}^R$ be the inferred occlusion map $\mathbf{Z} \in [0,1]^R$ binarized using the threshold $\beta \in [0,1]$. The function h takes the set of candidates $\mathcal{C}$ and the binarized boundary and occlusion maps as inputs, and returns the ordered set $(\mathcal{C}, \geqslant)$:

$$(\mathcal{C}, \geqslant) = h(\mathcal{C}, (\mathbf{Y} > \alpha), (\mathbf{Z} > \beta) \tag{4.6}$$

Our implementation of h consists in, first, computing for each instance candidate $\mathbf{C}_k$ an occlusion-based affordance score $s(\mathbf{C}_k) \in \mathbb{R}$, then rank the set of candidates according to this affordance score. The affordance score $s(\mathbf{C}_k)$ computed for each candidate is defined such that:

$$\forall (\mathbf{C}, \mathbf{C}') \in \mathcal{C} \times \mathcal{C}, \ \mathbf{C} \geqslant \mathbf{C}' \Leftrightarrow s(\mathbf{C}) \geqslant s(\mathbf{C}') \tag{4.7}$$
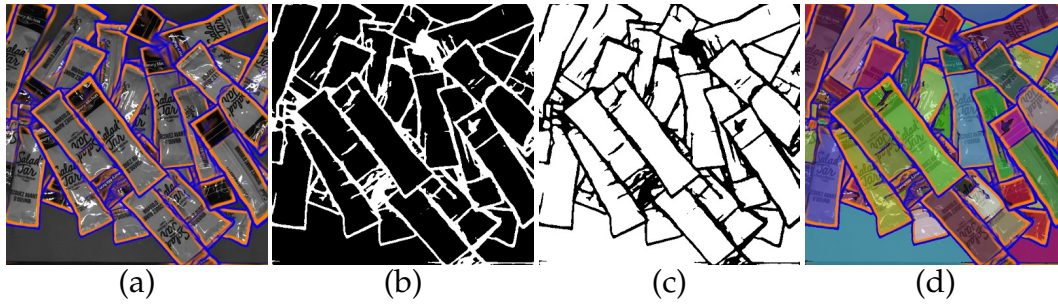
Figure 4.23: Example of instance candidate generation by connected component labeling [65]. From left to right: network inference (a); binarized boundary map (b); complementary of the binarized boundary map (c); resulting connected components (d). Best viewed in color

(a) An instance boundary pixel whose **outer side is occluded**, thus inducing a **high non-occlusion score** for this instance boundary pixel

(b) An instance boundary pixel whose **inner side is occluded**, thus inducing a **low non-occlusion score** for this instance boundary pixel

Figure 4.24: Computation of a local non-occlusion score at pixel location $\mathbf{p}$ by XNOR between neighborhoods $P_{\mathbf{p}}$ of the inferred occlusion map $\mathbf{Z}$ and an instance candidate mask $\mathbf{C}_k$ generated from the inferred boundary map $\mathbf{Y}$

Affordance Score Generally, the most affordable instance of a pile is most likely to be the one not occluded by any neighboring instances and on top of the pile. This translates into finding the instance $\mathbf{C}^\star$ whose boundaries are not caused by any occluded parts, and with largest area. If we note $s_1(\mathbf{C}_k)$ the ratio of boundaries not due to occluded parts and $s_2(\mathbf{C}_k)$ the area of a candidate $\mathbf{C}_k$, then we look for the most affordable instance $\mathbf{C}^\star$ such that:

$$\mathbf{C}^\star = \arg\max_{\mathbf{C}_k \in \mathcal{C}} \sum_{i=1}^{2} s_i(\mathbf{C}_k) \tag{4.8}$$

Specifically, we calculate the ratio $s_1(\mathbf{C}_k)$ of boundary pixels whose inner side is occluding the outer side, according to the network inference $\mathbf{Z}$, as follows:

$$s_1(\mathbf{C}_k) = \mu \sum_{\mathbf{p} \in \mathcal{P}} \underbrace{P_{\mathbf{p}}(\mathbf{C}_k \otimes (\mathbf{Z} > \beta))}_{\substack{\text{local non-occlusion} \\ \text{by element-wise} \\ \text{XNOR function}}} \underbrace{(\mathbf{Y_p} > \alpha)\mathbf{C}_{k\mathbf{p}}}_{\substack{\text{at each} \\ \text{boundary} \\ \text{pixel}}}$$

$$\mu = \underbrace{(|P| \sum_{\mathbf{p} \in \mathcal{P}} (\mathbf{Y_p} > \alpha)\mathbf{C}_{k\mathbf{p}})^{-1}}_{\substack{\text{normalization by the} \\ \text{patch area and perimeter}}} \tag{4.9}$$

where $P_{\mathbf{p}}$ is the patch extraction operator, which extracts a square neighborhood centered around the pixel $\mathbf{p}$, and $|P|$ the area of a patch. If $\mathbf{M} \in \mathcal{X}^R$ is a matrix, then $P_{\mathbf{p}}\mathbf{M}$ is the square submatrix of $\mathbf{M}$ centered around $\mathbf{p} \in \mathcal{P}$. If $v_1(\mathbf{C}_k) = 1$, then the candidate $\mathbf{C}_k$ is not occluded at all. If $v_1(\mathbf{C}_k) < 1$, then a ratio of $(1 - v_1(\mathbf{C}_k))$ boundaries are caused by occlusion(s) with neighboring instance(s). Figure 4.24 illustrates the computation of $P_{\mathbf{p}}(\mathbf{C}_k \otimes (\mathbf{Z} > \beta))$.

The area $s_2(\mathbf{C}_k)$ of an instance $\mathbf{C}_k$ is trivially defined as:

$$s_2(\mathbf{C}_k) = \sum_{\mathbf{p} \in \mathcal{P}} \mathbf{C}_{k\mathbf{p}} \tag{4.10}$$

4.4.3 Discussion

As depicted by Figure 4.25, the proposed approach gives good qualitative results in images densely populated with instances. A qualitative comparison with a box proposal-based approach [45] on non-annotated real images is also provided in Figure 4.26. We quantitatively evaluate our overall strategy, including the bicameral inference and the grasp generation, on a real-world robotic setup in Chapter 5.
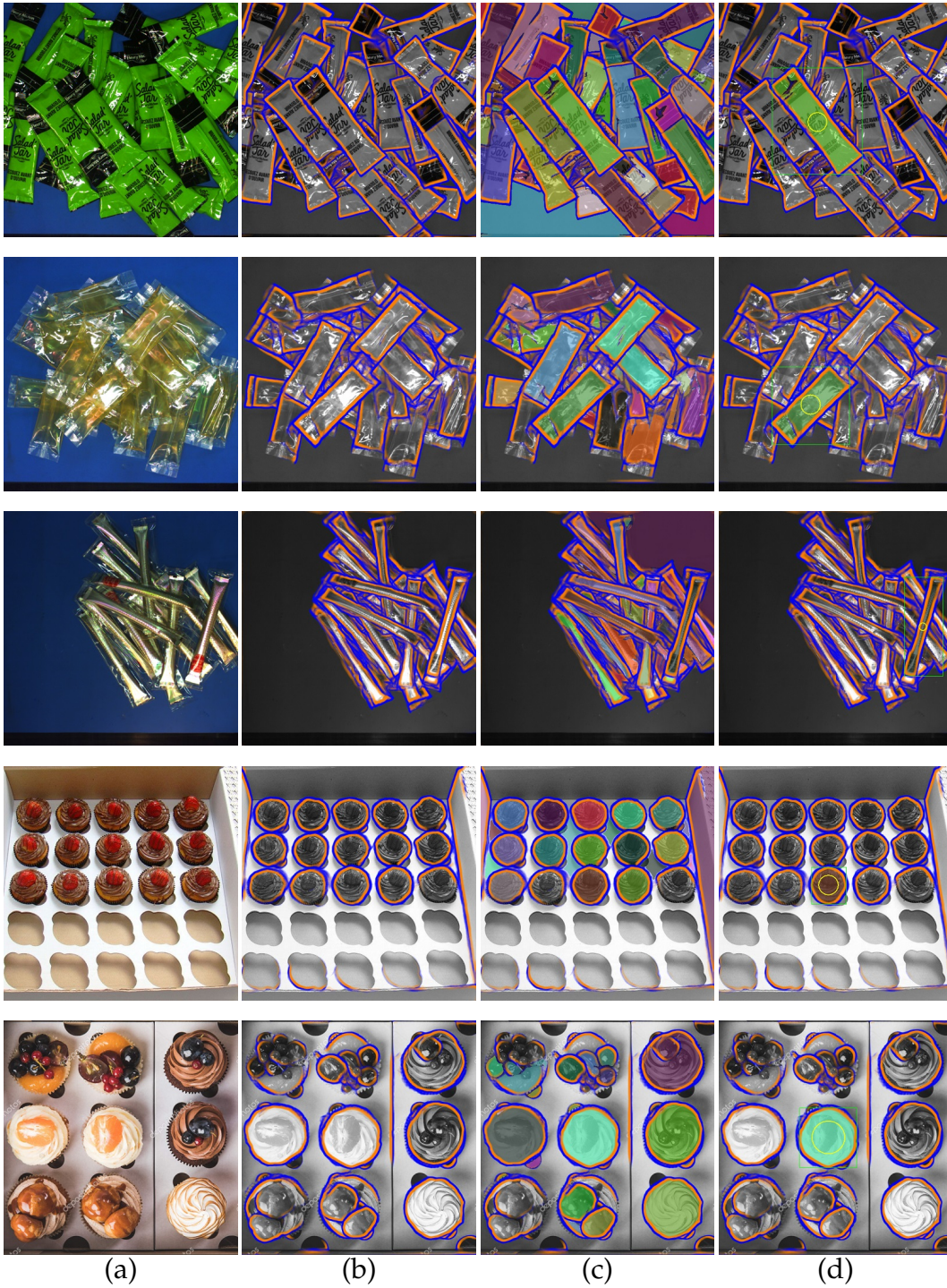
Figure 4.25: Qualitative results, and corresponding intermediate states, using our approach after training on our synthetic data (see later Chapter 5 for details). From left to right: real image as input (a) ; bicameral network inference (b); connected components (c); most affordable instance (d).

Figure 4.26: Comparative instance segment proposals after training on Mikado. From top to bottom: input (a); results using a proposal-based approach [45] (b); using the proposed approach (c). On (b), the rectangles are the detected box proposals. On (c), the rectangles and circles represent instance-centered parallel-jaw and suction-cup grasps respectively.

Hyperparameters In our pipeline for generating an ordered set of candidates $(\mathcal{C}, \geqslant)$, we introduced three hyperparameters: the thresholds α and β for binarizing the boundary and occlusion maps respectively, and the size of the considered neighborhood $P_{\mathbf{p}}$ when computing the affordance score. The thresholds α and β can be set both to 0.5 by default as the network is trained for binary classification. In practice, we empirically found that a lower value of 0.1 enables more stable performances, as it leads to more closed boundaries, thus grasp coordinates most likely inside an object (see Section B.1 for visualizing the impact of these thresholds). The size of the patch operator $P_{\mathbf{p}}$ is set relatively to the average thickness of the inferred occlusion edges, such that it covers this thickness in the four directions around the pixel location $\mathbf{p}$. In our experiments, we use a 9×9 patch operator.

4.4.4   Perspectives

Loss Functions The hyperparameters introduced in Section 4.4 are strongly dependent of the bicameral loss function which controls the quality of the inferred boundaries and occlusions (*c.f.* Section 4.1.2). In our experiments, we used balanced cross-entropy loss functions, but such loss functions induce thick and sometimes broken boundaries. Although thick boundaries are not a critical issue for instance segmentation, we are aware of recent works [43, 108, 197] that proposed alternative loss functions for learning crisper and thinner boundaries in the context of imbalanced distribution. Specifically, [43] combined the cross-entropy with the so-called Dice loss that additionnally compares the similarity between the two pixel sets. [197] performed edge alignment during training by solving a minimum cost bipartite assignment problem. In the application context of object detection, which faces the same class-imbalance issue, [108] introduced the so-called focal loss to put more focus on the hard misclassified training examples, thus giving sparser results. Other recent works addressed the lack of connectivity between the inferred boundary pixels [13, 133]. Specifically, [133] introduced a topology-aware loss function designed for minimizing the ℓ_2 distance between deep representations of the inferred and ground-truth boudary maps. [13, 133] also both proposed a refinement procedure that iteratively applies the same model over the previous delineation to refine the prediction at each step, but at the cost of multiple forward passes.

Instance Embedding Our implementation of function g (*c.f.* Section 4.4.2) for associating pixels into instances assumes that the connected components deduced from the binarized network inference are instance

(a) Oversegmentation
due to false positives

(b) Overclustering
due to unclosed boundaries
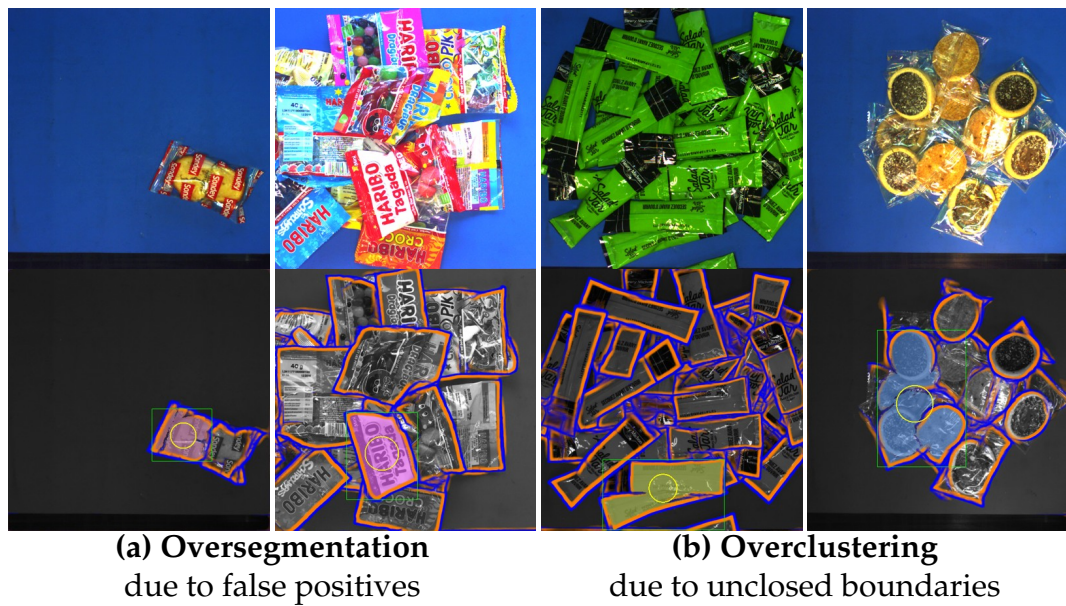
Figure 4.27: Real-world examples of failure cases due to false positives and unclosed boundaries, using the proposed approach. False positives induce instance parts as candidates (a), while unclosed boundaries grouped instances as candidates (b). A solution is to consider instance embedding [115, 137] for a smoother pixel clustering. Top: input; bottom: results.

Failure　　　　Success　　　　Failure　　　　Success
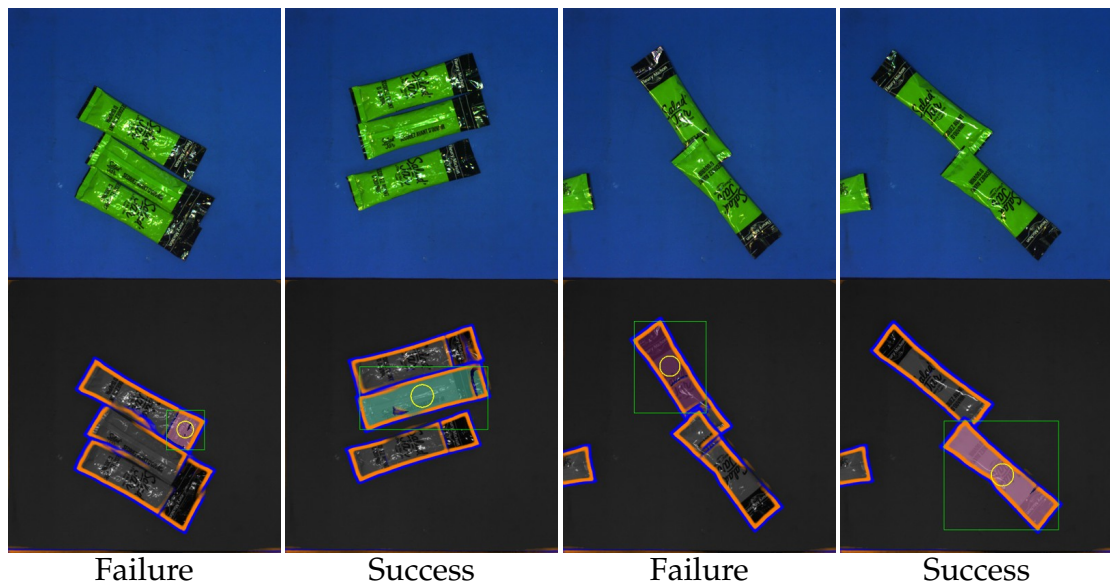
Figure 4.28: Real-world examples of non-equivariance cases, using the proposed approach. As boundaries and occlusions are not encoded independently of the corresponding instance poses despite data augmentation, similar scenes may lead to different results. A solution is to consider pose-equivariant learning structures [160]. Top: input; bottom: result.

candidates. In practice, the inferred boundaries may not be closed for some instances, thus inducing groups of instances as instance candidates. Conversely, false positives may lead to oversegment some instances, thereby generating instance parts as instance candidates (see Figure 4.27). Interestingly, recent works suggested that the convolutional pixel representations can be augmented with a distance to an implicit reference point of the instance to which they belong [115, 137]. Considering a pixel clustering based on the pixel features including this instance embedding, instead of binary decisions, should enable a smoother function for generating instance candidates.

Pose Equivariance   As deep convolutional networks are not rotation-equivariant, nor scale-equivariant, boundaries are not encoded independently of the instance poses although our training data is accordingly augmented with geometric deformations. As a result, similar scenes, that differ only by visually minor translations, rotations or scaling, may lead to very different detections from the bicameral network. Figure 4.28 illustrates cases of this lack of equivariance. Implementing instead a bicameral structuring from pose-equivariant learning modules such as [160] is likely to introduce this property in the learning.

4.5   Conclusion

4.5.1   Summary

In this chapter, we proposed a novel fully convolutional structure, referred to as *bicameral*, for jointly inferring instance boundaries and their occlusion-based orientation from a single image. The proposed bicameral network is composed of a single encoder embedding boundaries and occlusions in a joint feature space, two cascaded decoders for recovering boundaries and occlusions respectively, linked altogether by skip connections.

Specifically, we first detailed the proposed architecture, in contrast with the state-of-the-art approaches for oriented boundary detection [186] and amodal instance segmentation [206]. We evaluated the proposed network on the related state-of-the-art real-world datasets, *i.e.* PIOD [186] and COCOA [206], and a synthetic dataset, namely Mikado, which better adresses the scenario of instances piled up in bulk in accordance with our application context (see Chapter 5 for details on the generation and plausiblity). We then conducted an ablation study of the proposed architecture to highlight the role of each of his components, including the relative layout of the encoder and decoders, the presence of skip connections, and the encoder's backbone.

In anticipation of detecting the most affordable instance of a pile in high-throughput bin-picking applications, we finally proposed a simple scheme to translate the bicameral network inference into relevant grasp coordinates, based on independent local operations on the inferred boundary and occlusion maps.

4.5.2 Contributions

Unlike the state-of-the-art approaches for both oriented boundary detection and amodal instance segmentation, which rely on two independent streams for boundaries and occlusions respectively, jointly encoding boundaries and occlusions in a single representation enables to leverage the strong link between boundaries and occlusions in scenes of many instances on top of each other.

As a result, **the proposed bicameral network achieves state-of-the-art performances on both real-word and synthetic data**. For detecting boundaries, the proposed method outperforms the two-stream baseline for oriented boundary detection by more than 3 points on PIOD, and 1 point on Mikado, and the two-stream baseline for amodal segmentation by near 20 points on COCOA. For occlusions, the performance gain reaches more than 1 point and 10 points for oriented boundary detection and amodal segmentation respectively.

Our ablation study showed that, **compared with alternative layouts of the encoder and decoders, the proposed bicameral structuring proves the most performance-enhancing** on PIOD and Mikado. Skip connections between the encoder and each decoder enable to efficiently combine low-level and higher-level semantics when decoding boundaries and occlusions from the encoder representation. Skip connections between the decoders enables a better joint error minimization. Building a bicameral network from a very deep encoder composed of densely connected convolutional blocks increases the performance gain to 6 and more than 5 points on PIOD for boundaries and occlusions respectively.

The proposed method for detecting the most affordable instance of a pile from a synthetically trained bicameral network inference proves qualitatively efficient on non-annotated real images densely populated with instances, while enabling a highly parallelizable implementation. These observations thus suggest promising quantitative results for bin-picking applications. In Chapter 5, we further investigate synthetic training data for real-world bin-picking, then quantify the proposed method on a real-world robotic setup.

Chapter 5

Application to Bin-Picking

In Chapter 4, we presented a performance-enhancing "bicameral" convolutional structure for jointly inferring instance boundaries and occlusions, and consequently detecting the most affordable instance of a pile from a single image. In this chapter, we want to apply this bicameral network to real-world bin-picking applications.

Real image → Human annotators ~**30min per image** → Annotated real image

(a) Common approach based on hand-made annotations

Human supervisor → Physics and render engines ~**5min per image** → Annotated synthetic image

(b) Our training data generation pipeline
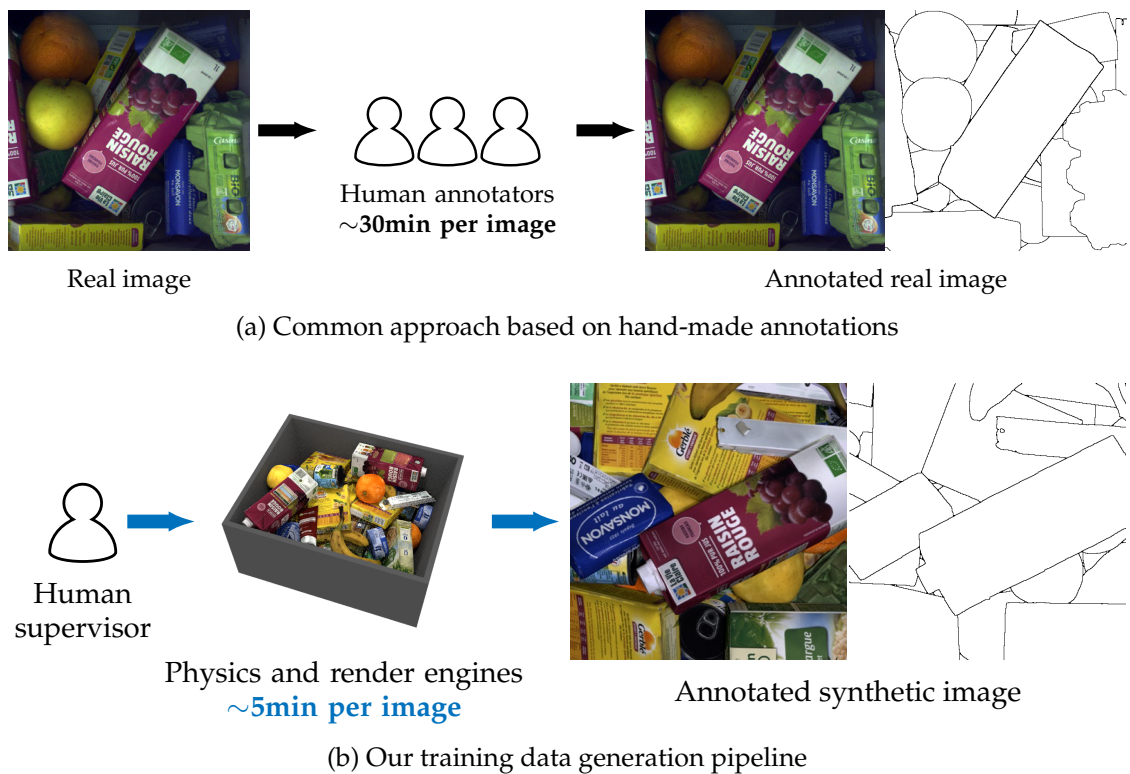
Figure 5.1: Our simulation-based approach for generating training data (b) compared with the common approach that relies on manual annotations (a). See also video "mikado01" in supplementary material.

However, there is no real-world training dataset for large-scale bin-picking and annotating real images with oriented boundaries is tedious and time-consuming. As illustrated by Figure 5.1, we thus propose to synthetically train our bicameral FCN.

5.1   Synthetic Training Data

In this section, we describe our pipeline for generating synthetic training data from off-the-shelf rendering and physics engines.

5.1.1   Data Generation

We generate synthetic data using custom code on top of Blender [21] by simulating scenes of objects piled up in bulk and rendering the corresponding top views, as depicted in Figure 5.2. More precisely, after modelling a static open box and, on top, a perspective camera, some mesh templates are instantiated in random initial pose and successively dropped above the box using the Blender's physics engine (a video showing the generation of a scene is provided in supplementary material). We then render the camera view, and the corresponding depth image, using the Cycles rendering engine. In this configuration, we ensure a wide pose variability, a lot of occlusions between instances, and the ground-truth occlusion boundaries can be trivially derived from depth.

In our application context, we consider piles of many instances with inner variability and using only RGB. Specifically, we target the real-world scenario of unpiling packaged food products. We thus generate RGB images of food products piled up in bulk by randomly applying local and global deformations to mesh templates that we texture successively from a set of texture images retrieved using the Google Images search engine[1] and manually cropped to remove any background. Each scene is composed of many instances using the same texture image as we target homogeneous piles. Besides, to prevent the network from simply substracting the background, we apply to the box a texture randomly chosen among a set of background images, retrieved using the Google Images search engine as well. Between each image generation, we also randomly jitter the cameras and light locations to prevent the network from learning a fixed source of light, and so fixed reflections and shadows. Note that the proposed pipeline is not specific to the case of food products.

---

[1]https://images.google.com/

Inputs:
Mesh template
Texture images

Physics simulation of piles of sachets

Top-view RGB-D camera rendering

Training and test data preparation
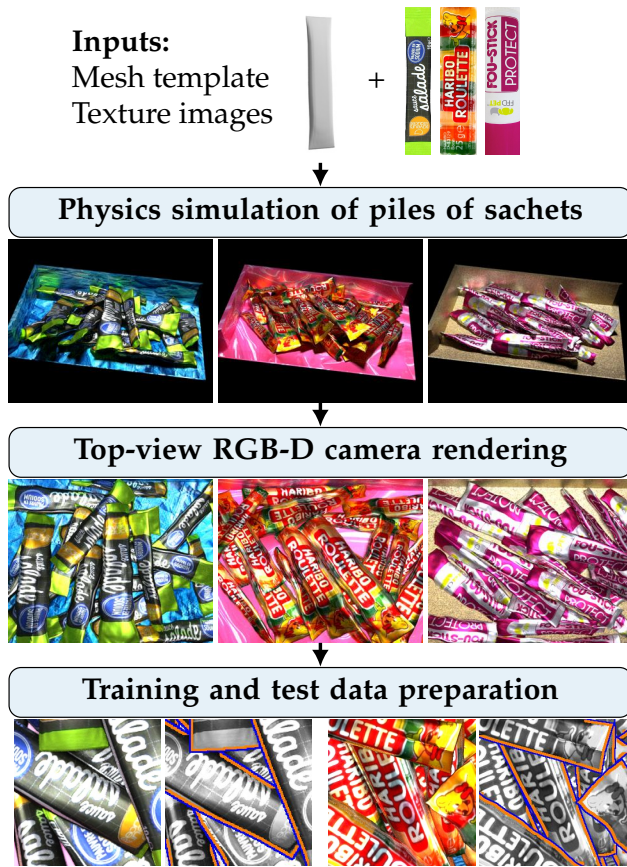
Figure 5.2: Overview of the Mikado pipeline (best viewed in color). Given a mesh template and texture images, piles of deformed instances are generated using a physics engine. A top-view camera is then rendered to capture RGB and depth. The synthetic images and their annotations (ground-truth boundaries are in blue, unoccluded side in orange) are finally prepared to be fed-forward through the network. See also video "mikado02" in supplementary material.

Top-view RGB-D camera rendering

RGB   Segmentation   Depth

Generating ground-truth boundaries and occlusions

Instance   Local depth-based   Boundaries and
boundaries   segmentations   unoccluded side

Training and test data preparation
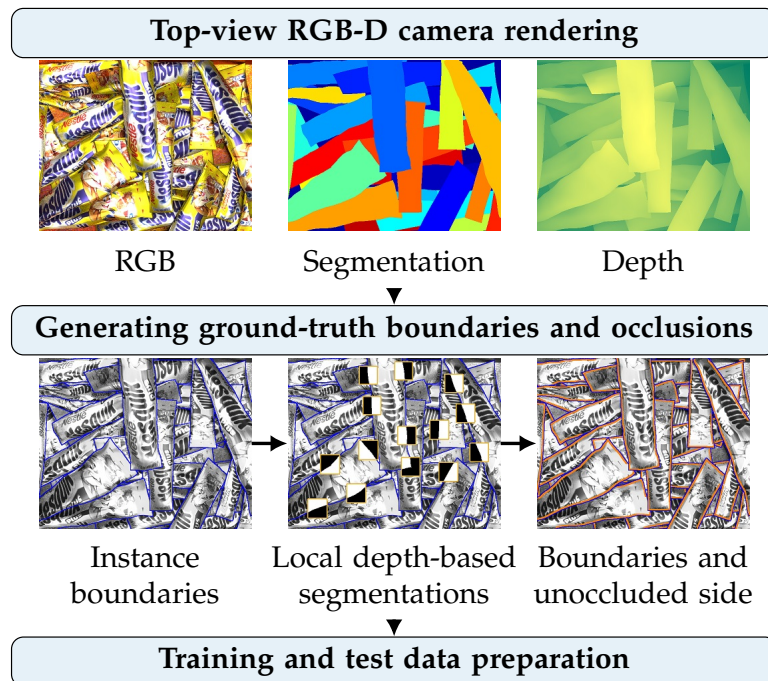
Figure 5.3: Pipeline for generating the ground-truth boundaries and occlusions (best viewed in color). At each boundary pixel, a depth-based binary segmentation of the neighborhood is performed to label each side, such that the higher side is set to 1 and the lower side to 0. In the end, the ground truth consists of instance boundaries (blue) and their unoccluded side (orange).

For our experiments, we generate two datasets: *Mikado* and a large extension referred to as *Mikado+*. Mikado and Mikado+ differ by the number of images but also the mesh templates, textures and backgrounds used in simulation (see Table 5.1). Samples and characteristics of the two datasets, compared with the amodal dataset of [55], referred to D2SA, which contains real images of supermarket items annotated with the modal and amodal masks, are provided in Figure 5.4 (see Figures C.1 and C.2 for a comprehensive overview of our textures and background images). The largest generated dataset finally comprises on average up to 31.5 instances per image, hence 6 times more instances and 22 times more inter-instance occlusions per image than D2SA.

5.1.2   Data Augmentation

As our RGB images are generated using heuristic rendering models, the training and evaluation may be biased by a lack of realism in the sense that the simulated camera models limitedly synthesize the properties and imperfections of real-world sensors. However, we want the proposed network to capture generalizable invariants from the synthetic data. Randomizing the image characteristics that are to vary at inference time is thereby a key condition for successful inferences in real-world conditions. We consequently augment our synthetic data statically at simulation time and dynamically during training.

Static Augmentation   During simulation, each dropped instance is augmented with a number of random geometric deformations: global transformations including isotropic and anisotropic scaling, tapering, bending, and twisting; local deformations of the mesh surface. As illustrated by Figure 5.5, such deformations enable to increase the local geometry variability, and consequently the rendered aspect. In addition with the variations of mesh templates, textures, and backgrounds, this results in statically augmented synthetic images.

Dynamic Augmentation   During training, we randomly filter one image out of two with a gaussian blur and jitter independently the RGB values, as shown in Figure 5.6, at both training and testing time. The parameters for gaussian filtering and value jittering are randomly chosen within empirically predefined intervals. This prevents the network from overfitting the too perfect synthetic color variations. In addition, the Mikado+ images are augmented with random permutation of the RGB channels and random under or over-exposition, as also illustrated by Figure 5.6. Unlike Mikado, Mikado+ thus depicts more color, texture and luminance variations as well.

	Mikado	**Mikado+**
Mesh templates	1	4
Backgrounds	40	600
Textures	120	2,400

Table 5.1: Differences between Mikado and Mikado+

D2SA [55]   **Mikado**   **Mikado+**

	Dataset	**D2SA**[1] **[55]**	**Mikado (Ours)**	**Mikado+ (Ours)**
Overall	Average resolution	1962×1569	640×512	640×512
	Number of images	5,600	2,400	**14,560**
	Number of instances	28,703	48,184	**459,002**
	Ground-truth annotations	Human-made	Computer-generated	
Per image	Number of instances	5.1	**20.1**	**31.5**
	Inter-instance occlusions	2.8	**52.9**	**60.5**
	Background pixels	79%	24%	24%

[1] The statistics on D2SA are only on the train and validation subsets as the test subset is not provided.

Figure 5.4: Samples and characteristics of the state-of-the-art real-world dataset for boundary and occlusion detection on piles [55] compared with our synthetic data. Unlike D2SA, the variety of occlusions is better represented in Mikado and the dataset can be extensively enriched at low cost (Mikado+).

Random geometric deformations in simulation?

No   **Yes**   No   **Yes**

Side view   Top view
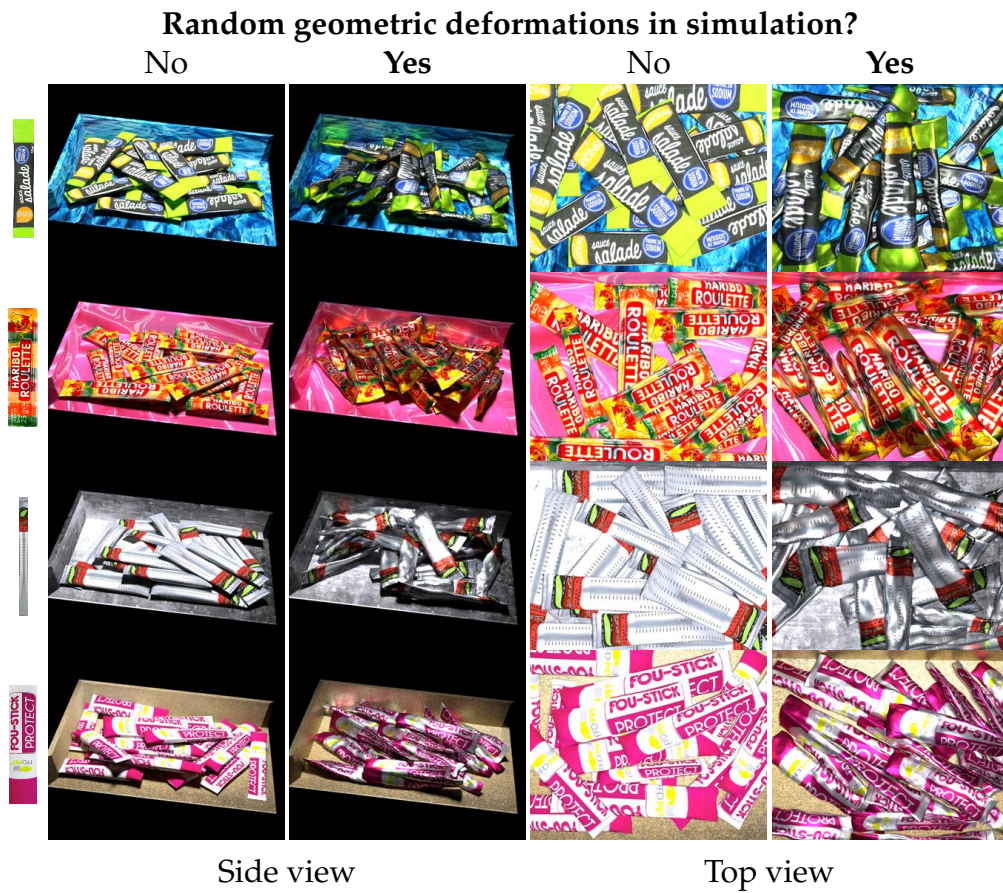
Figure 5.5: Augmenting the mesh template with random geometric deformations qualitatively induces much wider ranges of texture and light variations

Raw   Jittered   Blurred   Final

Raw   Recolored   Darkened   Final
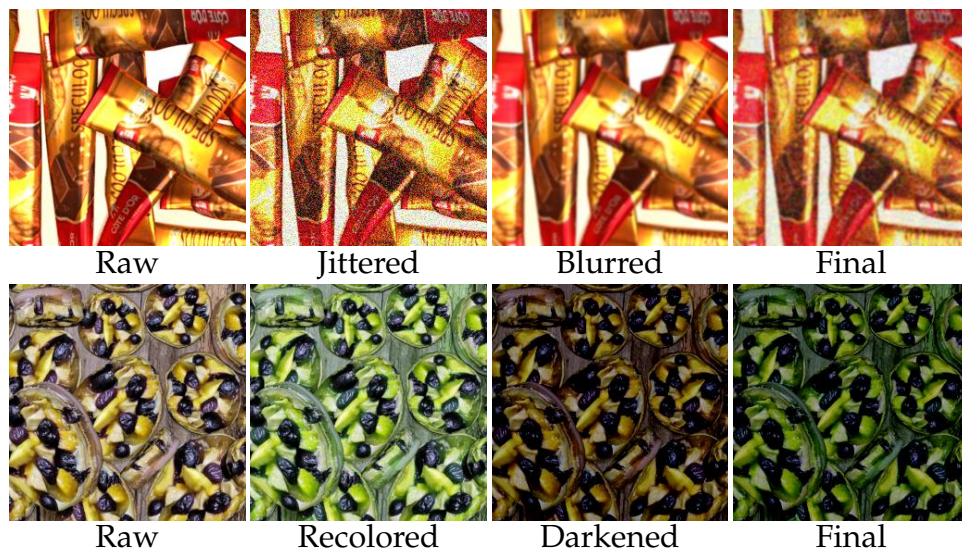
Figure 5.6: Synthetic data augmentation for Mikado and Mikado+ (first row), and Mikado+ only (second row)

5.2   Synthetic Data Plausibility Check

Unlike hand-made annotations, which are costly to obtain, simulation for generating training data is a virtually unlimited source of unbiased ground-truth annotations. However, one may raise the question whether such data is realistic. The answer is obviously no, but we argue that Mikado is plausible for training a bicameral network for real-world applications. In this section, we thus conduct experiments to jointly check the plausiblity and show the benefits of our synthetic data, independently of robotic interactions.
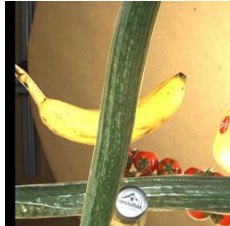
5.2.1   Experimental Setup

We show that the proposed synthetic data is plausible for real-world applications by evaluating the transferability of features learned from Mikado to real data. In line with [194], features learned from a source domain are transferable if they can be repurposed and boost generalization on a target domain. Specifically, we train the proposed network on Mikado, then retrain on the amodal dataset of [55] referred to as D2SA ($c.f$. Figure 5.4), only the decoders and some of the top encoder blocks, as deep features transition from general to specific by the last layers. Furthermore, as a proof of the benefits of synthetic data in contrast real-world datasets which are hardly extensible, we study how a richer synthetic data distribution, $i.e$. Mikado+, impacts the domain adaptation. As the ranges of texture, shape, and pose variations are more widely represented in Mikado+, better transferable invariants are expected to be learned. In addition, we compare with the augmentation strategy of [55], referred to as D2SA+, which consists in overlaying manually isolated instances of real images into fake training images. Specifically, we conduct three sets of experiments for comparing:

1. bicameral networks finetuned on D2SA without and after pretraining on Mikado, with different encoder block at which the network is choped and retrained (Figure 5.7), to expose the most transferable features learned from Mikado;

2. bicameral networks finetuned on D2SA using the most transferable synthetic features and different number of finetuning images, to reduce the need of hand-made annotations and compare with the augmentation strategy of [55], referred to as D2SA+;

3. bicameral networks finetuned on D2SA using the most transferable features learned from Mikado or Mikado+, to show the impact of a richer synthetic data distribution (Mikado+) on domain adaptation.

When finetuning on D2SA we define a block as a set of convolutional layers between two pooling layers; a VGG16-based encoder is therefore composed of 5 blocks (*c.f*. Fig. 5.7). A block is said "frozen" when its corresponding parameters remain unchanged during finetuning.

Why D2SA As Target Domain?   We consider D2SA instead of PIOD or COCOA (*c.f*. Table 3.16) for transfer learning from Mikado because the texture, shape, and pose distributions of PIOD and COCOA are very different from Mikado. Indeed, [14, 15] show that a low divergence between the source and target domain distributions is a necessary condition for the success of domain adaptation. Unlike PIOD and COCOA, which contain natural images of indoor and urban scenes with people, cars and animals, D2SA and Mikado both contain top-view images of household objects in bulk.

	D2SA [55]	**D2SA+*** [55]	**Mikado+ (Ours)**
Training images	512	2,960	28,800
Validation images	56	328	4,800
Test images	5,992	5,992	–
Training iterations	960	5,550	108,000
Training epochs	15	15	30
In experiments	1–3		2–3

* D2SA+ refers to the augmentation strategy of [55], consisting in creating fake images by overlaying isolated instances.

Table 5.2: Per-dataset folds for our cross-validation experiments after offline data augmentation

Data Preparation   To robustly assess the generalizability of each model, each experiment is cross-validated using three folds (see Table 5.2). Folds of D2SA are defined with respect to the initial split proposed by their authors. Specifically, the original training images are used for training or validation in our folds, and the original validation images for test. The original test images are never used as they are not publicly available. To present comparative results more significative, curves and scores are averaged on the three folds. For training, the networks are not directly fed with the original images but several sub-images

randomly extracted from each original image, and augmented offline with random geometric transformations (flipping, scaling and rotation). Note that performances are not impacted by cropping given that the bicameral network is fully convolutional.

Training Settings   For each dataset and each experiment, each network is trained and tested using Caffe [92], and the exact same settings (including fixed random seeds). At training time, we use the Adam solver [95] with $\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, a fixed learning rate of 10^{-4}, a weight decay of 10^{-4}, a ℓ_2 regularization, and a batch size of eight 256×256 images. The training images are randomly permuted at each epoch. As we solve a non-convex optimization problem, without theoretical convergence guarantees, the number of training iterations is chosen for each dataset from an empiric analysis on training and validation subsets. As generally adopted, the optimization is stopped when the validation error stagnates or increases while the training error keeps decreasing. Please note that although the chosen stopping criterion may not be optimal for reaching the best performances on each dataset, it is however sufficient for significative comparative performances on a given dataset since each network in a comparison is trained under the exact same conditions. For the experiments without finetuning from weights pretrained on Mikado or Mikado+, each network has its encoder initialized with weights pretrained on ImageNet [159], and its decoder(s) with the Xavier method [61]. The decoders are also equipped with dropout layers (with a ratio of 0.5) after each convolutional block at training time, to avoid overfitting.

Evaluation Metrics   We use the same evaluation metrics as introduced in Section 4.1.3: the best F-score on dataset scale (ODS) and the average precision (AP). Whereas ODS highlights one binarization threshold that gives the best compromise between recall and precision, AP conveys the area under the precision-recall curve over the full recall interval. As matching tolerance, *i.e.* the maximum ℓ_2-distance to the closest ground-truth pixel for a pixel predicted positive to be considered as a good hit, we consider a hard value of 0 pixels for Mikado (which contains perfect ground-truth boundaries) but a state-of-the-art value of $0.0075\sqrt{W^2 + H^2} (\simeq 2.7$ pixels in our evaluations) for D2SA that contains approximate hand-made annotations, where $W \in \mathbb{N}^\star$ and $H \in \mathbb{N}^\star$ are respectively the image width and height. We perform evaluation without non-maximum suppression, which may artificially improve precision.
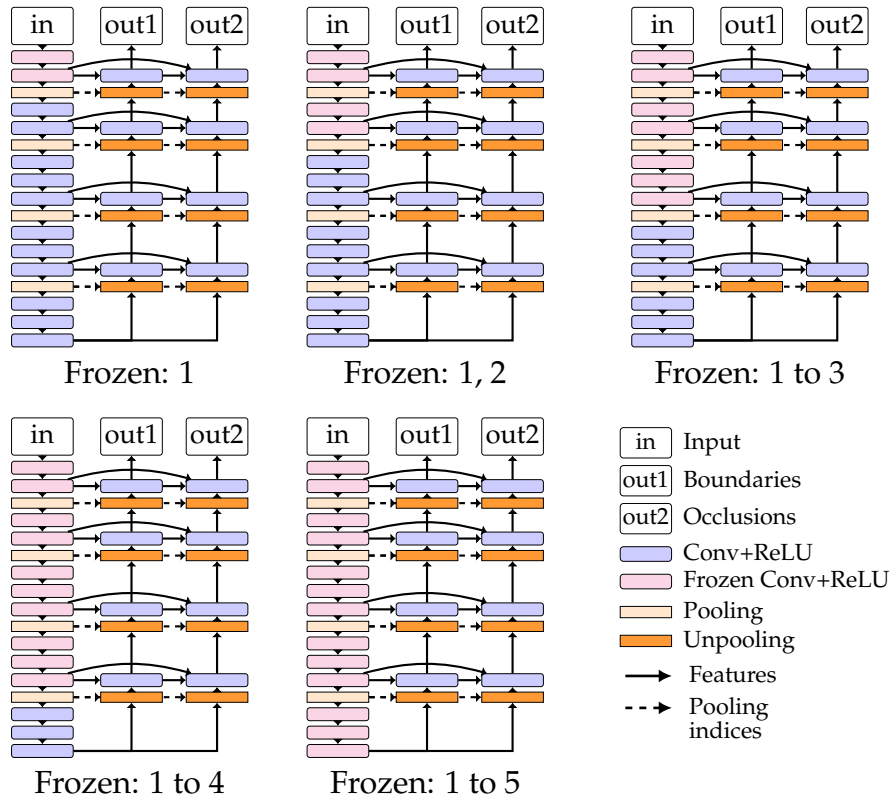
Figure 5.7: A bicameral structure with frozen encoder blocks

Pretraining images	Finetuning images	Frozen encoder blocks* (Fig. 5.7)	Boundaries		Occlusions	
			ODS	AP	ODS	AP
None	D2SA	None	.700	.715	.725	.756
	D2SA+		.783	.792	.785	.795
Mikado		None	–	.652	.649	.458	.400
		None	.780	.808	.794	.830
		1	.783	.803	.797	.829
	D2SA	1, 2	.780	.802	.793	.827
		1, 2, 3	**.793**	**.819**	**.810**	**.849**
		1, 2, 3, 4	.759	.799	.769	.819
		1, 2, 3, 4, 5	.767	.815	.773	.823

* A block is a set of convolutional layers between two pooling layers; a VGG16-based encoder is therefore composed of 5 blocks.

Table 5.3: Comparative performances of the proposed network on D2SA [55] using different pretraining conditions. Performances on both boundaries and occlusions are maximized when freezing at finetuning time the first three encoder blocks pretrained on Mikado (see Figure 5.7).
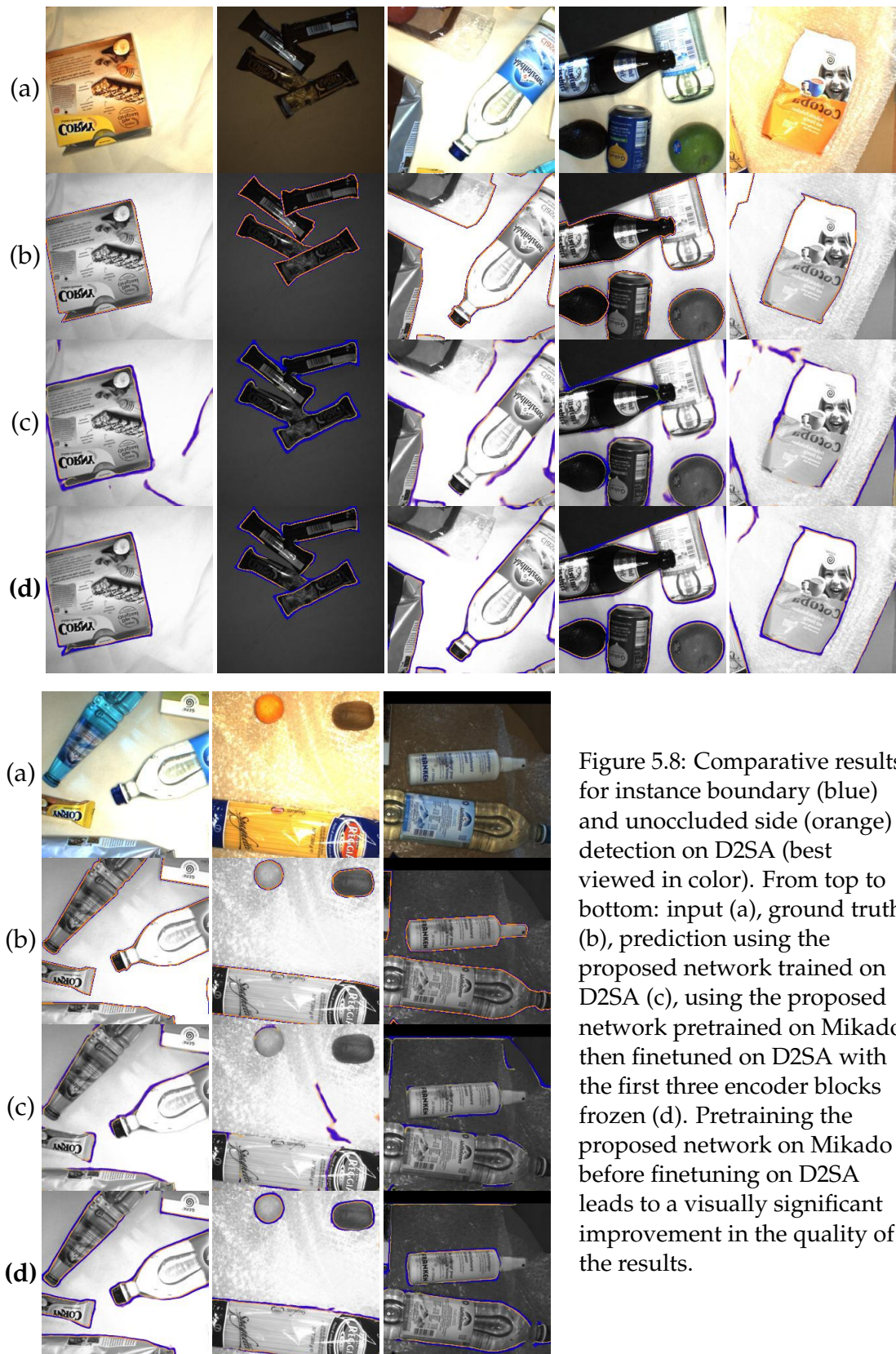
Figure 5.8: Comparative results for instance boundary (blue) and unoccluded side (orange) detection on D2SA (best viewed in color). From top to bottom: input (a), ground truth (b), prediction using the proposed network trained on D2SA (c), using the proposed network pretrained on Mikado then finetuned on D2SA with the first three encoder blocks frozen (d). Pretraining the proposed network on Mikado before finetuning on D2SA leads to a visually significant improvement in the quality of the results.

5.2.2   Transfer Learning Experiments

In line with [194], we first expose the most transferable features learned by a bicameral network from Mikado, and the impact of their reuse on the performances on D2SA (Table 5.3). We then study the performances obtained with the most transferable features, with respect to the percentage of the initial number of finetuning images and the synthetic training data distribution (Mikado or Mikado+), compared with the D2SA and D2SA+ baselines (Figure 5.9).

Synthetic Data Instead of Hand-Made Annotations   As Mikado is a computer-generated dataset, one may raise the question whether it is realistic. The answer is obviously no, but we argue that it is plausible for both a significative evaluation and real-world applications. First, in Chapter 4 when comparing network designs, the same overall relative results are obtained on PIOD, a dataset of manually annotated natural images. Second, the synthetic features learned from Mikado can be repurposed for inference on real images. Specifically, Mikado enables a transferable feature learning in line with [194], *i.e.* first training the network on a source dataset, then retraining only the task-specific layers on the target one. In our transfer learning experiments, we show that using local features pretrained on Mikado enables much better results on D2SA, a dataset of real-world piles of supermarket items [55]. As reported by Table 5.3 and qualitatively corroborated by Figure 5.8, a gain of more than 10 points in AP for boundaries and 9 points for occlusions is achieved when finetuning the proposed network on D2SA with the first three encoder blocks frozen (*c.f.* Figure 5.7) after pretraining on Mikado, instead of training all the layers only on D2SA. This suggests that a network trained on Mikado can learn a more general concept of depth ordering as our dataset presents a wider variety of occlusion relations, including both inter-instance and object/background boundaries. [55] also introduces an augmentation procedure to enrich the training subset with more piles of objects (D2SA+). Their procedure consists in creating new images by overlaying manually isolated instances. Table 5.3 reports that our simulation-based pretraining outperforms D2SA+ as well. Despite the domain shift, simulation enables more physics-consistent rendering at boundaries and less redundancy in terms of poses, unlike brute-force overlaying of manually delineated instance segments from real images. Furthermore, almost equivalent performances on D2SA are achieved, while reducing the number of costly human-labeled real images for finetuning. Figure 5.9 shows that a bicameral network finetuned on D2SA, with the first three encoder blocks frozen after pretraining on Mikado, using only 25% of the initial D2SA finetuning subset still outperforms a bicameral network trained

only on D2SA or D2SA+.
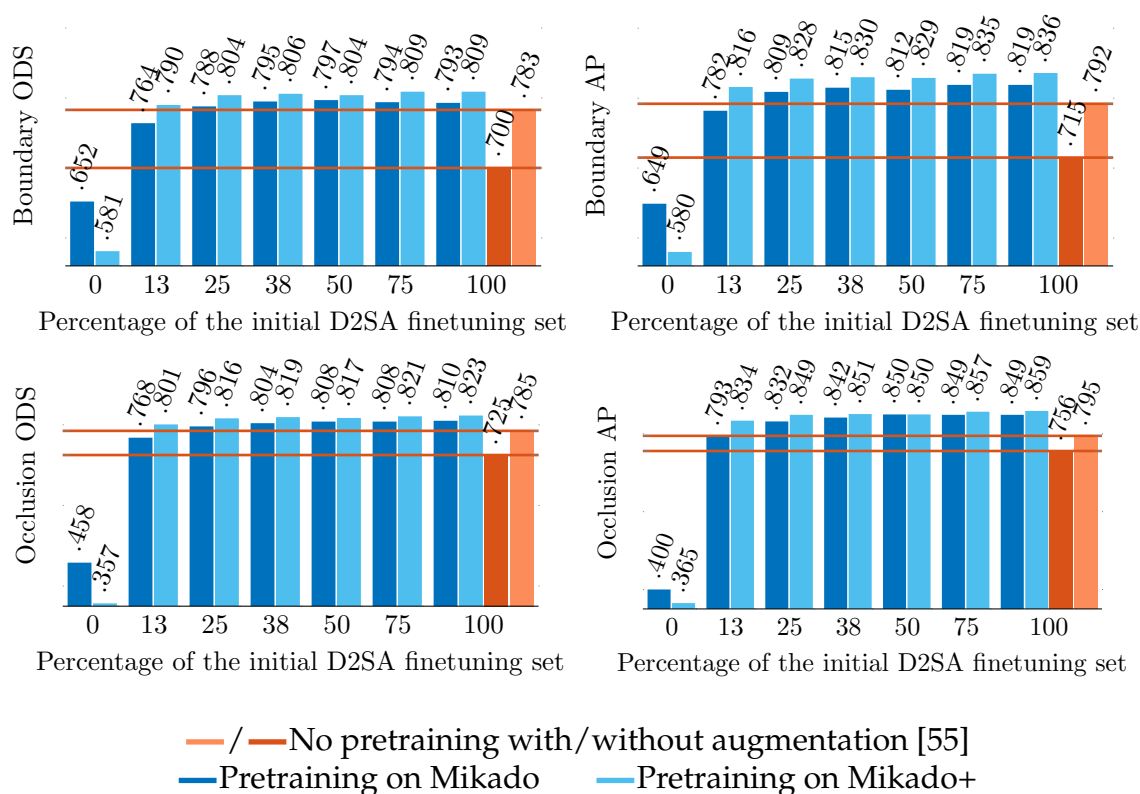
Figure 5.9: Performances of the proposed bicameral network pretrained on Mikado/Mikado+ then finetuned on D2SA with the encoder blocks 1, 2, 3 frozen, with respect to the number of real images retained for finetuning. Exploring a wider range of configurations in simulation (Mikado+) enables to learn more abstract local representations of the boundaries and occlusions, thus achieving state-of-the-art performances while drastically reducing the number of real images for finetuning. Best viewed in color

Synthetic Data for Learning More Generalizable Invariants   Unlike real-world datasets, a synthetic dataset is readily extensible. By enriching Mikado with 20 times more texture images, 15 times more background images and 4 mesh templates, namely Mikado+, we expect more transferable local and global invariants to be learned as the ranges of color, texture, shape, and pose variations are better represented. Table 5.3 indeed reports that pretraining on Mikado+ instead of training only on D2SA increases AP by 10.1 points for boundaries and 7.8 points for occlusions while using only 13% of the initial D2SA finetuning set (Figure 5.9). This corresponds to a gain of 3.4 points for boundaries and 4.1 points for occlusions over using Mikado in the same conditions.

These observations imply that Mikado+ enables to learn more abstract local representations than Mikado. However, when applied on D2SA without finetuning, the Mikado+ model proves less effective than the model pretrained on Mikado. Consistently with the results after finetuning on D2SA, this could be explained by an overgeneralization of the task-specific layers. The neurons indeed co-adapt to capture the most discriminative patterns that are not likely to be the colors nor the object and background textures in Mikado+. An over-randomization of the colors and textures may disconnect the learned representations from concrete examples. This has nevertheless the advantage of easing the finetuning on D2SA, as the real-world scenes then appear as a specific variation consistent with the learned abstract representations.

All these observations are incentives to favour synthetic training data when pixel-wise annotations on real-world images are hardly collectable. As also previously seen in Section 4.2 and illustrated in Figure 4.8, hand-made annotations introduce biases in the training and evaluation due to inaccuracy, incompleteness, or inconsistency. These biases are strongly attenued by using instead synthetic data. Indeed, leveraging unbiased ground-truth annotations and randomizing the scene parameters that are to vary at inference time enable a better representation of the training data distributions.

5.2.3 Conclusion

The proposed synthetic data, referred to as Mikado, proves plausible for real-world applications in the sense that it enables the learning of deep features transferable to real data, while drastically reducing the need of hand-made annotations for finetuning. Specifically, reusing the first three encoder blocks in a VGG16-based bicameral encoder trained on Mikado increases AP by more than 10 points for both boundaries and occlusions over the baseline, and near 5 points over D2SA+, *i.e.* the augmentation strategy of [55] based on manual instance delineation. State-of-the-art performances over D2SA and D2SA+ are still achieved using these representations learned from Mikado with only 25% of the initial number of finetuning D2SA images, and only 13% by enriching the synthetic training data distribution (Mikado+). **Randomizing the shape, texture and light variations indeed enables the bicameral network to capture more generalizable multiscale invariants.**

Note that the proposed Mikado pipeline is virtually not limited to RGB. Depending on the application requirements, Mikado could be easily adapted for an alternative input modality such as depth (see Section C.1.2).

5.3   Real-World Experimental Evaluation

We showed that the proposed synthetic data is plausible for real-world applications. In this section, we now conduct experiments on a real-world robotic setup, in order to answer the following questions:

- Does a synthetically trained bicameral network enable effective bin-picking performances in real-world conditions?

- How does the proposed solution compare with the gripper-oriented industrial baseline?

- Can a bicameral network achieve real-time performances for high-throughput bin-picking applications?

5.3.1   Real-World Experimental Setup

As illustrated by Figure 5.10, our real-world robotic setup consists of a single RGB camera to capture a top view of the scene, a FANUC robotic arm equipped with a suction cup as end effector, and a Nvidia Jetson TX2 as processing unit, all connected to a wired Ethernet network.
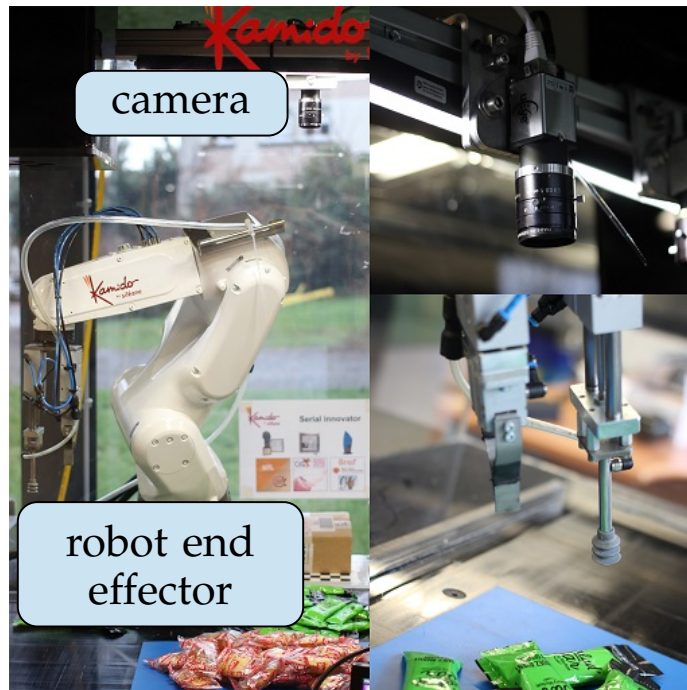
Figure 5.10: Overview of the real-world robotic setup for our experiments. A single RGB camera (top right) captures a top-view of the scene. A robotic arm (left) equipped with a vacuum suction gripper (bottom right) extracts instances one by one.

The intrinsic and extrinsic parameters of the camera are estimated using standard off-the-shelf calibration tools [25, 203] based on the pinhone cameral model [71]. As a single camera is not sufficient to estimate the depth of a pixel using the state-of-the-art pinhole camera model, we

determine the depth by intersecting the corresponding 3D ray with a
user-defined plane, in practice a plane close and parallel to the bottom
of the scene. This proves to be a reasonable approximation in practice,
given the relatively low height of the pile and the robot compliance.

5.3.2   Experimental Protocol

In order to evalutate the proposed approach in real-world conditions,
we conduct a set of experiments that consist in **unpiling packaged food
products using a bicameral network trained on Mikado or Mikado+
without any finetuning on real images**. Specifically, each experiment
is a sequence of open-loop cycles, aimed at extracting one by one some
product instances piled up in bulk. Consistently with mainstream
scenarios in the food industry, each scene is always composed of many
instances of the same object.

	Criterion	Description
(A)	Size	Area of an instance
(B)	Isotropy	Aspect ratio of an instance
(C)	Shape	Curvature of the instance boundaries
(D)	Textureness	Presence of complex patterns such as text
(E)	Colorfulness	Number and variations of colors
(F)	Transparency	Presence of transparent or translucent parts
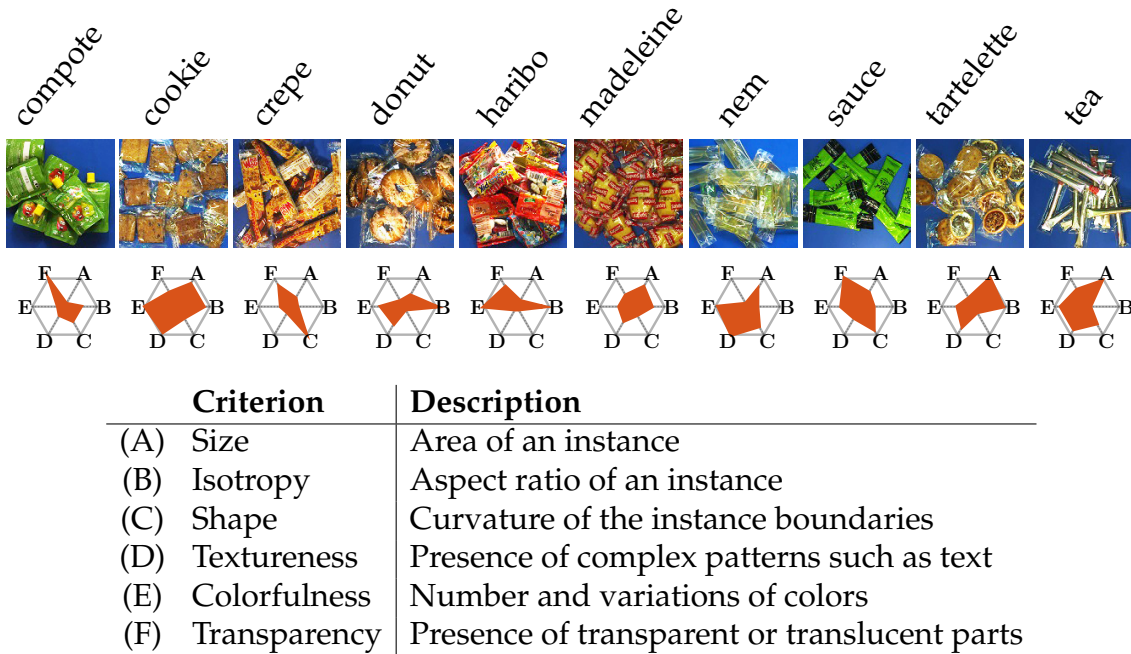
Figure 5.11: Overview of the 10 products used in our real-world experiments.
The spider charts show their signature according to six qualitative criteria:
size (A); isotropy (B); shape (C); textureness (D); colorfulness (E); transparency
(F). Each product represents a different category of packaged food, as the
signature shapes are all different. Best viewed in color

Each cycle comprises the following steps:

1. capturing a top-view RGB image of the scene;

2. detecting the most affordable instance;

Question	Three possible answers
Q1: Is the extraction successful?	Yes; No; More than one instances
Q2: Is the grasp centered on the instance?	Yes; No; Between two instances
Q3: Is the detected instance occluded?	Not at all; Weakly; Highly

(a) Definition of an observation

t=1	t=2	t=3	t=4
Q1: Yes	**Q1:** Yes	**Q1:** Yes	**Q1:** Yes
Q2: Yes	**Q2:** No	**Q2:** Yes	**Q2:** Yes
Q3: No	**Q3:** Weakly	**Q3:** No	**Q3:** No

t=1	t=2	t=3	t=4
Q1: Yes	**Q1:** Yes	**Q1:** No	**Q1:** Yes
Q2: Yes	**Q2:** Yes	**Q2:** Between two	**Q2:** Yes
Q3: No	**Q3:** No	**Q3:** –	**Q3:** No
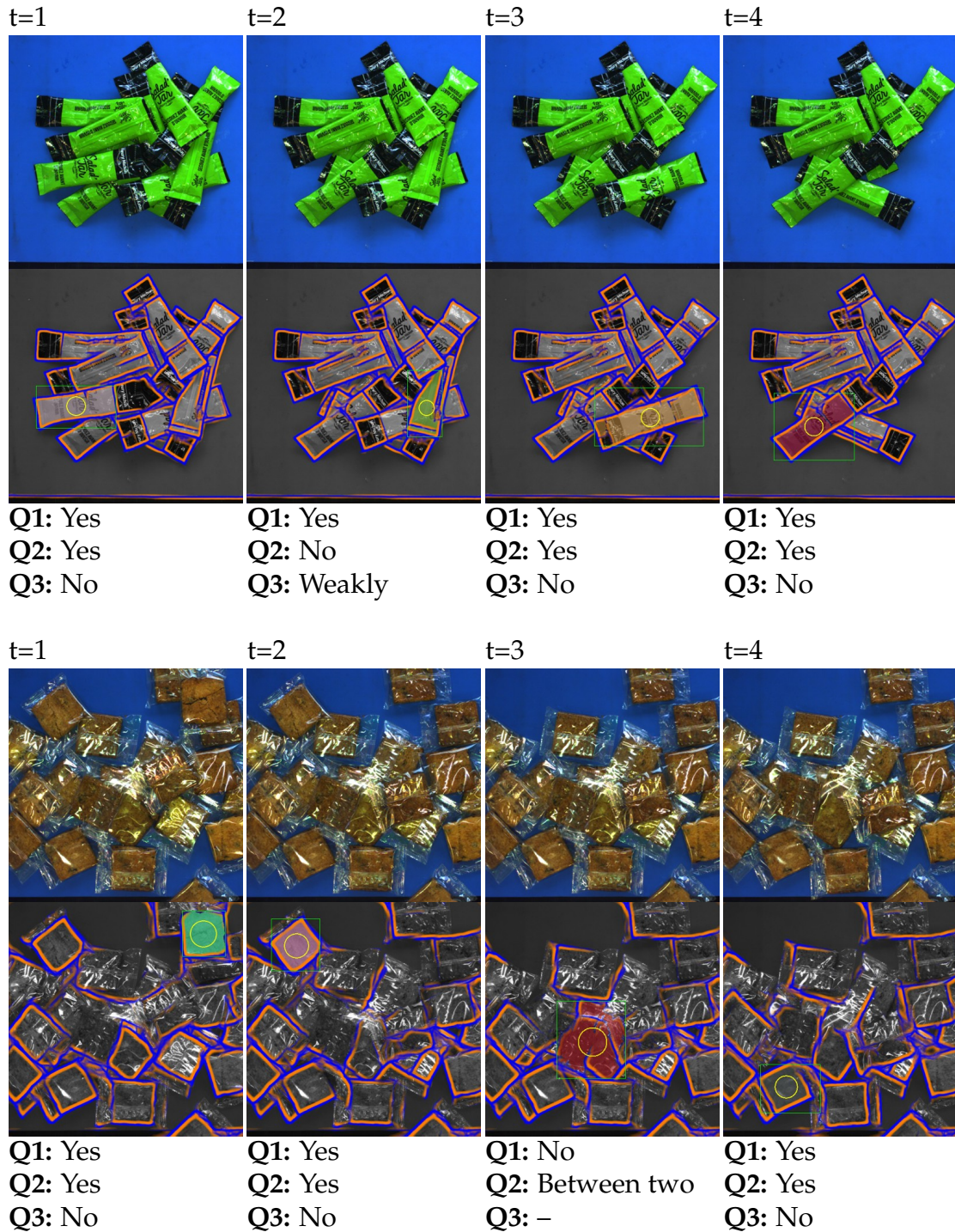
(b) Extracts from bin-picking sequences with the corresponding observations

Figure 5.12: Observations in our real-world experiments. Best viewed in color

3. extracting the instance at the sent coordinates;

4. observing the result.

As depicted in Figure 5.12, each observation consists in answering three questions to characterize the instance detection and extraction. In total, we collected 3,082 observations, organized in 134 sequences, over 10 different products. An overview of the different products instantiated in a pile, is provided in Figure 5.11. Each product presents a different combination of geometric and photometric characteristics, thereby representing a different category of packaged food.

Evaluation Metrics   We calculate the evaluation metrics in Table C.2 over all the observations and per product. Specifically, we are interested in the observed success rate (S_R) and the success of the proposed method independently of the extraction (S_V). The margin between S_V and S_R thus reports the failures due to imperfect physical settings, typically by lack of grasp adhesiveness. We additionnally analyze the causes of success and failure by evaluating how object-centered grasps and occlusion-aware detections contribute to the performances.

Metric	Measures the amount of
Real success	Successful extractions, *i.e.* whose instance is taken away
Virtual success	Extractions either successful, or failed but whose grasp is centered on a non-occluded instance
Centered success	Successful extractions whose grasp is centered
Non-occluded success	Successful extractions whose instance is unoccluded
Not-centered failure	Failed extractions because of not centered grasps
Occluded failure	Failed extractions because of occluded instances

Table 5.4: Definition of our real-world performance metrics

Our experimental analysis is organized as follows. First, we study how a synthetically trained bicameral network behaves in real-world bin-picking conditions. Second, we compare the proposed approach with our industrial baseline. Finally, we focus on computation times and how to achieve real-time performances as requested by high-throughput applications.

5.3.3   Generalization from Synthetic Training

In this section, we study how a bicameral network trained on the proposed synthetic data performs on real-world piles of instances, how the synthetic training data distribution impacts these performances, and how sensitive is the network to light changes.

Overall Performances   As reported by Figure 5.13 (see Table C.3 for more details), the proposed method achieves an overall success rate of 74%. As 27% of failures are actually good detections, *i.e.* grasps centered on unoccluded instances, but due to lack of gripper adhesiveness, the success virtually reaches 81% over all products. In our experiments, we found four reasons of non-adhesiveness:

- The user-defined bottom plane is set too high. As a consequence, the end effector doesn't reach the instance.

- The grasp is on a non-planar surface. Vacuum thereby cannot be created at the grasp coordinates.

- The target instance is in unstable equilibrium. As a result, when the gripper approaches, the instance is moved away from its initially detected location (see Figure 5.15).

| Input | Result | (a) | (b) | (c) |
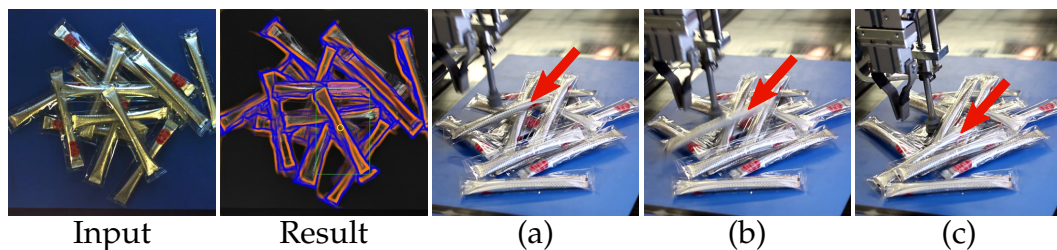
Figure 5.15: Example of good detection but failed extraction because the instance is in unstable equilibrium. When the end effector approaches (a), the instance is moved away (b), thereby resulting in a failed extraction (c).

- The instance's center of gravity differs from the instance mask centroid. This happens notably for half-empty sachets whose content is not homogeneously distributed in the sachet, such as haribo for which 85% of detections are visually correct but only 52% of extractions are successful.

Consistently with our objective, most of the successful extractions result from detected grasps centered on unoccluded instances: over all products, 91% of successful extractions are consequent to instance-centered grasps and 85% to grasps on non-occluded instances. For most
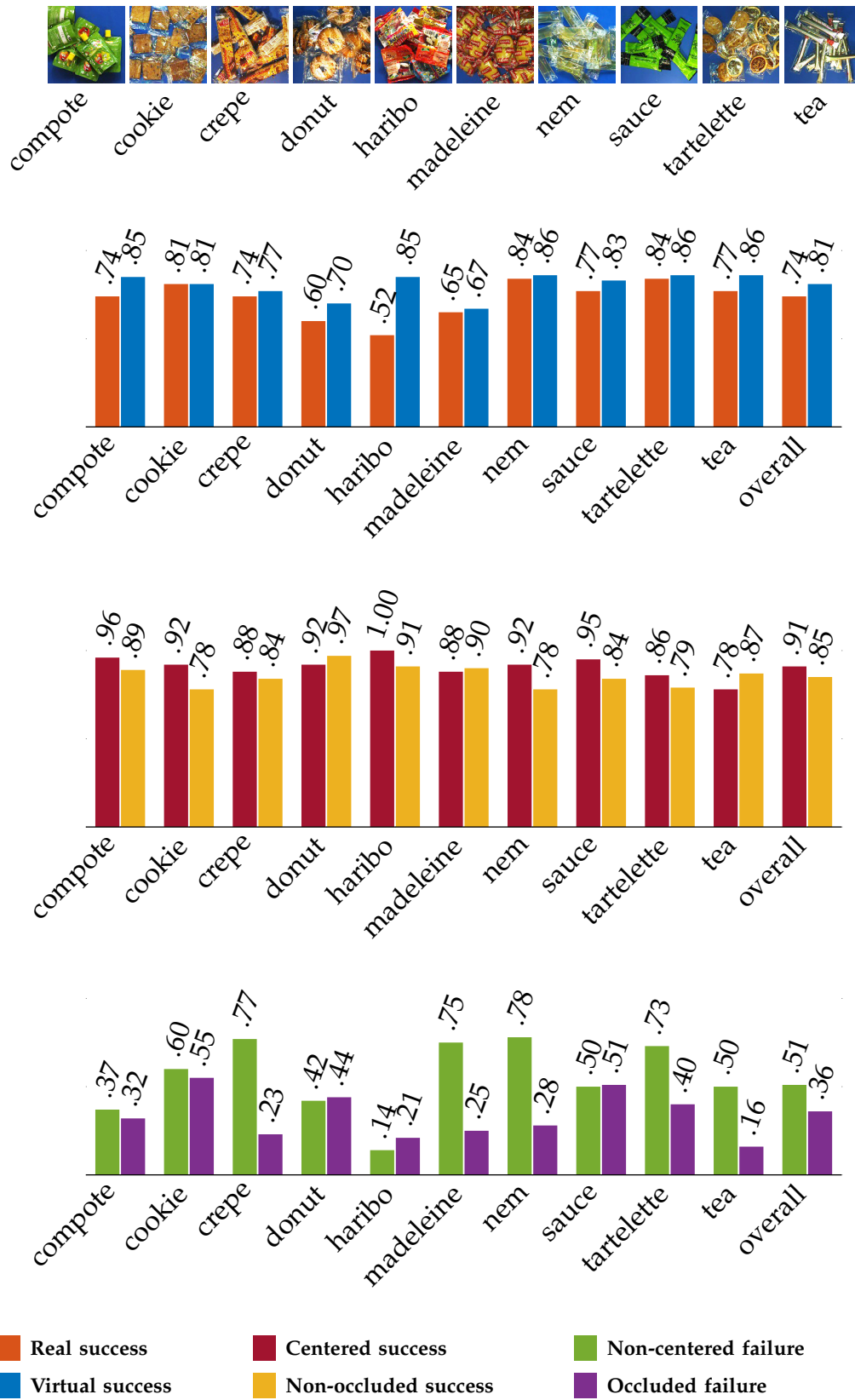
Figure 5.13: Per-product and overall performances using a bicameral network trained on Mikado+. Most of the successful extractions result from detected grasps centered on unoccluded instances, consistently with our approach.

Examples of success for each product
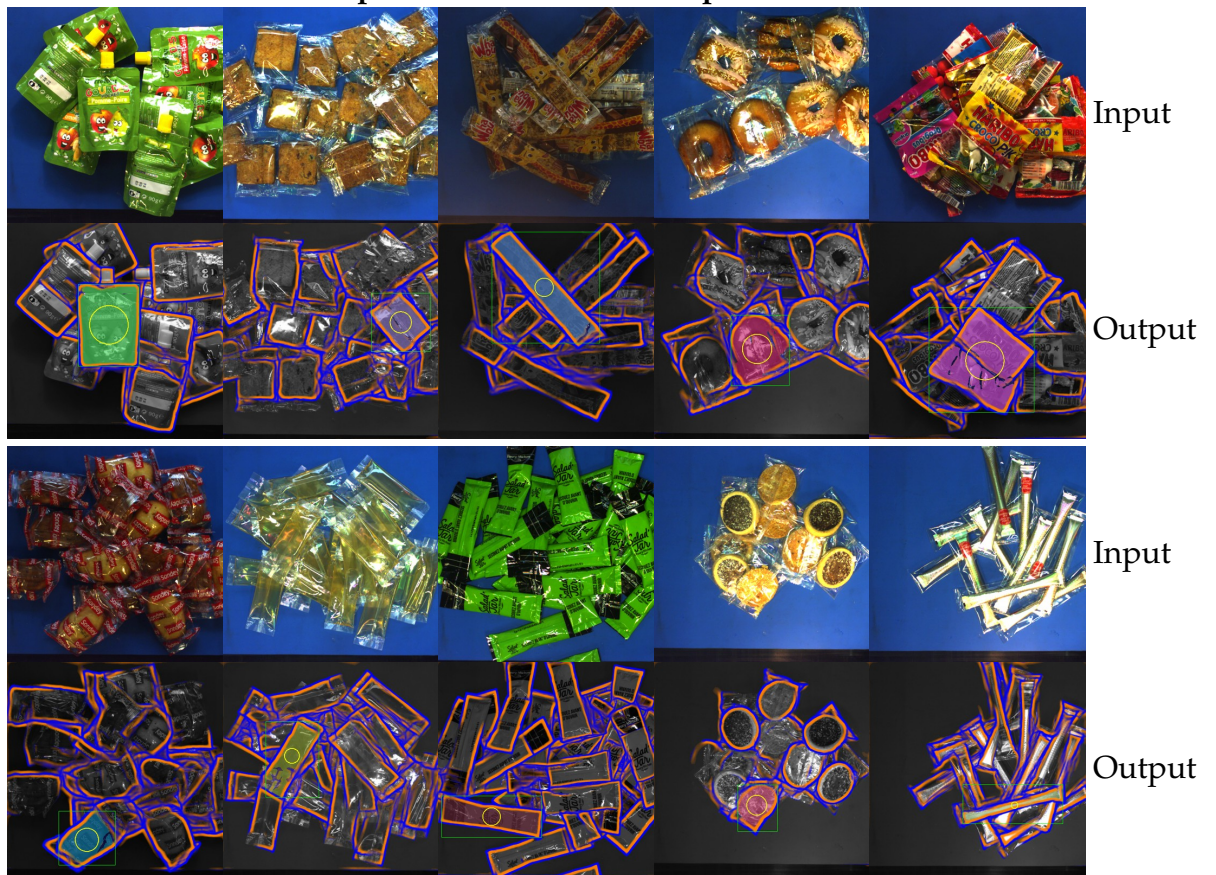
Examples of failure due to oversegmentation or overclustering
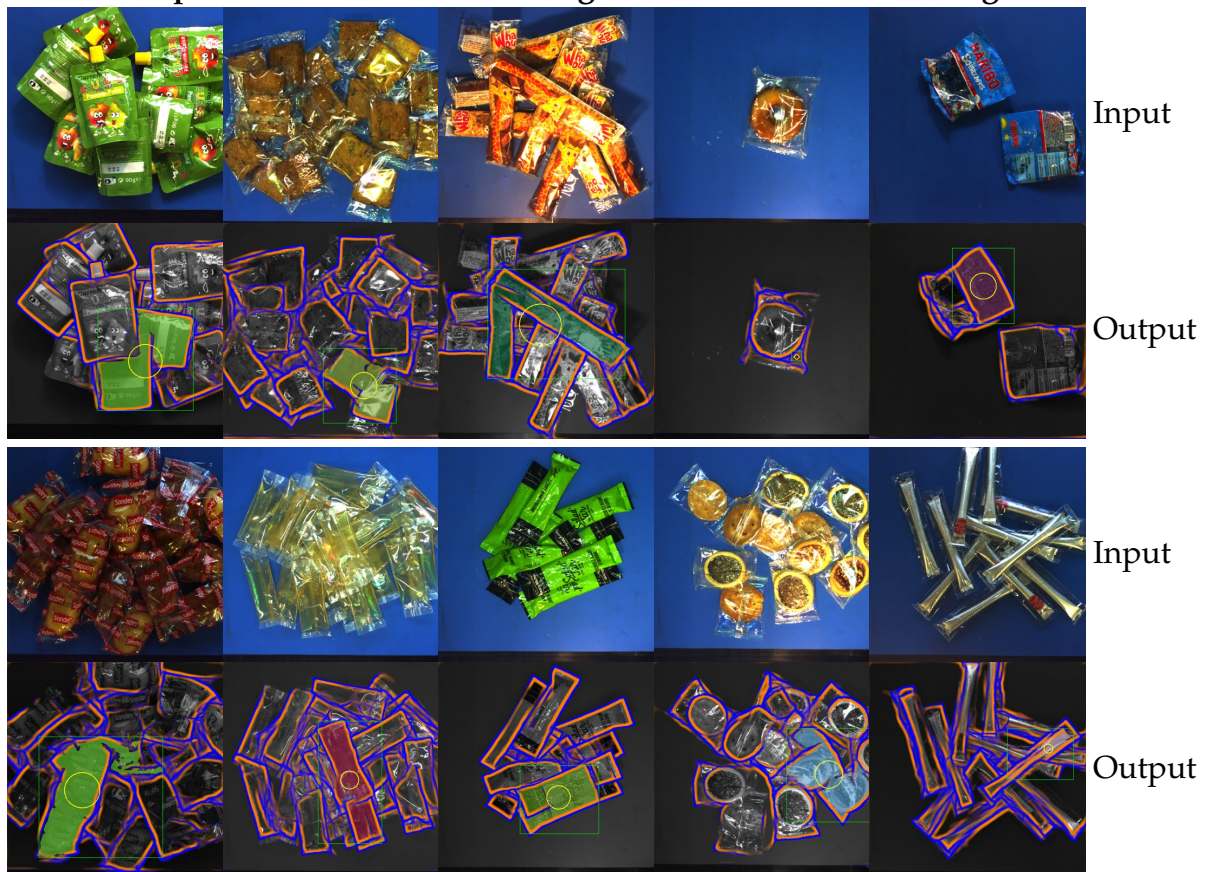
Figure 5.14: Results using a bicameral network trained on Mikado+. See also video "binpicking01" in supplementary material.

of the products, at least 50% of the failures are non-centered grasps. In accordance with our motivations, this illustrates that grasps not centered on the instances are a major concern in bin-picking applications. The difficulties arising from occlusions are more product-dependent. For example, instances of tea are prone to strong occlusions but as each instance is rigid and very light, occluded instances can be easily extracted (16% of the failures); the pile is consequently completely reconfigured. By contrast, instances of sauce, whose geometry is nevertheless similarly anisotropic, are non-rigid and relatively heavier, thereby harder to extract if occluded (50% of the failures).

Impact of the Synthetic Data Distribution   In Figure 5.16 (see Table C.4 for more details), we compare on stick-like products two bicameral networks trained on Mikado and Mikado+ respectively. Mikado contains only stick-like sachets and a hundred of textures while Mikado+ includes different shapes and thousands of textures (*c.f*. Figures C.1 and C.2). As expected from Section 5.2, in which we showed that learning from Mikado+ instead of Mikado produces more abstract representations, a bicameral network trained on Mikado+ outperforms the same network trained on Mikado, by 7, 9, and 32 points on sauce, tea and nem respectively. The obtained bin-picking performances are qualitatively corroborated by Figure 5.17: a model trained on Mikado+ instead of Mikado tends to produce more closed instance boundaries and less false positive, thus enhancing the generation of an affordable candidate.

Impact of the Lighting Conditions   Beyond learning texture and geometry invariants, one may wonder whether a synthetically trained bicameral network can be robust to light changes as well. In industrial environments, the robotic setup space is often confined to avoid external light pertubations. However, this may result in deploying costly cumbersome structures and power-consuming lights. A solution agnostic to light changes would thereby enable machines more independent of their environment. Figure 5.18 reports that the real-world performances are implicitly linked to the lighting conditions. As it is hard to anticipate and synthesize the real-world lighting conditions, randomizing the image luminance during training (Mikado+) enables performances more independent of the light changes. A model trained on Mikado reaches an average success rate of 63% with extrema of 80% and 20%, against an average success of 79% with tighter extrema of 60% and more than 90% for a model trained on Mikado+.
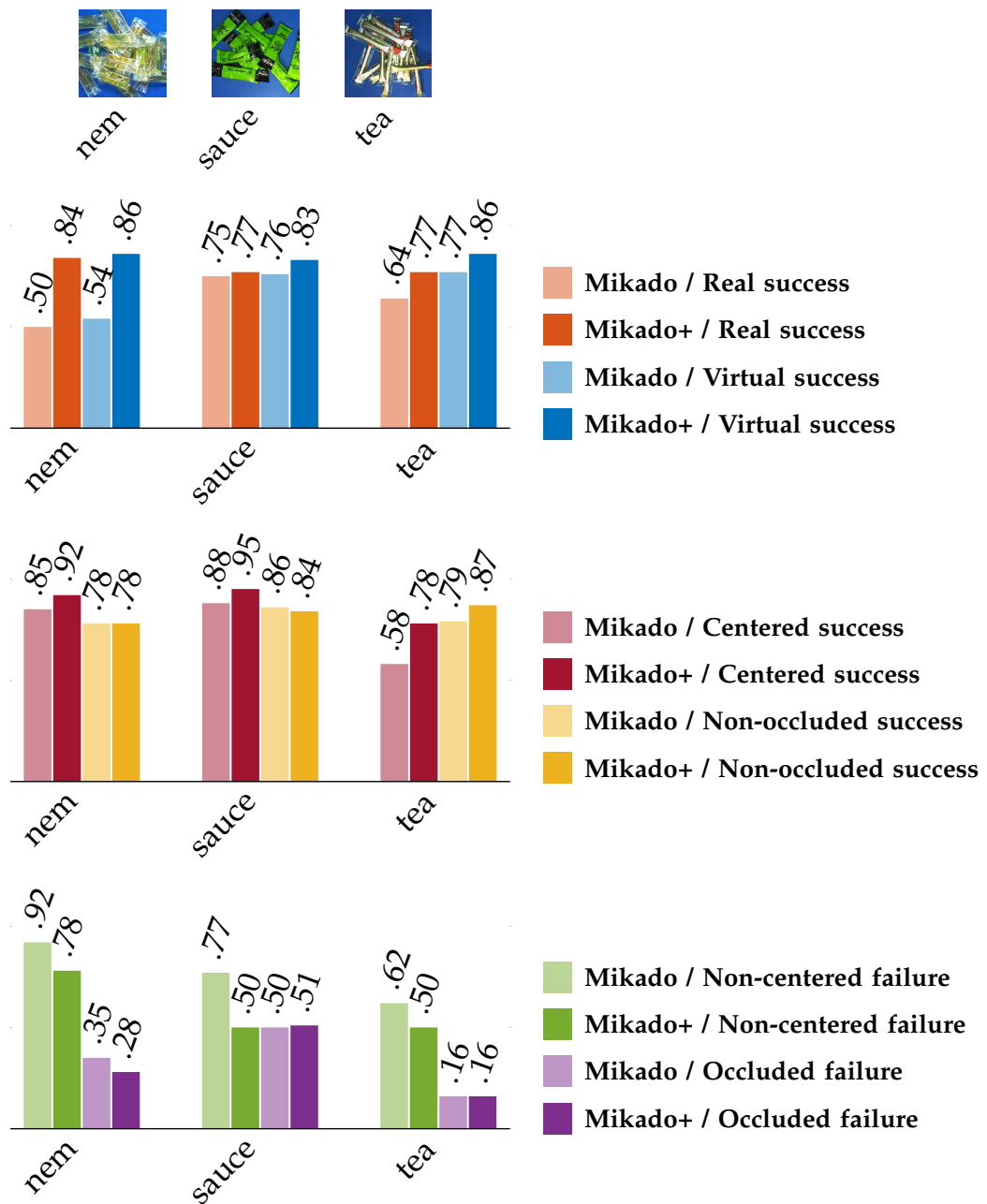
Figure 5.16: Per-product performances using the proposed approach, with respect to the synthetic training distribution, *i.e.* Mikado or Mikado+. Training on Mikado+ instead of Mikado enables to boost performances, as the bicameral network can learn more generalizable invariants.
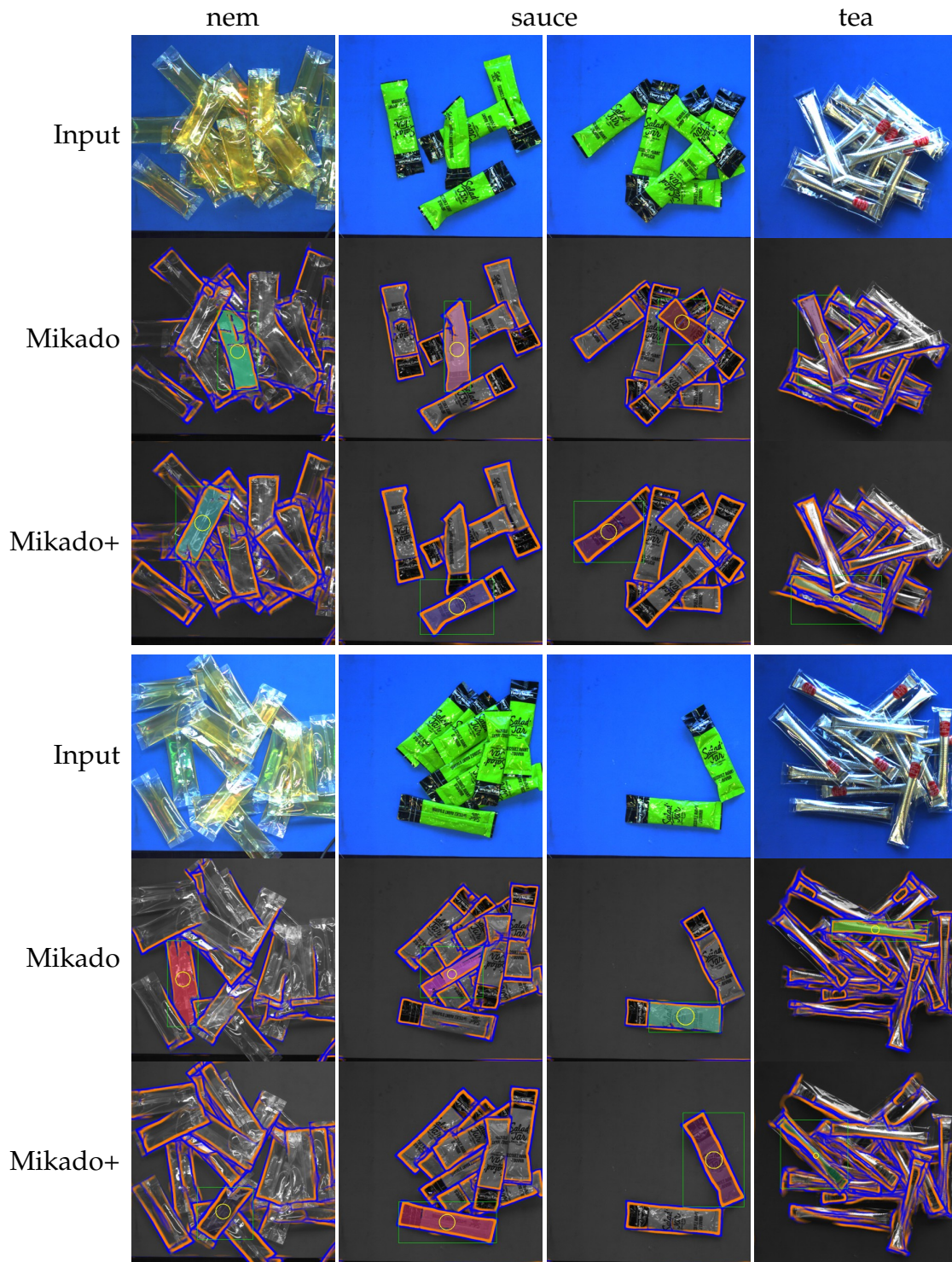
Figure 5.17: Comparative results using bicameral networks trained on Mikado and Mikado+ respectively (best viewed in electronic form). Randomizing the textures, shapes, and light conditions (Mikado+) enables more closed boundaries and less false positives, thus enhancing the real-world bin-picking performances.
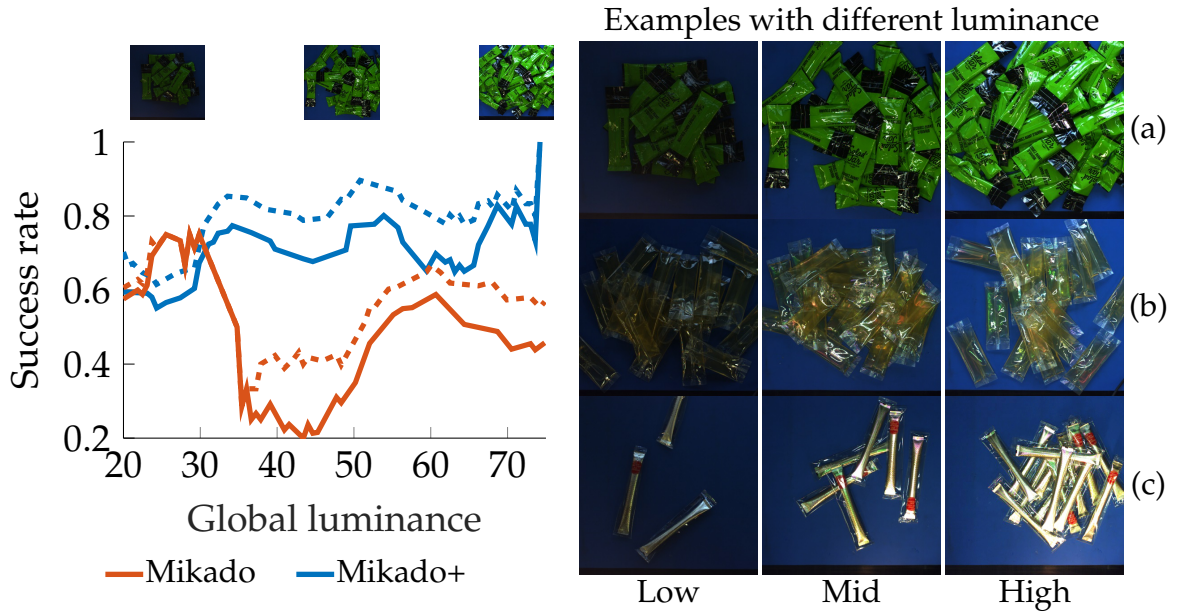
Figure 5.18: Real (solid lines) and virtual (dashed lines) success rates S_R and S_V, averaged over three products, namely sauce (a), nem (b) and the (c), with respect to the global image luminance. Randomizing the luminance of the synthetic training images (Mikado+) enables more robustness to real-world light changes.

Conclusion A synthetically trained bicameral network trained on Mikado+ achieves an average bin-picking success rate of 80% over the 10 products. Randomizing the textures, the shapes and the lighting conditions, *i.e.* using Mikado+ instead of Mikado, enables better and more stable performances, consistently with our synthetic data plausibility check in Section 5.2.

5.3.4 Comparison with the Industrial Baseline

In this section, we compare on two products, sauce and madeleine, the proposed approach with the industrial baseline, which consists in detecting grasps independently of the object models, just like ours, but without an explicit notion of instance. As depicted in Figure 5.19, the baseline is a gripper-dependent proprietary algorithm that takes a depth image as input and returns ranked grasp opportunities based on the detection of planar regions in the case of a vacuum-suction gripper.

As reported by Figure 5.20 (see Table C.5 for more details), the proposed model outperforms the industrial baseline by 28 and 34 points on sauce and madeleine respectively. Unlike the gripper-oriented baseline, our approach enables instance-centered grasps and avoids occluded instances, hence a much higher success rate. For example, when using the

| Overview | Input (depth map) | Output |
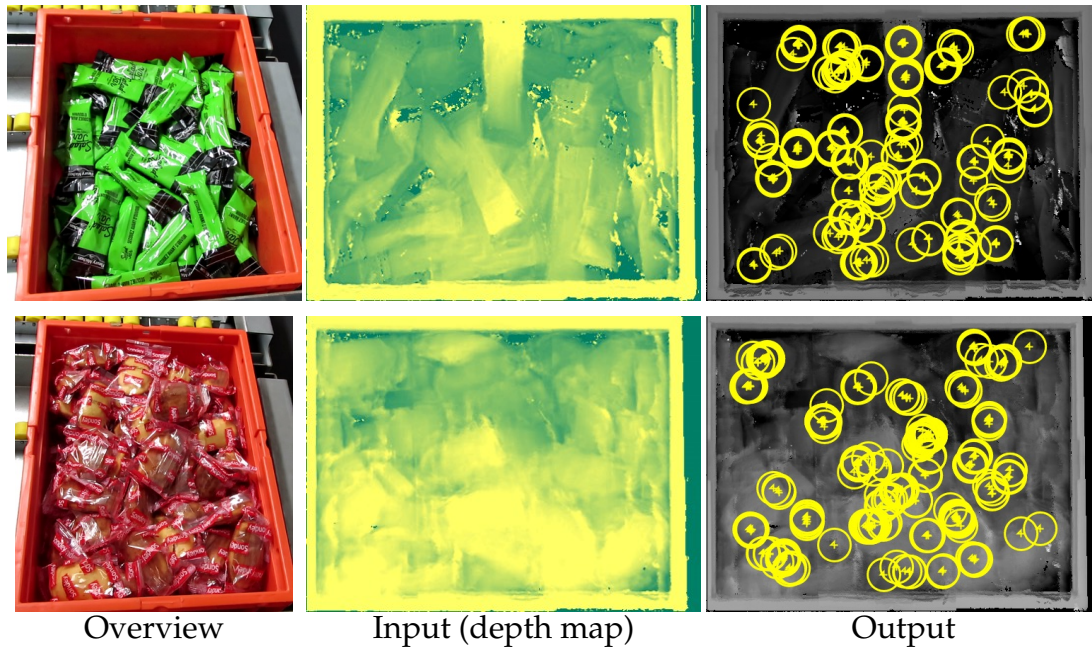
Figure 5.19: Examples of grasp detection using the industrial baseline, which consists in detecting planar regions from the depth map, independently of the object model but without an explicit notion of instance. The depth map is here obtained by laser triangulation.

baseline on instances of sauce, whose non-rigid stick-like shape makes object-centered and occlusion-aware grasps important conditions of success, 87% of the failed extractions are not centered and 35% on occluded instances, *i.e.* respectively 44% (.87 × .51) and 18% (.35 × .51) of the robot operating cycles. By contrast, only 12% of the robot cycles are impacted by these causes of failure using our object-oriented approach.

5.3.5   Achieving Real-Time Performances

We showed that the proposed method enables state-of-the-art performances over various products. In these experiments, the proposed method takes 1,800ms on average to process an image on the Nvidia Jetson TX2. However, in high-throughput bin-picking applications, the demanded cadency often reaches about 60 robot picks per minute, thereby hardly leaving 100ms for computing relevant grasp coordinates. In this section, we thus investigate faster variants of the bicameral network. Specifically, we first study how reducing the number of convolutional filters in the bicameral network impacts the performances. We then analyze the resulting computation times and bottlenecks with respect to different hardwares.
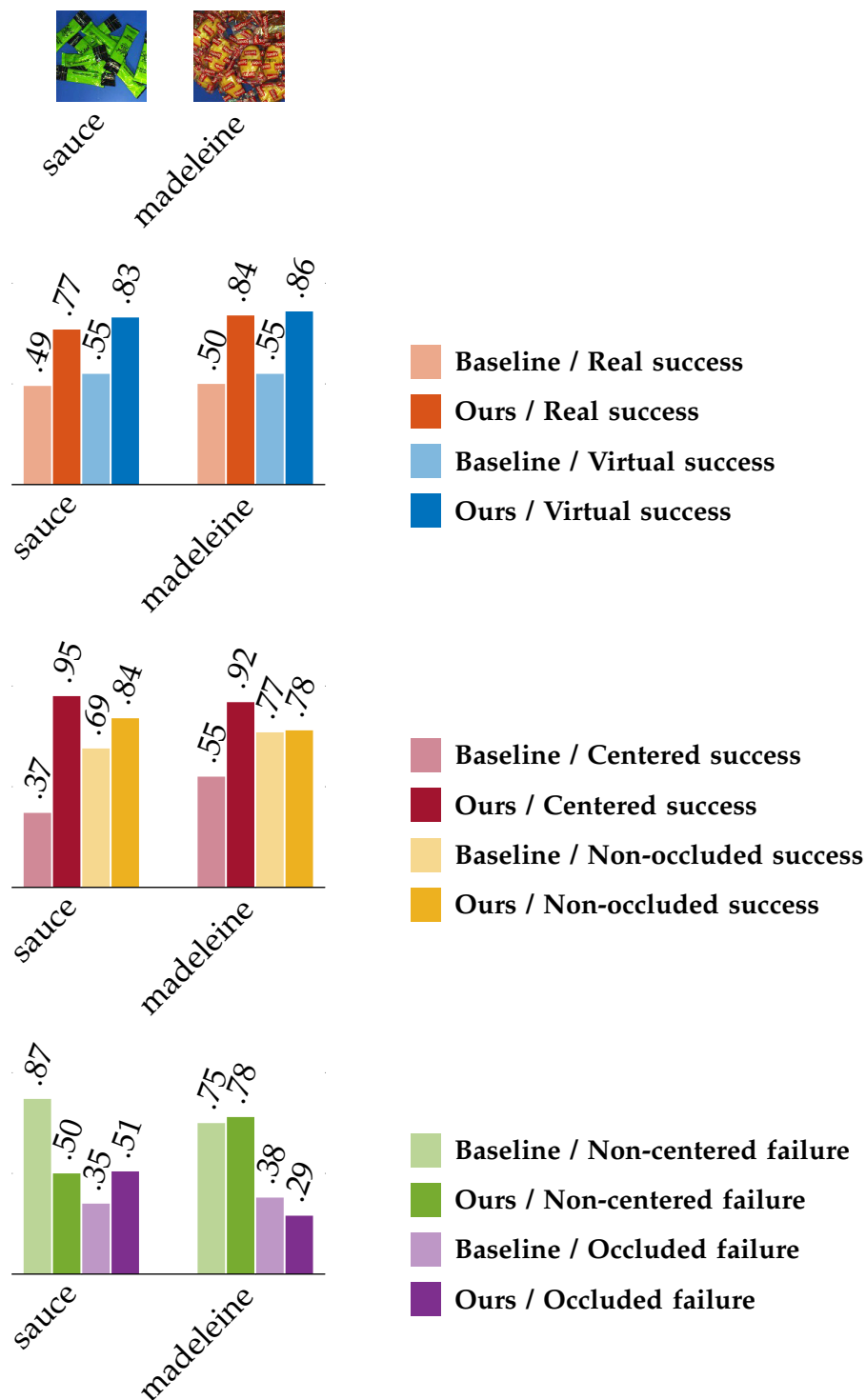
Figure 5.20: Comparison of the proposed approach, using a bicameral network trained on Mikado+, with the industrial baseline. Unlike the gripper-oriented baseline, which detects grasps independently of the object model but without explicit notion of instance, our approach enables instance-centered grasps and avoids occluded instances, hence many more successful extractions.

Dimensionality Reduction   The bicameral network inference duration is intrinsically linked to the number of parameters, and more specifically to the number of feature maps and applied filters at each convolutional layer. Lessening the number of filters however reduces the dimensionalities for encoding information, so the performances. We thus study how reducing the number of filters in the bicameral network impacts the performances on synthetic data and in real-world bin-picking conditions.

	F100	F75	F50	F25
conv1X	64	48	32	16
conv2X	128	96	64	32
conv3X	256	192	128	64
conv4X	512	384	256	128
conv5X	512	384	256	128
deconv4x	256	192	128	64
deconv3x	128	96	64	32
deconv2x	64	48	32	16
deconv1x	32	24	16	8
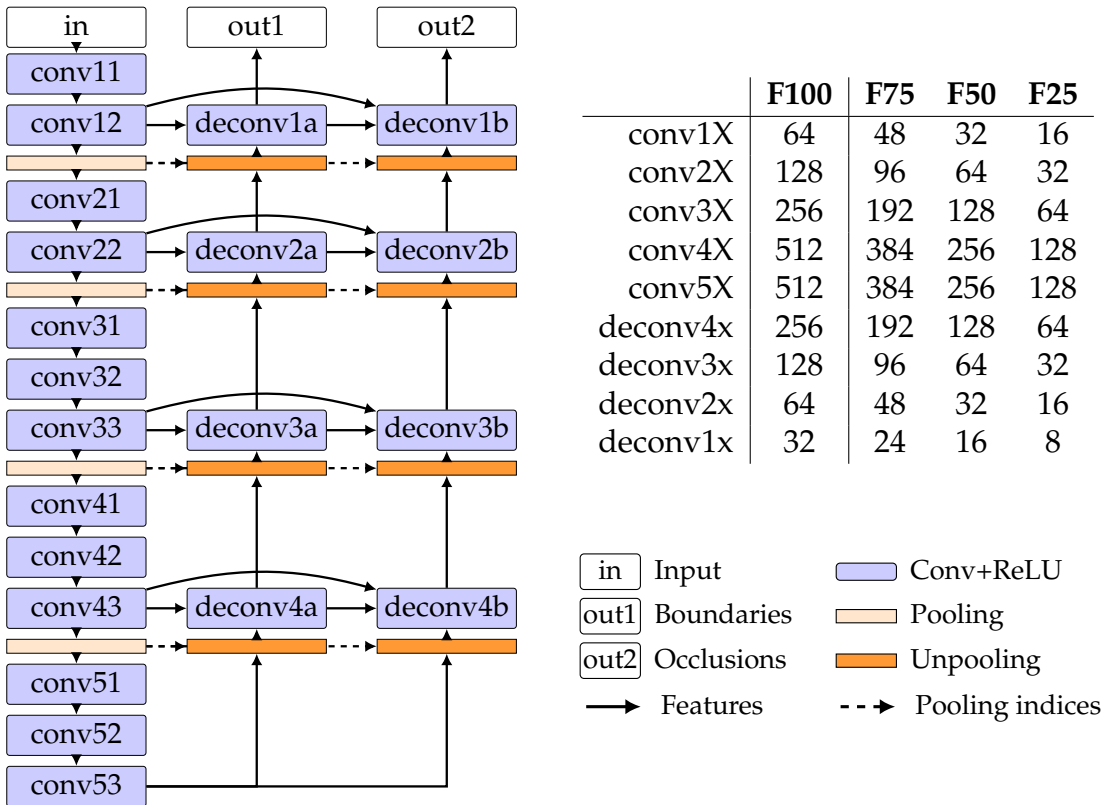
Figure 5.21: Number of output feature maps after each convolutional layers, for different dimensionality reductions. F100 refers to our canonical VGG16-based architecture used in the previous experiments. Legend is the same as Figure 4.2. Best viewed in color

Figure 5.21 presents three variants of a bicameral network, namely F75, F50 and F25, retaining respectively only 75%, 50%, and 25% of the initial number of convolutional filters. As reported by Table 5.5, reducing the number of filters induces a gradual loss of performances on Mikado. Using the lightest variant (F25), AP drops by 1.4 and 1.2 points for boundaries and occlusions respectively. This observation is corroborated by the results of transfer learning (*c.f.* Section 5.2) from Mikado+ to D2SA in Figure 5.22. The performance drop on D2SA is nevertheless

Bicameral architecture	Number of parameters	Boundaries		Occlusions	
		ODS	AP	ODS	AP
F100	34,301,250 (×1.0)	.769	.847	.801	.884
F75	19,296,050 (×.56)	.756	.832	.791	.876
F50	8,577,442 (×.25)	.730	.802	.768	.852
F25	2,145,426 (×.06)	.657	.706	.691	.765

Table 5.5: Cross-validated performances for instance boundary and unoccluded side detection on Mikado using a bicameral structuring trained on Mikado, with respect to the percentage of initial number of convolutional filters (*c.f.* Figure 5.21). Reducing the number of convolutional filters in a bicameral network induces a gradual loss of performances.
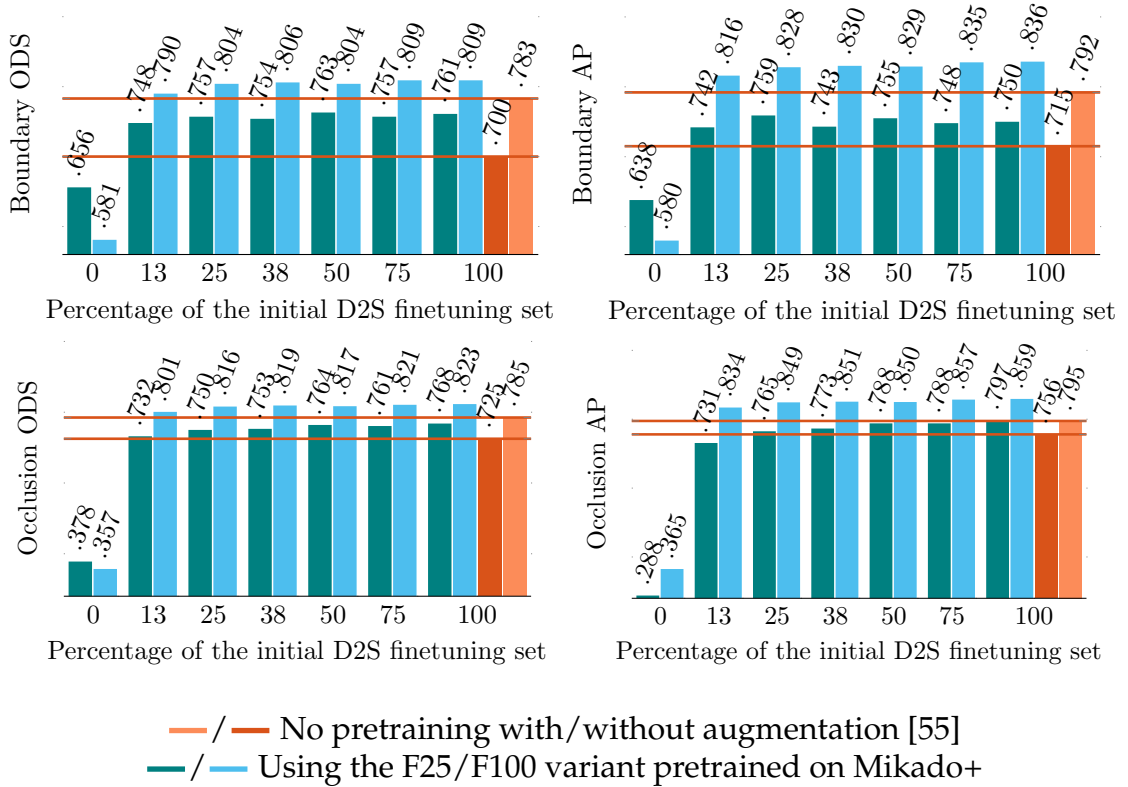
Figure 5.22: Cross-validated performances on D2SA of the proposed network pretrained on Mikado+ then finetuned on D2SA with the encoder blocks 1, 2, 3 frozen, with respect to the percentage of D2SA images retained for finetuning and the percentage of the initial number of filters (F25 or F100). Despite an expected performance drop compared with a full bicameral design (F100) pretrained on Mikado+ or trained using the data augmentation strategy of [55], retaining only 25% of the initial number of convolutional filters (F25) still enables to achieve state-of-the-art performances for boundaries and occlusions.
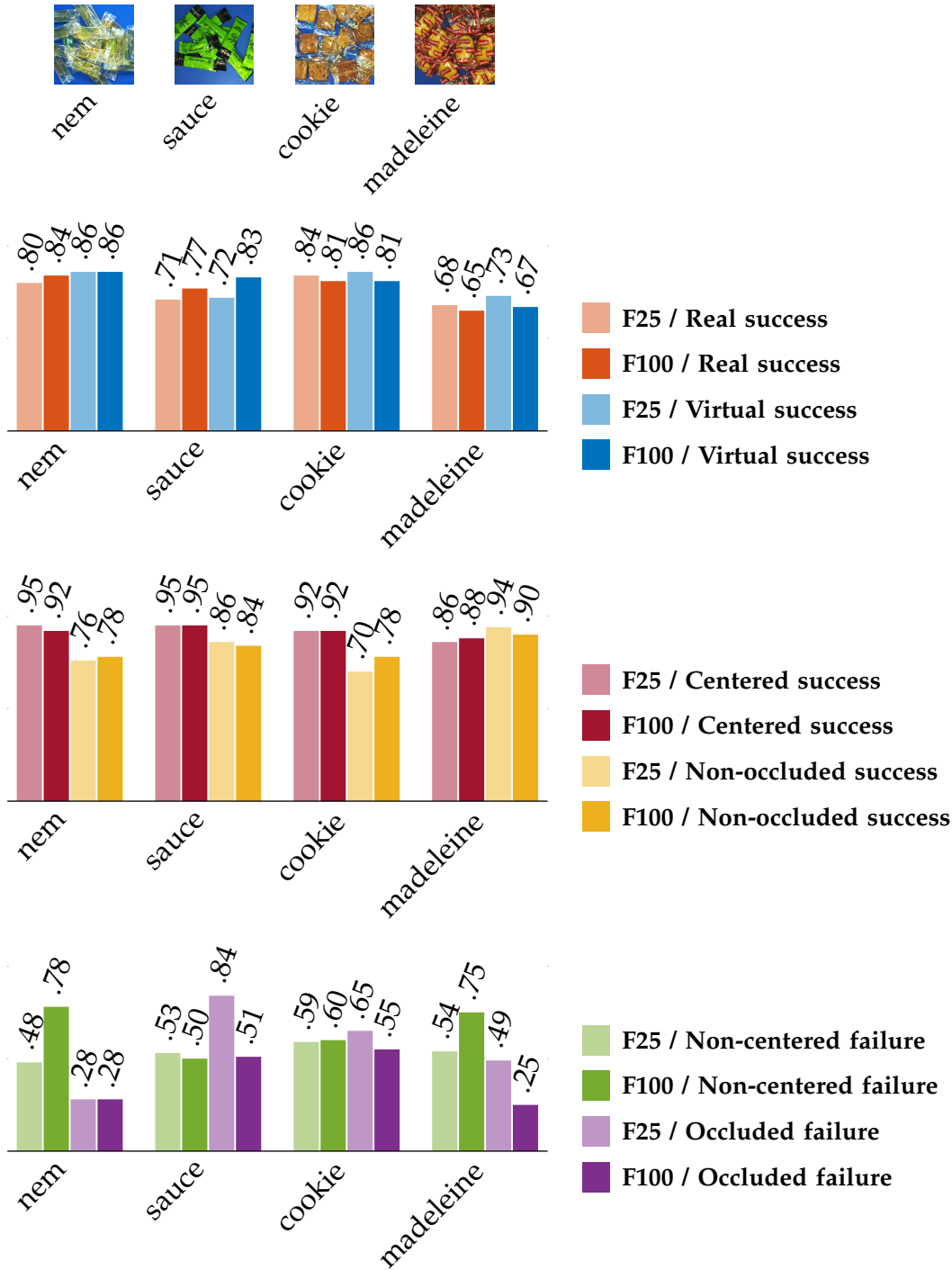
Figure 5.23: Per-product performances using a bicameral network trained on Mikado+, with respect to the percentage of number of convolutional filters (F25 or F100). In anticipation of computation time reduction, the number of convolutional filters can be reduced without severe performance loss.
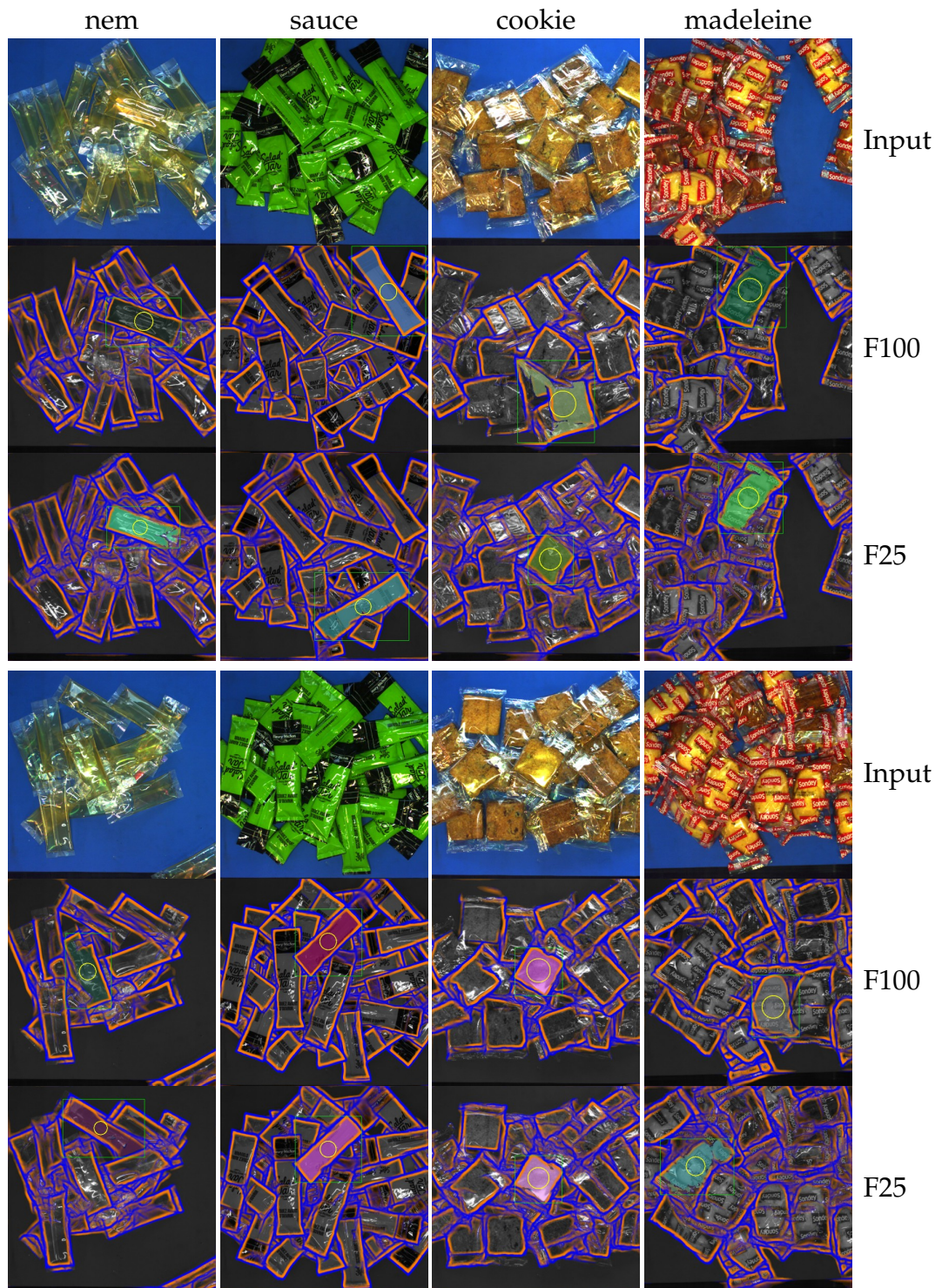
Figure 5.24: Examples of successful extractions using the F100 and F25 bicameral variants trained on Mikado+ (best viewed in electronic form). Although the boundaries and occlusions are slightly degraded along with the number of filters (F25), both bicameral variants lead to comparable real-world bin-picking performances. See also videos "binpicking02" and "binpicking03" in supplementary material for real-time results with the F25 variant.

below 1 point. More interestingly, both ODS and AP for boundaries and occlusions remain higher than the baseline scores obtained without pretraining on Mikado+, thus suggesting that a bicameral network with only 25% of the initial number of filters can still encode representations generalizable to real-world images. In real-world bin-picking conditions, the F25 variant achieves success rates comparable to those of F100. As shown by Figure 5.23 (see Table C.6 for more details), which reports the comparative bin-picking results between F25 and F100 on nem, sauce, cookie and madeleine, the performance loss remains strictly below 5 points. The benefits of our approach, *i.e.* grasps centered on non-occluded instances, are maintained as well: on average, 92% of succesful extractions are centered and 82% on unoccluded instances (see Figure 5.24 for qualitative results). For nem and sauce, a larger fraction of failures is due to grasps on partially occluded instances, in accordance with the lesser capability to separate instances reported by the scores on Mikado and D2SA (*c.f.* Table 5.5 and Figure 5.22).

Overall Computation Time   We showed that the number of convolutional filters in a bicameral network can be reduced without severe performance loss for real-world bin-picking. We now quantify the expected gain in computation time from this dimensionality reduction. Specifically, we first benchmark the proposed method on two GPU-enabled hardwares: a power-limited device for embedded applications, the Nvidia Jetson TX2, and a regular computer referred to as PC Win10 (see Figure 5.25 for detailed specifications). We then provide insights on the bottlenecks and factors impacting the computation times.

As reported by Figure 5.25, the overall method takes on average per image 1,848ms on Jetson TX2 and 136ms on PC Win10, while the industrial baseline 2,411ms on PC Win10. Our method's elapsed time is reduced to 694ms and 81ms respectively using instead the F25 bicameral variant. Specifically, retaining only 25% of the initial number of convolutional filters in the bicameral network reduces by 84% and 81% the inference duration on Jetson TX2 and PC Win10 respectively, as shown by Figure 5.26.

Impact Factors and Bottlenecks   Our method is two-step: first, inferring instance boundaries and occlusions using a bicameral network, second, localizing and ranking instances from the network inference (see Table 5.6). Contrary to the second step, the bicameral network inference is independent of the number of instances in the image, and fully executed on GPU. Our current implementation of the second step is 90% on GPU and 10% on CPU. By default (F100), the network inference is the bottleneck, *i.e.* 89% of the overall computation time on
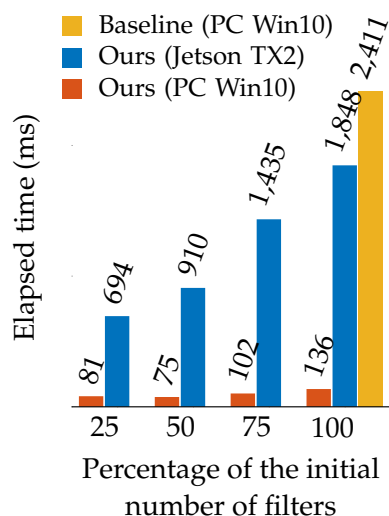
	Jetson TX2	PC Win10
CPU	ARMv8 Cortex-A57 4 cores @ 2GHz + NVIDIA Denver2 2 cores @ 2GHz	Intel Core i7 7700 4 cores @ 3.6GHz
GPU	NVIDIA Tegra X2 Pascal Architecture 256 cores @ 1300MHz	NVIDIA GTX 1080Ti Pascal Architecture 3584 cores @ 1500MHz
RAM	CPU/GPU 8Gb (shared)	CPU 32Gb GPU 11Gb
OS	Ubuntu 16	Windows 10

Figure 5.25: Elapsed time (ms) on two hardwares for generating the next grasp coordinates from a single 512×512 image using our approach (blue and red), with respect to the number of convolutional filters, and compared with the industrial baseline (yellow). Note: the industrial baseline uses only the CPU, while our approach is mostly GPU-based.
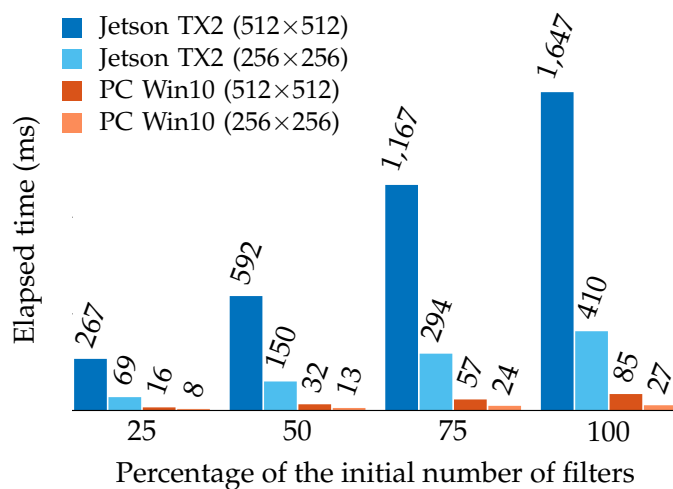
Figure 5.26: Elapsed time (ms) for the bicameral network inference alone, with respect to the percentage of the initial number of convolutional filters, using different hardwares (*c.f.* Figure 5.25) and different image resolutions. Each time value is an average over 100 consecutive iterations using only C++/CUDA code.
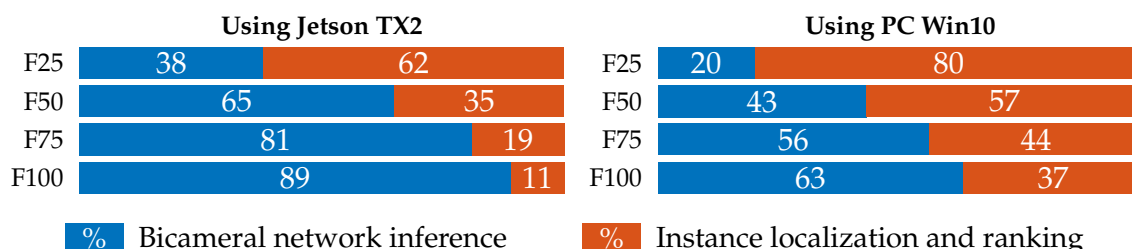
Figure 5.27: Computation time repartition with respect to the percentage of the initial number of filters (F), using different hardwares (*c.f.* Fig. 5.25)

Impact factors	Inferring boundaries and occlusions	Localizing and ranking instances
Number of pixels	✓	✓
Number of GPU cores	✓	✓
Number of filters	✓	
Number of instances		✓

Table 5.6: Factors impacting the computation time for each of the two steps of the proposed method

Jetson TX2 and 63% on PC Win10. Lessening the number of filters gradually reduces the percentage of the overall computation time taken by the bicameral network (see Figure 5.27). Using the F25 variant, the elapsed time repartition is then very different: only 38% for the network on Jetson TX2 and 20% on PC Win10. The computation time repartition is different depending on the hardware, because a GTX 1080Ti contains many more cores than a Tegra X2 for the same task. This results in different levels of occupancy, *i.e.* the ratio of active *warps*[2] on a Streaming Multiprocessor (SM) to the maximum number of active warps supported by the SM.

Our method's second step is dependent on the number of instances in the image. As reported by Figure 5.28a, the overall computation time and the variations of this elapsed time increase with a higher number of instances. Using a default number of filters (F100), the elapsed time for 1 to 35 instances varies from 1,700ms to 2,100ms on Jetson TX2 and on from 100ms to 200ms on PC Win10. Using the F25 bicameral variant, these intervals are dropped to 490–1,600ms and 30–140ms on Jetson TX2 and PC Win10 respectively. These variations are intrinsically correlated to the bicameral network inference. Specifically, inferred boundary maps with many false positive are likely to produce many connected components, proportionately to the number of instances in the image. Figure 5.28b empirically shows this correlation for each bicameral variant.

5.4 Conclusion

5.4.1 Summary

In this chapter, we applied a bicameral network, introduced in Chapter 4, to real-world bin-picking. As there is no annotated real data for our applications and collecting such data is unsustainable in industrial

---

[2]A warp is a set of threads running concurrently on a SM.
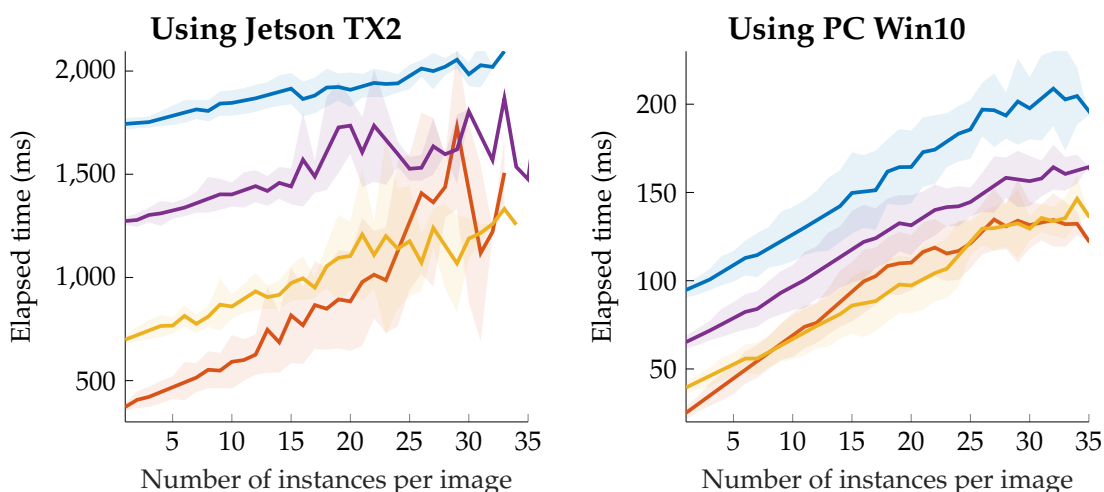
(a) Overall elapsed time using different hardwares (*c.f.* Figure 5.25). Both the elapsed time and the variations of this elapsed time increase with a higher number of instances.
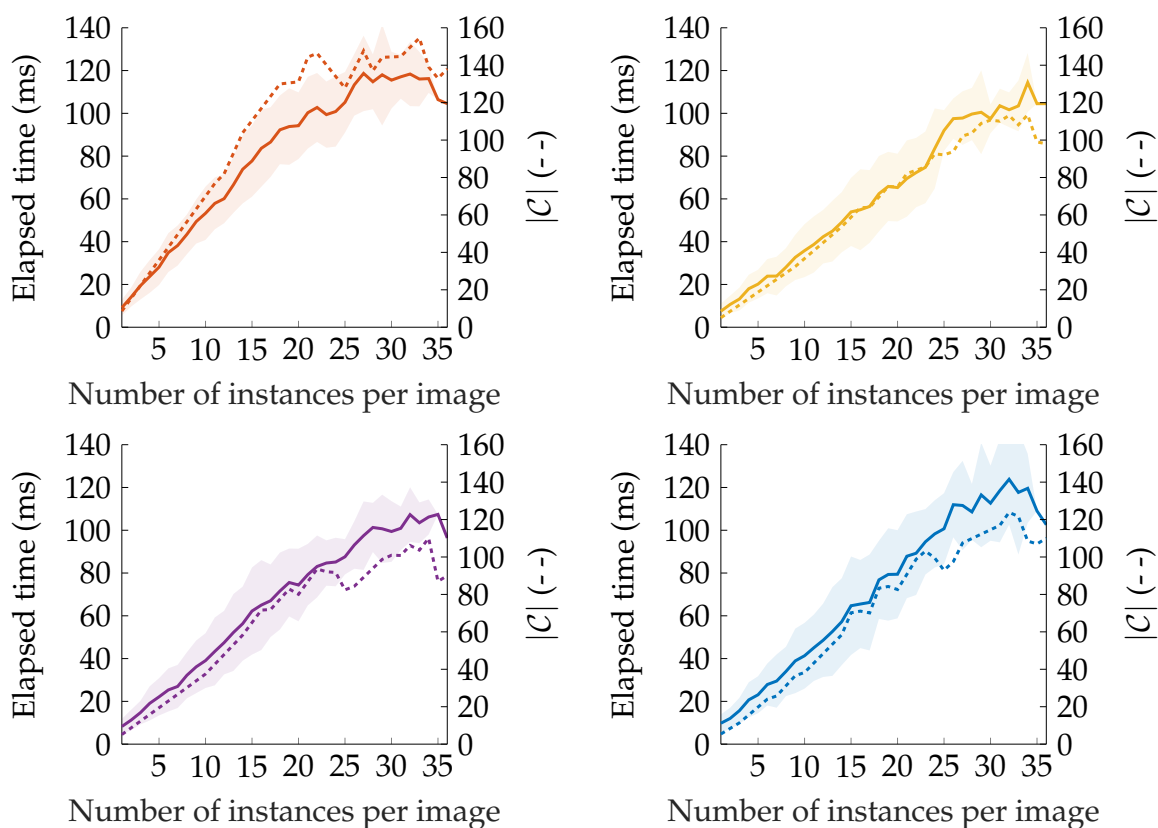
(b) Elapsed time excluding the network inference duration, using PC Win10 (*c.f.* Figure 5.25). Right axis and dashed lines: number $|\mathcal{C}|$ of connected components generated from the network inference. This shows that the computation time variations are strongly correlated to the number of connected components, thereby to the quality of the bicameral network inference.

Figure 5.28: Elapsed time (ms) and corresponding standard variation for detecting the most affordable instance from a single 512×512 image using the proposed approach, with respect to the number of instances per image and the percentage of initial convolutional filters.

processes, we first proposed a simulation-based pipeline to generate synthetic training images from off-the-shelf rendering and physics engines. We then checked the plausibility of such computer-generated data for real-world applications by evaluating the transferability of the learned representations to a real-world target domain. We finally conducted extensive experiments over ten types of packaged food products on a real-world robotic setup using synthetically trained bicameral models without any finetuning on real images. Specifically, we analyzed more than 130 real-world bin-picking sequences, *i.e.* more than 3,000 observations, to quantify the results obtained with a synthetic training in terms of successful extractions and computation time. Our three-fold experimental analysis focused on:

1. the robustness to the real-world product variations and light changes, with respect to the synthetic data distribution;

2. the benefits of our approach's properties compared with the object model-free gripper-oriented industrial baseline;

3. the conditions for real-time performances, with respect to the number of convolutional filters in a bicameral network.

5.4.2 Contributions

Plausible Synthetic Training Data   The proposed training data generation pipeline enables to extensively produce **unbiased pixel-wise ground-truth annotations**, while drastically reducing the workload for human annotators, whose task then consists only in setting up the simulator. On a GPU-enabled computer, the simulation of a bin-picking scene and the corresponding top-view image takes on average 5min, against more than 30min for manually annotating a real image.

The proposed **Mikado synthetic data proves plausible for real-world applications** in the sense that it enables the learning of performance-enhancing representations transferable to real data. Specifically, our experiments on transfer learning from Mikado to D2SA [55] shows that reusing the local features learned from Mikado increases AP by up to 10 points over the baseline, and up to 5 points over the augmentation strategy of [55] as well. Furthermore, enriching the synthetic data distribution (Mikado+) with more geometry, texture, and light variations enables to still achieve state-of-the-art performances on D2SA while reducing by up to 13% the number of real images for finetuning.

Real-Time State-of-the-Art Performances   In real-world bin-picking conditions, **a bicameral network trained on Mikado+, without any**

finetuning on real images, demonstrated a success probability of 80% on average over ten different packaged food products, excluding the failures due to gripper non-adhesiveness. Our synthetically trained model achieves stationnary performances despite light changes, thus opening the path to more versatile industrial infrastructures. As proposed, introducing the notion of instance enables a higher success rate due to more detected grasps centered on non-occluded instances. As a result, the proposed approach outperforms the object model-free gripper-oriented industrial baseline by near 30 points.

Real-time performances are achieved on a GPU-enabled regular computer (more than 3,500 GPU cores), thus meeting the conditions of high-throughput bin-picking applications, *i.e.* overall computation times under 100ms. On a power-limited embedded device, near real-time performances are achieved by reducing the number of filters in the bicameral network. For example, the proposed method takes on average less than 700ms on the Jetson TX2 (256 cores) using only 25% of the initial number of convolutional filters, without severe performance loss in terms of successful extractions.

As a result, the proposed object-oriented approach establishes a new baseline for model-free bin-picking.

Chapter 6

Conclusion

In this chapter, we summarize our work and contributions. We finally draw some research directions for future work.

6.1   Summary

We addressed the problem of generic instance segmentation in the context of robotic random bin-picking, *i.e.* the task of unpiling many object instances on top of each other with a robotic arm for feeding automated lines in industrial environments. Current industrial approaches consist in detecting either instance poses of an explicit object model, or grasp opportunities from a gripper-dependent physics model. Such approaches however prove unsuitable for model-free or deformable objects, and piles with strong inter-instance occlusions respectively. Our object-oriented approach thus introduces an explicit notion of instance independently of the object and gripper models. Specifically, our approach consists in detecting the most affordable instances of a pile by jointly delineating instances and inferring their spatial layouts from a single image, using a novel fully convolutional network (FCN) trained on synthetic data.

After briefly laying the mathematical tools related to deep convolutional networks in Chapter 2, we reviewed in Chapter 3 the state of the art on generic instance segmentation, whose mainstream strategy follows a two-step paradigm: first, detecting rectangle region proposals that might contain an instance; second, coloring the instance supposedly inside each box proposal by a binary segmentation. The notion of occlusion is then introduced by additionally learning the amodal mask, *i.e.* the mask including both the visible and occluded instance parts. This approach however cumulates the difficulties of isolating bulk instances in rectangles and coloring something invisible. Alternatively, the instance boundaries and their occlusion-based orientation are

detected separately, while boundaries are mostly caused by occlusions in bin-picking scenes. Both of these approaches also rely on hand-made pixel-wise annotations, which can hardly be collected in industrial contexts.

In Chapter 4, we analyzed the performances of the proposed network architecture, referred to as *bicameral*, which is composed of a deep encoder shared by two cascaded decoders. Specifically, we first compared the bicameral FCN with the state-of-the-art networks for oriented boundary detection and box proposal-based amodal segmentation on real-world and synthetic data. We then conducted an ablation study for exposing the role of the bicameral characteristics. We finally described how to employ a bicameral network inference for detecting the most affordable instance of a pile.

In Chapter 5, we addressed the problem of deploying a bicameral network into real-world bin-picking applications. We first described a cost-effective training data generation pipeline based on physics and rendering engines, referred to as *Mikado*. We then conducted transfer learning experiments to check the plausibility of the proposed synthetic training data. Finally, we developed a real-world robotic setup to deploy our model on real-world piles of packaged food products. We tested different training settings and architectural variations, in various lighting conditions, to quantify their impact on the overall picking performances.

6.2 Contributions

Our contribution for object-oriented bin-picking is two-fold: **joint representation learning for boundary and occlusion detection** and **abstract learning from synthetic data**, independently of the explicit object and gripper models.

Specifically, we demonstrated in Chapter 4 that, using a carefully designed fully convolutional network, learning instance boundaries and occlusions in a single feature space enables to outperform the state-of-the-art two-stream representations for the same task on synthetic and real-world data, consistently with the observation that boundaries and occlusions are strongly correlated in bin-picking scenes. In addition, decoding such a single representation by cascading two convolutional decoders enables a performance-enhancing encoder-decoder structure for recovering accurate instance boundaries augmented in cascade with an occlusion-based orientation.

In Chapter 5, we showed that plausible synthetic training data can be generated using off-the-shelf rendering and physics engines, thus eliminating the burden of hand-made pixel-wise annotations. Our ex-

tensive experimental study first reported that pretraining a bicameral FCN for oriented boundary detection on our synthetic data, instead of using only target real images, increases the real-world cross-dataset performances, as the learned local representations prove more transferable to real data. Enriching the synthetic data distribution by randomizing the textures and backgrounds in simulation also enables to learn more abstract local invariants, while reducing by more than 85% the number of real images for finetuning.

Our proof-of-concept experiments on a real-world robotic setup finally demonstrated state-of-the-art performances over ten various model-free deformable objects, high and low-textured, transparent and opaque, thus opening a promising land for large-scale industrial applications towards fully automated factories. In practice, a synthetically trained bicameral network (without finetuning on real data) achieves real-time picking performances, with an **average success probability of 80% in less than 100ms per image** on a single GPU-equipped computer, thus meeting the typical conditions of high-throughput bin-picking applications. As a result, we established a new baseline for model-free object-oriented bin-picking.

6.3   Perspectives

This work unveils long-term perspectives towards adaptive autonomous robotics in unknown environments.

In addition to detecting instance boundaries and occlusions, the proposed network architecture can be intuitively extended to become **"multicameral"** for learning other complementary tasks, such as instance localization, object categorization and grasp detection, within a complete end-to-end synthetic training. Specifically, the task of instance localization can be integrated by embedding the notion of pixel connectivity into each pixel feature [115, 137]. The task of learning explicit object categories by pixel-wise categorization [35] should provide more information to separate instances in the case of heterogeneous piles, *e.g.* in waste sorting applications, and should enable to discard the background from the set of instance candidates in the case of homogeneous piles. Ultimately, the image representations learned using such a multicameral architecture can gain more robustness to pose variations by introducing pose-equivariant learning modules instead of convolutional layers, as suggested by [160].

From a more general point of view, the proposed model only learns then applies his knowledge, hence stationary performances but without paths for improvement over time. Recent works on meta-learning however suggest that our model could **learn to learn** as well. Specifically,

recent studies on unsupervised domain adaptation showed that our synthetic training can include a notion of cross-domain similarity by learning to compare cross-domain features with learned prototypes [145]. If a long-term collection of real images is possible during production, then generative adversarial networks could be employed as well to refine our synthetic images so that their aspect better matches the real-world camera model without the need of hand-made labels [22, 23]. In such unsupervised domain adaption frameworks, our network should be retrained on a distant server to avoid penalizing the production, and using a careful incremental learning strategy that avoids catastrophic forgetting [31, 85]. More interestingly, an online backpropagation-free adaptation of a trained model is possible by few-shot dynamic learning [58, 204]. In such a training, a convolutional model learns from little online data to adapt some of its parameters to new tasks, *e.g.* novel object categories and lighting conditions in our application context.

For budget-constrained applications, our synthetically trained models must be **further compressed for achieving high-throughput application-compliant performances on low-powered embedded devices**, such as the Jetson devices from NVIDIA. As suggested by our experiments in Chapter 5, real-time performances using the proposed network on an embedded device are achievable by reducing the number of filters. More advanced studies on the topic have shown that a trained model can be further pruned while limiting the performance drop. Specifically, [195] introduced the class of so-called slimmable networks, *i.e.* networks with layer depths that can be dynamically adjusted, for permitting instant and adaptive accuracy-efficiency trade-offs at runtime. In the same vein, [30] employed a network architecture search algorithm to find the architecture achieving the optimal trade-off between accuracy and latency. Interestingly, such strategies can be used for training networks with conditional input or output as well. For example, one can imagine a multicameral model with a switchable branch for inferring object categories, activated for heterogeneous piles and deactivated for homogeneous piles.

Bibliography

[1] HALCON. https://www.mvtec.com/products/halcon/, 2019. MVTec Software Gmbh.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[3] W. Abbeloos and T. Goedemé. Point Pair Feature Based Object Detection for Random Bin Picking. In *Conference on Computer and Robot Vision (CRV)*, pages 432–439, 2016.

[4] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2274–2282, 2012.

[5] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis Machine Intelligence (TPAMI)*, 33(5):898–916, 2011.

[6] A. Arnab and P. H. S. Torr. Pixelwise Instance Segmentation with a Dynamically Instantiated Network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 879–888. IEEE Computer Society, 2017.

[7] U. Asif, M. Bennamoun, and F. A. Sohel. Model-Free Segmentation and Grasp Selection of Unknown Stacked Objects. In *European Conference on Computer Vision (ECCV) Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 659–674. Springer, 2014.

[8] 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 Conference on Artificial Intelligence (IJCAI)*, pages 4875–4882, 2018.

[9] A. Ayvaci, M. Raptis, and S. Soatto. Occlusion Detection and Motion Estimation with Convex Optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 100–108, 2010.

[10] A. Ayvaci, M. Raptis, and S. Soatto. Sparse Occlusion Detection with Optical Flow. *International Journal of Computer Vision (IJCV)*, 97(3):322–338, 2012.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.

[12] M. Bai and R. Urtasun. Deep Watershed Transform for Instance Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2858–2866. IEEE Computer Society, 2017.

[13] A. Batra, S. Singh, G. Pang, S. Basu, C. Jawahar, and M. Paluri. Improved Road Connectivity by Joint Learning of Orientation and Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10385–10393. IEEE Computer Society, 2019.

[14] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[15] S. Ben-David, T. Lu, T. Luu, and D. Pál. Impossibility Theorems for Domain Adaptation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR Proceedings*, pages 129–136. JMLR.org, 2010.

[16] G. Bertasius, J. Shi, and L. Torresani. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4380–4389. IEEE Computer Society, 2015.

[17] G. Bertasius, J. Shi, and L. Torresani. High-for-Low and Low-for-High: Efficient Boundary Detection from Deep Object Features and Its Applications to High-Level Vision. In *International Conference on Computer Vision (ICCV)*, pages 504–512. IEEE Computer Society, 2015.

[18] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[19] A. Bietti and J. Mairal. Invariance and Stability of Deep Convolutional Representations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6211–6221, 2017.

[20] T. Birdal and S. Ilic. Point Pair Features Based Object Detection and Pose Estimation Revisited. In *International Conference on 3D Vision (3DV)*, pages 527–535, 2015.

[21] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2016.

[22] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. In *International Conference on Robotics and Automation (ICRA)*, pages 4243–4250. IEEE, 2018.

[23] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 95–104. IEEE Computer Society, 2017.

[24] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *European Conference on Computer Vision (ECCV) Part II*, volume 8690 of *Lecture Notes in Computer Science*, pages 536–551. Springer, 2014.

[25] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. Software available from opencv.org.

[26] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley. Symmetry Aware Evaluation of 3D Object Detection and Pose Estimation in Scenes of Many Parts in Bulk. In *International Conference on Computer Vision Workshops (ICCVW)*, pages 2209–2218. IEEE Computer Society, 2017.

[27] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley. Defining the Pose of Any 3D Rigid Object and an Associated Distance. *International Journal of Computer Vision (IJCV)*, 126(6):571–596, 2018.

[28] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger. Universal robotic gripper based on the jamming of granular material. *Proceedings of the National Academy of Sciences (PNAS)*, 107(44):18809–18814, 2010.

[29] H. Caesar, J. R. R. Uijlings, and V. Ferrari. COCO-Stuff: Thing and Stuff Classes in Context. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1209–1218. IEEE Computer Society, 2018.

[30] H. Cai, L. Zhu, and S. Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations (ICLR)*, 2019.

[31] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-End Incremental Learning. In *European Conference on Computer Vision (ECCV) Part XII*, volume 11216 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2018.

[32] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *Computing Research Repository (CoRR)*, abs/1512.03012, 2015.

[33] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam. MaskLab: Instance Segmentation by Refining Object Detection With Semantic and Direction Features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4013–4022. IEEE Computer Society, 2018.

[34] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2018.

[35] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *European Conference on Computer Vision (ECCV) Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 833–851. Springer, 2018.

[36] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *Computing Research Repository (CoRR)*, abs/1512.01274, 2015.

[37] Y. Chen, J. Yang, and M. Yang. Extracting Image Regions by Structured Edge Prediction. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 1060–1067. IEEE Computer Society, 2015.

[38] C. Choi, Y. Taguchi, O. Tuzel, M. Liu, and S. Ramalingam. Voting-based pose estimation for robotic assembly using a 3D sensor. In *International Conference on Robotics and Automation (ICRA)*, pages 1724–1731. IEEE, 2012.

[39] F. Chu, R. Xu, and P. A. Vela. Real-World Multi-object, Multi-grasp Detection. *IEEE Robotics and Automation Letters*, 3(4):3355–3362, 2018.

[40] M. Cissé, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval Networks: Improving Robustness to Adversarial Examples. In *International Conference on Machine Learning (ICML)*, volume 70, pages 854–863. PMLR, 2017.

[41] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–387, 2016.

[42] M. V. den Bergh, X. Boix, G. Roig, and L. J. V. Gool. SEEDS: Superpixels Extracted Via Energy-Driven Sampling. *International Journal of Computer Vision (IJCV)*, 111(3):298–314, 2015.

[43] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu. Learning to Predict Crisp Boundaries. In *European Conference on Computer Vision (ECCV) Part VI*, volume 11210 of *Lecture Notes in Computer Science*, pages 570–586. Springer, 2018.

[44] A. Depierre, E. Dellandréa, and L. Chen. Jacquard: A Large Scale Dataset for Robotic Grasp Detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3511–3516. IEEE, 2018.

[45] T.-T. Do, A. Nguyen, and I. D. Reid. AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection. In *International Conference on Robotics and Automation (ICRA)*, pages 1–5. IEEE, 2018.

[46] P. Dollár and C. L. Zitnick. Structured Forests for Fast Edge Detection. In *International Conference on Computer Vision (ICCV)*, pages 1841–1848. IEEE Computer Society, 2013.

[47] Y. Domae, H. Okuda, Y. Taguchi, K. Sumi, and T. Hirai. Fast graspability evaluation on single depth maps for bin picking with general grippers. In *International Conference on Robotics and Automation (ICRA)*, pages 1997–2004. IEEE, 2014.

[48] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T. Kim. Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3583–3592. IEEE Computer Society, 2016.

[49] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research (JMLR)*, 12:2121–2159, 2011.

[50] D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2366–2374, 2014.

[51] M. Everingham, S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015.

[52] Facebook. Caffe2: A New Lightweight, Modular, and Scalable Deep Learning Framework. Software available from caffe2.ai, 2017.

[53] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941. IEEE Computer Society, 2016.

[54] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.

[55] P. Follmann, T. Böttger, P. Härtinger, R. König, and M. Ulrich. MVTec D2S: Densely Segmented Supermarket Dataset. In *European Conference on Computer Vision (ECCV) Part X*, volume 11214 of *Lecture Notes in Computer Science*, pages 581–597. Springer, 2018.

[56] H. Fu, C. Wang, D. Tao, and M. J. Black. Occlusion Boundary Detection via Deep Exploration of Context. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 241–250. IEEE Computer Society, 2016.

[57] D. Geiger, B. Ladendorf, and A. L. Yuille. Occlusions and binocular stereo. *International Journal of Computer Vision (IJCV)*, 14(3):211–226, 1995.

[58] S. Gidaris and N. Komodakis. Dynamic Few-Shot Visual Learning Without Forgetting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4367–4375. IEEE Computer Society, 2018.

[59] R. B. Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 1440–1448. IEEE Computer Society, 2015.

[60] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. IEEE Computer Society, 2014.

[61] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[62] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[63] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[64] N. Grammalidis and M. G. Strintzis. Disparity and occlusion estimation in multiocular systems and their coding for the communication of multiview image sequences. *Transactions on Circuits and Systems for Video Technology (TCSVT)*, (3):328–344.

[65] C. Grana, D. Borghesani, and R. Cucchiara. Optimized Block-Based Connected Components Labeling With Decision Trees. *IEEE Transactions on Image Processing*, 19(6):1596–1609, 2010.

[66] M. Grard, R. Brégier, F. Sella, E. Dellandréa, and L. Chen. Object Segmentation in Depth Maps with One User Click and a Synthetically Trained Fully Convolutional Network. In *2017 International Workshop on Human-Friendly Robotics*, volume 7 of *Springer Proceedings in Advanced Robotics*. Springer, 2018.

[67] M. Grard, E. Dellandréa, and L. Chen. A Bicameral Decoder for Jointly Predicting Instance Boundaries and Nearby Occlusions

from a Single Image. In International Journal of Computer Vision (IJCV), Special Issue on Deep Learning for Robotic Vision. Submission in July 2018. First revision in January, 2018.

[68] S. Gupta, R. B. Girshick, P. A. Arbeláez, and J. Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *European Conference on Computer Vision (ECCV) Part VII*, volume 8695 of *Lecture Notes in Computer Science*, pages 345–360. Springer, 2014.

[69] K. Harada, K. Nagata, T. Tsuji, N. Yamanobe, A. Nakamura, and Y. Kawai. Probabilistic approach for object bin picking approximated by cylinders. In *International Conference on Robotics and Automation (ICRA)*, pages 3742–3747. IEEE, 2013.

[70] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik. Simultaneous Detection and Segmentation. In *European Conference on Computer Vision (ECCV) Part VII*, volume 8695 of *Lecture Notes in Computer Science*, pages 297–312. Springer, 2014.

[71] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[72] M. A. Hasnat, O. Alata, and A. Trémeau. Unsupervised RGB-D image segmentation using joint clustering and region merging. In *British Machine Vision Conference (BMVC)*, 2014.

[73] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE Computer Society, 2017.

[74] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *European Conference on Computer Vision (ECCV) Part III*, volume 8691 of *Lecture Notes in Computer Science*, pages 346–361. Springer, 2014.

[75] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE Computer Society, 2016.

[76] S. He and R. W. H. Lau. Oriented Object Proposals. In *International Conference on Computer Vision (ICCV)*, pages 280–288. IEEE Computer Society, 2015.

[77] S. He, R. W. H. Lau, W. Liu, Z. Huang, and Q. Yang. SuperCNN: A Superpixelwise Convolutional Neural Network for Salient

Object Detection. *International Journal of Computer Vision (IJCV)*, 115(3):330–344, 2015.

[78] X. He and A. Yuille. Occlusion Boundary Detection Using Pseudo-depth. In *European Conference on Computer Vision (ECCV) Part IV*, volume 6314 of *Lecture Notes in Computer Science*, pages 539–552. Springer, 2010.

[79] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *International Conference on Computer Vision (ICCV)*, pages 858–865. IEEE Computer Society, 2011.

[80] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2257–2264. IEEE Computer Society, 2010.

[81] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. R. Bradski, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Asian Conference on Computer Vision (ACCV) Part I*, volume 7724 of *Lecture Notes in Computer Science*, pages 548–562. Springer, 2012.

[82] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering Occlusion Boundaries from a Single Image. In *International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE Computer Society, 2007.

[83] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus. Haptic identification of objects using a modular soft robotic gripper. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1698–1705. IEEE, 2015.

[84] J. H. Hosang, R. Benenson, P. Dollár, and B. Schiele. What Makes for Effective Detection Proposals? *IEEE Transactions on Pattern Analysis Machine Intelligence (TPAMI)*, 38(4):814–830, 2016.

[85] L. Hou, A. Agarwal, D. Samaras, T. M. Kurc, R. R. Gupta, and J. H. Saltz. Robust Histopathology Image Analysis: to Label or to Synthesize? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8533–8542. IEEE Computer Society, 2019.

[86] R. Hu, P. Dollár, K. He, T. Darrell, and R. B. Girshick. Learning to Segment Every Thing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4233–4241. IEEE Computer Society, 2018.

[87] Huang, Gao and Liu, Zhuang and van der Maaten, Laurens and Weinberger, Kilian Q. Densely Connected Convolutional Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269. IEEE Computer Society, 2017.

[88] J. Hughes, U. Culha, F. Giardina, F. Günther, A. Rosendo, and F. Iida. Soft Manipulators and Grippers: A Review. *Frontiers in Robotics and AI*, 2016, 2016.

[89] A. Humayun, F. Li, and J. M. Rehg. RIGOR: Reusing Inference in Graph Cuts for Generating Object Regions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 336–343. IEEE Computer Society, 2014.

[90] A. Humayun, F. Li, and J. M. Rehg. The Middle Child Problem: Revisiting Parametric Min-Cut and Seeds for Object Proposals. In *International Conference on Computer Vision (ICCV)*, pages 1600–1608. IEEE Computer Society, 2015.

[91] A. Humayun, O. Mac Aodha, and G. J. Brostow. Learning to find occlusion regions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2168. IEEE Computer Society, 2011.

[92] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *International Conference on Multimedia*, MM'14, pages 675–678. ACM, 2014.

[93] E. Johns, S. Leutenegger, and A. J. Davison. Deep Learning a Grasp Function for Grasping under Gripper Pose Uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4461–4468. IEEE, 2016.

[94] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *International Conference on Computer Vision (ICCV)*, pages 1530–1538. IEEE Computer Society, 2017.

[95] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[96] P. Krähenbühl and V. Koltun. Geodesic Object Proposals. In *European Conference on Computer Vision (ECCV) Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 725–739. Springer, 2014.

[97] P. Krähenbühl and V. Koltun. Learning to propose objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1574–1582. IEEE Computer Society, 2015.

[98] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012.

[99] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[100] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989.

[101] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[102] J. Lee, S. Kang, and S. Park. 3D Pose Estimation of Bin Picking Object using Deep Learning and 3D Matching. In *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 328–334, 2018.

[103] I. Lenz, H. Lee, and A. Saxena. Deep Learning for Detecting Robotic Grasps. In *Robotics: Science and Systems (RSS)*, 2013.

[104] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1119–1127. IEEE Computer Society, 2015.

[105] G. Li, Y. Xie, L. Lin, and Y. Yu. Instance-Level Salient Object Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 247–256. IEEE Computer Society, 2017.

[106] K. Li and J. Malik. Amodal Instance Segmentation. In *European Conference on Computer Vision (ECCV) Part II*, volume 9906 of *Lecture Notes in Computer Science*, pages 677–693. Springer, 2016.

[107] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch Tokens: A Learned Mid-level Representation for Contour and Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3158–3165. IEEE Computer Society, 2013.

[108] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. In *International Conference on Computer Vision (ICCV)*, pages 2999–3007. IEEE Computer Society, 2017.

[109] 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 *European Conference on Computer Vision (ECCV) Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[110] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy. Progressive Neural Architecture Search. In *European Conference on Computer Vision (ECCV) Part I*, Lecture Notes in Computer Science, pages 19–35. Springer, 2018.

[111] F. Liu, C. Shen, G. Lin, and I. D. Reid. Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. *IEEE Transactions on Pattern Analysis Machine Intelligence (TPAMI)*, 38(10):2024–2039, 2016.

[112] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path Aggregation Network for Instance Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768. IEEE Computer Society, 2018.

[113] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision (ECCV) Part I*, Lecture Notes in Computer Science, pages 21–37. Springer, 2016.

[114] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai. Richer Convolutional Features for Edge Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5872—5881. IEEE Computer Society, 2017.

[115] Y. Liu, S. Yang, B. Li, W. Zhou, J. Xu, H. Li, and Y. Lu. Affinity Derivation and Graph Merge for Instance Segmentation. In *European Conference on Computer Vision (ECCV) Part III*, volume 11207 of *Lecture Notes in Computer Science*, pages 708–724. Springer, 2018.

[116] C. Lu, S. Liu, J. Jia, and C. Tang. Contour Box: Rejecting Object Proposals without Explicit Closed Contours. In *International Conference on Computer Vision (ICCV)*, pages 2021–2029. IEEE Computer Society, 2015.

[117] P. Luo, G. Wang, L. Lin, and X. Wang. Deep Dual Learning for Semantic Image Segmentation. In *International Conference on Computer Vision (ICCV)*, pages 2737–2745. IEEE Computer Society, 2017.

[118] J. Mahler and K. Goldberg. Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences. In *Conference on Robot Learning (CoRL)*, pages 515–524, 2017.

[119] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In *Robotics: Science and Systems (RSS)*, 2017.

[120] J. Mahler, M. Matl, X. Liu, A. Li, D. V. Gealy, and K. Goldberg. Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning. In *International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[121] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. J. Kuffner, and K. Goldberg. Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards. In *International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016.

[122] M. Maire. Simultaneous Segmentation and Figure/Ground Organization Using Angular Embedding. In *European Conference on Computer Vision (ECCV) Part II*, Lecture Notes in Computer Science, pages 450–464. Springer, 2010.

[123] M. Maire, T. Narihira, and S. X. Yu. Affinity CNN: Learning Pixel-Centric Pairwise Relations for Figure/Ground Embedding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 174–182. IEEE Computer Society, 2016.

[124] S. Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A*, 374, 2015.

[125] K.-K. Maninis, J. Pont-Tuset, P. A. Arbeláez, and L. J. V. Gool. Convolutional Oriented Boundaries. In *European Conference on Computer Vision (ECCV) Part I*, volume 9905 of *Lecture Notes in Computer Science*, pages 580–596. Springer, 2016.

[126] K.-K. Maninis, J. Pont-Tuset, P. A. Arbeláez, and L. V. Gool. Deep Retinal Image Understanding. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI) Part II*, volume 9901 of *Lecture Notes in Computer Science*, pages 140–148. Springer, 2016.

[127] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *International Conference on Computer Vision (ICCV)*, pages 416–423. IEEE Computer Society, 2001.

[128] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. *IEEE Transactions on Pattern Analysis Machine Intelligence (TPAMI)*, 26(5):530–549, 2004.

[129] Z. Marton, F. Balint-Benczedi, Ó. M. Mozos, N. Blodow, A. Kanezaki, L. C. Goron, D. Pangercic, and M. Beetz. Part-Based Geometric Categorization and Object Reconstruction in Cluttered Table-Top Scenes. *Journal of Intelligent and Robotic Systems*, 76(1):35–56, 2014.

[130] M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26), 2019.

[131] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation? In *International Conference on Computer Vision (ICCV)*, pages 2697–2706. IEEE Computer Society, 2017.

[132] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[133] A. Mosinska, P. Márquez-Neila, M. Kozinski, and P. Fua. Beyond the Pixel-Wise Loss for Topology-Aware Delineation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3136–3145. IEEE Computer Society, 2018.

[134] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Detecting object affordances with Convolutional Neural Networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2765–2770. IEEE, 2016.

[135] A. M. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE Computer Society, 2015.

[136] M. Nieuwenhuisen, D. Droeschel, D. Holz, J. Stückler, A. Berner, J. Li, R. Klein, and S. Behnke. Mobile bin picking with an anthropomorphic service robot. In *International Conference on Robotics and Automation (ICRA)*, pages 2327–2334. IEEE, 2013.

[137] D. Novotný, S. Albanie, D. Larlus, and A. Vedaldi. Semiconvolutional Operators for Instance Segmentation. In *European Conference on Computer Vision (ECCV) Part I*, volume 11205 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 2018.

[138] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and Checkerboard Artifacts. *Distill*, 2016.

[139] A. E. Orhan and X. Pitkow. Skip Connections Eliminate Singularities. In *International Conference on Learning Representations (ICLR)*, 2018.

[140] G. Papandreou, T. Zhu, L. Chen, S. Gidaris, J. Tompson, and K. Murphy. PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model. In *European Conference on Computer Vision (ECCV) Part XIV*, Lecture Notes in Computer Science, pages 282–299. Springer, 2018.

[141] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2027–2034. IEEE Computer Society, 2013.

[142] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems Workshops (NIPSW)*, 2017.

[143] P. H. O. Pinheiro, R. Collobert, and P. Dollár. Learning to Segment Object Candidates. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1990–1998, 2015.

[144] P. H. O. Pinheiro, T. Y. Lin, R. Collobert, and P. Dollar. Learning to Refine Object Segments. In *European Conference on Computer Vision (ECCV) Part I*, volume 9905 of *Lecture Notes in Computer Science*, pages 75–91. Springer, 2016.

[145] P. O. Pinheiro. Unsupervised Domain Adaptation With Similarity Learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8004–8013. IEEE Computer Society, 2018.

[146] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marqués, and J. Malik. Multiscale Combinatorial Grouping for Image Segmentation and Object Proposal Generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(1):128–140, 2017.

[147] J. Pont-Tuset and L. J. V. Gool. Boosting Object Proposals: From Pascal to COCO. In *International Conference on Computer Vision (ICCV)*, pages 1546–1554. IEEE Computer Society, 2015.

[148] E. Potapova, K. M. Varadarajan, A. Richtsfeld, M. Zillich, and M. Vincze. Attention-driven object detection and segmentation of cluttered table scenes using 2.5D symmetry. In *International Conference on Robotics and Automation (ICRA)*, pages 4946–4952. IEEE, 2014.

[149] A. Pretto, S. Tonello, and E. Menegatti. Flexible 3D localization of planar objects for industrial bin-picking with monocamera vision system. In *International Conference on Automation Science and Engineering (CASE)*, pages 168–175. IEEE, 2013.

[150] M. Rad and V. Lepetit. BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth. In *International Conference on Computer Vision (ICCV)*, pages 3848–3856. IEEE Computer Society, 2017.

[151] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE, 2015.

[152] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE Computer Society, 2016.

[153] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE Computer Society, 2017.

[154] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *Computing Research Repository (CoRR)*, abs/1804.02767, 2018.

[155] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[156] X. Ren, C. C. Fowlkes, and J. Malik. Figure/Ground Assignment in Natural Images. In *European Conference on Computer Vision (ECCV) Part II*, volume 3952 of *Lecture Notes in Computer Science*, pages 614–627. Springer, 2006.

[157] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, pages 234–241. Lecture Notes in Computer Science. Springer, 2015.

[158] G. Ros, L. Sellart, J. Materzynska, D. Vázquez, and A. M. López. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243. IEEE Computer Society, 2016.

[159] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[160] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3859–3869, 2017.

[161] A. Salvador, M. Bellver, M. Baradad, F. Marqués, J. Torres, and X. G. i Nieto. Recurrent Neural Networks for Semantic Instance Segmentation. In *European Conference on Computer Vision (ECCV) Women in Computer Vision Workshop (WiCV)*. Springer, 2018.

[162] A. Schmitz, U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini. Design, realization and sensorization of the dexterous iCub hand. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 186–191. IEEE, 2010.

[163] H. Sedghi, V. Gupta, and P. M. Long. The Singular Values of Convolutional Layers. In *International Conference on Learning Representations (ICLR)*, 2019.

[164] F. Seide and A. Agarwal. CNTK: Microsoft's Open-Source Deep-Learning Toolkit. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, page 2135. ACM, 2016.

[165] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3982–3991. IEEE Computer Society, 2015.

[166] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *European Conference on Computer Vision (ECCV) Part V*, volume 7576 of *Lecture Notes in Computer Science*, pages 746–760. Springer, 2012.

[167] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.

[168] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*. IEEE Computer Society, 2015.

[169] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

[170] A. Stein and M. Hebert. Local Detection of Occlusion Boundaries in Video. In *British Machine Vision Conference (BMVC)*, 2006.

[171] S. C. Stein, M. Schoeler, J. Papon, and F. Wörgötter. Object Partitioning Using Local Convexity. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 304–311. IEEE Computer Society, 2014.

[172] D. Sun, C. Liu, and H. Pfister. Local Layering for Joint Motion Estimation and Occlusion Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1098–1105. IEEE Computer Society, 2014.

[173] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1139–1147, 2013.

[174] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. IEEE Computer Society, 2015.

[175] D. Tang, H. Fu, and X. Cao. Topology Preserved Regular Superpixel. In *International Conference on Multimedia and Expo (ICME)*, pages 765–768. IEEE, 2012.

[176] A. ten Pas, M. Gualtieri, K. Saenko, and R. P. Jr. Grasp Pose Detection in Point Clouds. *International Journal on Robotics Research (IJRR)*, 36(13-14):1455–1473, 2017.

[177] C. L. Teo, C. Fermüller, and Y. Aloimonos. Fast 2D border ownership assignment. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5117–5125. IEEE Computer Society, 2015.

[178] T. Tieleman and G. Hinton. RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[179] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, W. Zaremba, and P. Abbeel. Domain Randomization and Generative Models for Robotic Grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018.

[180] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars. A Deeper Look at Dataset Bias. In *Domain Adaptation in Computer Vision Applications*, Advances in Computer Vision and Pattern Recognition, pages 37–55. Springer, 2017.

[181] A. Torralba and A. A. Efros. Unbiased Look at Dataset Bias. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528. IEEE Computer Society, 2011.

[182] A. Ückermann, R. Haschke, and H. J. Ritter. Real-time 3D segmentation of cluttered scenes for robot grasping. In *IEEE-RAS International Conference on Humanoid Robots*, pages 198–203, 2012.

[183] J. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision (IJCV)*, 104(2):154–171, 2013.

[184] K. Wada, S. Kitagawa, O. Kei, and M. Inaba. Instance Segmentation of Visible and Occluded Regions for Finding and Picking Target from a Pile of Objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2048–2055. IEEE, 2018.

[185] C. Wang, L. Zhao, S. Liang, L. Zhang, J. Jia, and Y. Wei. Object proposal by multi-branch hierarchical segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3873–3881. IEEE Computer Society, 2015.

[186] P. Wang and A. L. Yuille. DOC: Deep OCclusion Estimation from a Single Image. In *European Conference on Computer Vision (ECCV) Part I*, volume 9905 of *Lecture Notes in Computer Science*, pages 545–561. Springer, 2016.

[187] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for Generic Object Detection. In *International Conference on Computer Vision (ICCV)*, pages 17–24. IEEE Computer Society, 2013.

[188] Y. Wang, X. Zhao, and K. Huang. Deep Crisp Boundaries. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1724–1732. IEEE Computer Society, 2017.

[189] K. Weicheng, H. Bharath, and M. Jitendra. DeepBox: Learning Objectness with Convolutional Networks. In *International Conference on Computer Vision (ICCV)*, pages 2479–2487. IEEE Computer Society, 2015.

[190] O. Williams, M. Isard, and J. MacCormick. Estimating Disparity and Occlusions in Stereo Video Sequences. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 250–257. IEEE Computer Society, 2011.

[191] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995. IEEE Computer Society, 2017.

[192] S. Xie and Z. Tu. Holistically-Nested Edge Detection. In *International Conference on Computer Vision (ICCV)*, pages 1395–1403. IEEE Computer Society, 2015.

[193] J. Yang, B. L. Price, S. Cohen, H. Lee, and M.-H. Yang. Object Contour Detection with a Fully Convolutional Encoder-Decoder Network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 193–202. IEEE Computer Society, 2016.

[194] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014.

[195] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[196] Z. Yu, C. Feng, M. Liu, and S. Ramalingam. CASENet: Deep Category-Aware Semantic Edge Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770. IEEE Computer Society, 2017.

[197] Z. Yu, W. Liu, Y. Zou, C. Feng, S. Ramalingam, B. V. K. V. Kumar, and J. Kautz. Simultaneous Edge Alignment and Learning. In *European Conference on Computer Vision (ECCV) Part III*, volume 11207 of *Lecture Notes in Computer Science*, pages 400–417. Springer, 2018.

[198] S. Zagoruyko, A. Lerer, T. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár. A MultiPath Network for Object Detection. In *British Machine Vision Conference (BMVC)*, 2016.

[199] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *Computing Research Repository (CoRR)*, abs/1212.5701, 2012.

[200] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision (ECCV) Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer, 2014.

[201] A. Zeng, S. Song, K. Yu, E. Donlon, F. R. Hogan, M. Bauzá, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. A. Funkhouser, and A. Rodriguez. Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching. In *International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[202] H. Zhang, P. Long, D. Zhou, Z. Qian, Z. Wang, W. Wan, D. Manocha, C. Park, T. Hu, C. Cao, Y. Chen, M. Chow, and J. Pan. DoraPicker: An autonomous picking system for general objects. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 721–726, 2016.

[203] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis Machine Intelligence (TPAMI)*, 22(11):1330–1334, 2000.

[204] F. Zhao, J. Zhao, S. Yan, and J. Feng. Dynamic Conditional Networks for Few-Shot Learning. In *European Conference on Computer Vision (ECCV) Part XV*, volume 11219 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2018.

[205] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng. Fully Convolutional Grasp Detection Network with Oriented Anchor Box. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7223–7230. IEEE, 2018.

[206] Y. Zhu, Y. Tian, D. N. Metaxas, and P. Dollár. Semantic Amodal Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3001–3009. IEEE Computer Society, 2017.

[207] C. L. Zitnick and P. Dollár. Edge Boxes: Locating Object Proposals from Edges. In *European Conference on Computer Vision (ECCV) Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 391–405. Springer, 2014.

[208] C. L. Zitnick and T. Kanade. A Cooperative Algorithm for Stereo Matching and Occlusion Detection. *IEEE Transactions on Pattern Analysis Machine Intelligence (TPAMI)*, 22(7):675–684, 2000.

[209] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710. IEEE Computer Society, 2018.

Appendix A

Deep Convolutional Networks

A.1 Kullback-Leibler Divergence

Let $p(\mathbf{x})$ be the probability distribution represented by the training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ and $\hat{p}(\mathbf{x}_n)$ the distribution of the approximated set $\{(\mathbf{x}_n, \hat{\mathbf{y}}_n)\}_{n=1}^{N}$. **Minimizing the objective term of Equation 2.4 is equivalent to minimizing the Kullback-Leibler divergence $KL(p, \hat{p})$ between the training distribution $p(\mathbf{x})$ and the distribution $\hat{p}(\mathbf{x})$ learned by the network $f_{\mathbf{W}}$.** The Kullback-Leibler divergence, which quantifies how much information is lost when approximating a distribution by another one, is defined as the expectation of the log difference between the target and approximating distributions:

$$KL(p, \hat{p}) := E[\log p(\mathbf{x}) - \log \hat{p}(\mathbf{x})] \tag{A.1}$$

Equation A.1 can be rewritten as follows:

$$
\begin{aligned}
KL(p, \hat{p}) &= \sum_{n=1}^{N} p(\mathbf{x}_n)(\log p(\mathbf{x}_n) - \log \hat{p}(\mathbf{x}_n)) \\
&= \underbrace{\sum_{n=1}^{N} p(\mathbf{x}_n) \log p(\mathbf{x}_n)}_{\text{entropy of } p} \; \underbrace{- \sum_{n=1}^{N} p(\mathbf{x}_n) \log \hat{p}(\mathbf{x}_n)}_{\text{cross-entropy between } p \text{ and } \hat{p}}
\end{aligned}
\tag{A.2}
$$

Equation A.2 shows that $KL(p, \hat{p})$ is actually the difference between the entropy of p and the cross-entropy between p and $\hat{p}$. Assuming that $\{\mathbf{x}_n\}_{n=1}^{N}$ are **independent observations**, the **entropy term is constant** and the **cross-entropy term can be equivalently minimized as follows**:

$$\min_{\mathbf{W}} p(\mathbf{x}_n) \log \hat{p}(\mathbf{x}_n) = \min_{\mathbf{W}} \sum_{i=1}^{C} (\mathbf{y}_n)_i \log(\hat{\mathbf{y}}_n)_i \tag{A.3}$$

I

As a result, we find the objective term of Equation 2.4:

$$\arg\min_{\mathbf{W}} KL(p, \hat{p}) = \arg\min_{\mathbf{W}} - \sum_{n=1}^{N} \sum_{i=1}^{C} (\mathbf{y}_n)_i \log(\hat{\mathbf{y}}_n)_i \qquad \text{(A.4)}$$

A.2 Bayesian Interpretation

From a Bayesian perspective, the network inference $\hat{z} = f_{\mathbf{W}}(\mathbf{x}) \in [0, 1]$ from Equation 2.5 can be viewed as a Maximum A Posteriori (MAP) probability estimate with a Gaussian prior on the weights.

Proposition 1. *If $\mathbf{Z}$ is a random variable that follows a Bernouilli distribution with parameter $\hat{z} = f_{\mathbf{W}}(\mathbf{x}) \in [0, 1]$, then*

$$\mathbf{W}^{\star} = \mathbf{W}_{MAP} := \arg\max_{\mathbf{W}} \ p(\mathbf{Z}|\mathbf{W})p(\mathbf{W}) \qquad \text{(A.5)}$$

Proof. Assuming independent observations, and given that logarithm is monotonic, $\mathbf{W}_{MAP}$ can be rewritten as follows:

$$\begin{aligned}
\mathbf{W}_{MAP} &= \arg\max_{\mathbf{W}} \ \log\left(p(\mathbf{Z}|\mathbf{W})p(\mathbf{W})\right) \\
&= \arg\max_{\mathbf{W}} \ \log\left(\prod_{n=1}^{N} p(z_n|\mathbf{W})p(\mathbf{W})\right) \\
&= \arg\max_{\mathbf{W}} \ \sum_{n=1}^{N} \log\left(p(z_n|\mathbf{W})p(\mathbf{W})\right) \\
&= \arg\max_{\mathbf{W}} \ \left(\sum_{n=1}^{N} \log p(z_n|\mathbf{W}) + N \log p(\mathbf{W})\right)
\end{aligned} \qquad \text{(A.6)}$$

As $\mathbf{Z}$ follows a Bernouilli distribution of parameter $\hat{z} = f_{\mathbf{W}}(\mathbf{x})$,

$$p(\mathbf{Z} = z|\mathbf{W}) = \hat{z}^z (1 - \hat{z})^{(1-z)} \qquad \text{(A.7)}$$

As $\mathbf{W}$ follows a Gaussian distribution, there exists $\sigma \in \mathbb{R}$ such that

$$p(\mathbf{W}) \propto \exp\left(-\frac{\|\mathbf{W}\|_2^2}{\sigma^2}\right) \qquad \text{(A.8)}$$

As a result,

$$\begin{aligned}
\log p(z_n|\mathbf{W}) &= z_n \log(\hat{z}_n) + (1 - z_n) \log(1 - \hat{z}_n) \\
\log p(\mathbf{W}) &\propto \|\mathbf{W}\|_2^2
\end{aligned} \qquad \text{(A.9)}$$

Considering that $\arg\min(-\cdot) = \arg\max(\cdot)$, we finally obtain

$$\mathbf{W}_{MAP} = \mathbf{W}^{\star} \qquad \text{(A.10)}$$

$\square$

Appendix B

Occlusion-Aware Instance Segmentation

B.1 Binarization Thresholds

In our pipeline for generating an ordered set of candidates ($c.f.$ Section 4.4), we introduced two hyperparameters α and β, that are thresholds for binarizing the boundary and occlusion maps respectively.

As shown by Figure B.1, these thresholds impact the generation of an affordable instance from the bicameral network inference. High threshold values tend to break boundaries, thus favouring overclustering. Low threshold values induce more closed boundaries and false positive in the binary maps, thus favouring oversegmentation. Favouring overclustering makes blinder as less instance candidates are produced and instance candidates tend to be grouped instances or instances grouped with the background. Conversely, favouring oversegmentation produces more instance candidates but these candidates tend to be instance parts.

In our bin-picking experiments, we set these thresholds to .1 as, in industrial bin-picking applications, we want to maximize the probability to succesfully grasp an instance. In the worst-case scenarios, $i.e.$ only instance parts as instance candidates or no instance candidates, we prefer attempting to grasp inside an instance even if the grasp is not optimal, rather than doing nothing. As the network inference is most likely correct or polluted with false positive, we set a low threshold value for both α and β. We thus avoid to break boundaries and face mostly two pick cases: either a correctly delineated non-occluded instance, or an instance part.
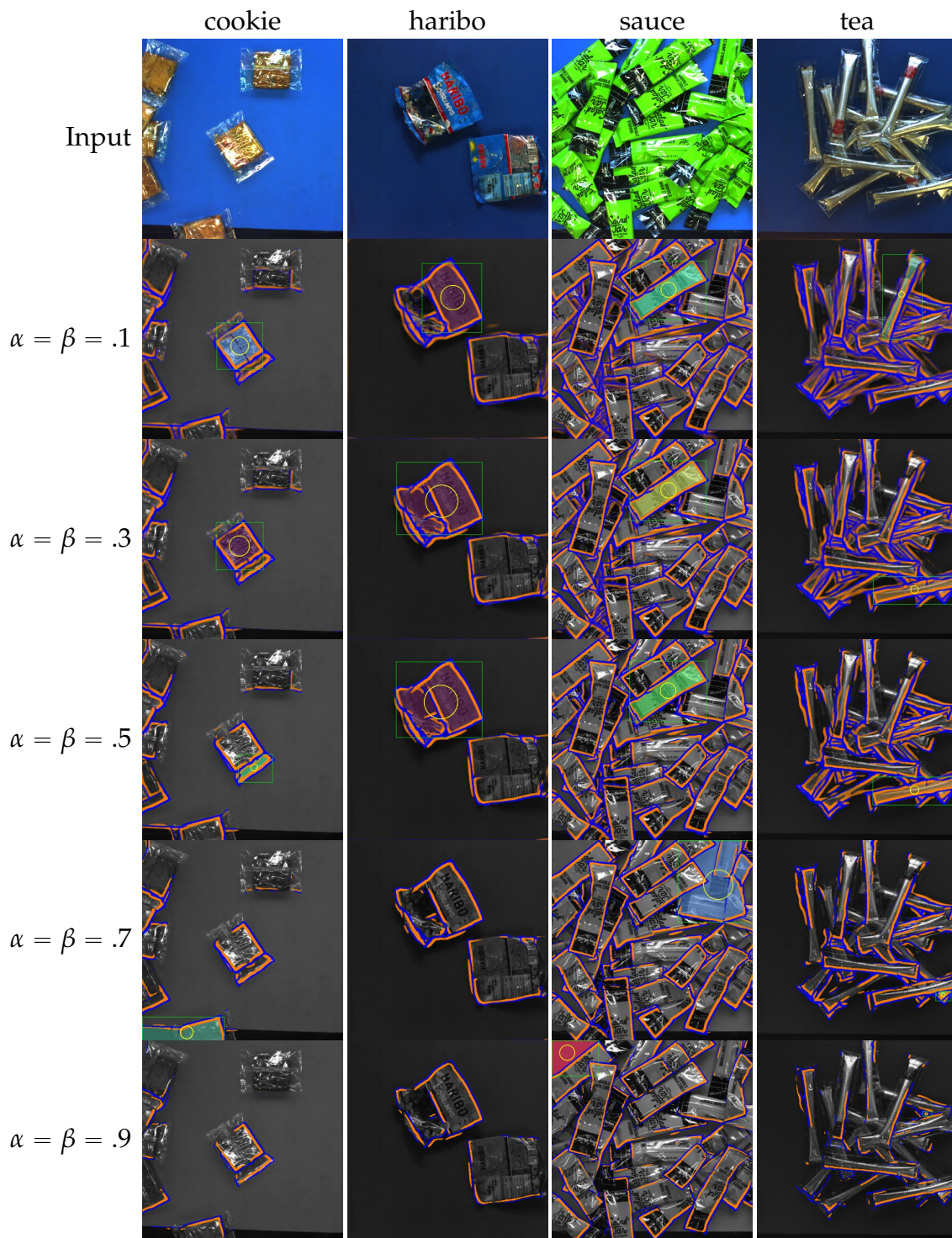
Figure B.1: Comparative results using a bicameral network trained on Mikado+ and different thresholds α and β for binarizing the network inference for boundaries and occlusions respectively. A high threshold value tends to break boundaries, thus favouring overclustering. A low threshold value induces more closed boundaries and false positive, thus favouring oversegmentation.

Appendix C

Application to Bin-Picking

C.1 Synthetic Training Data Generation

C.1.1 Textures and Backgrounds

The Mikado synthetic dataset was generated using 120 texture images respectively. The number of texture and background images was augmented to 2,400 and 600 respectively, for its extension Mikado+. While the texture images in Mikado are mainly for stick-like sachets, the Mikado+ textures include other food shapes as well: pastries, cookies, square sachets, rigid boxes. All the texture and background images were retrieved using the Google Image search engine and manually cropped to remove any background. A comprehensive overview of the texture and background images is provided in Figures C.1 and C.2 respectively.

C.1.2 Alternative Input Modality

The Mikado pipeline for generating synthetic training data could be adapted for an alternative input modality, such as depth. For example, instead of texture images, we collect a set of bottle models from ShapeNet [32] (*c.f.* Figure C.3). We then render two different views of the scene to model a binocular system. Using an off-the-shelf stereo-matching algorithm for 3D reconstruction, we generate synthetic top-view depth images of bottle instances piled up in bulk, with intra-class geometric variability. As qualitatively shown by Figure C.4, such images can employed to synthetically train a bicameral network for jointly inferring boundaries and occlusions from a single depth image.

Mikado
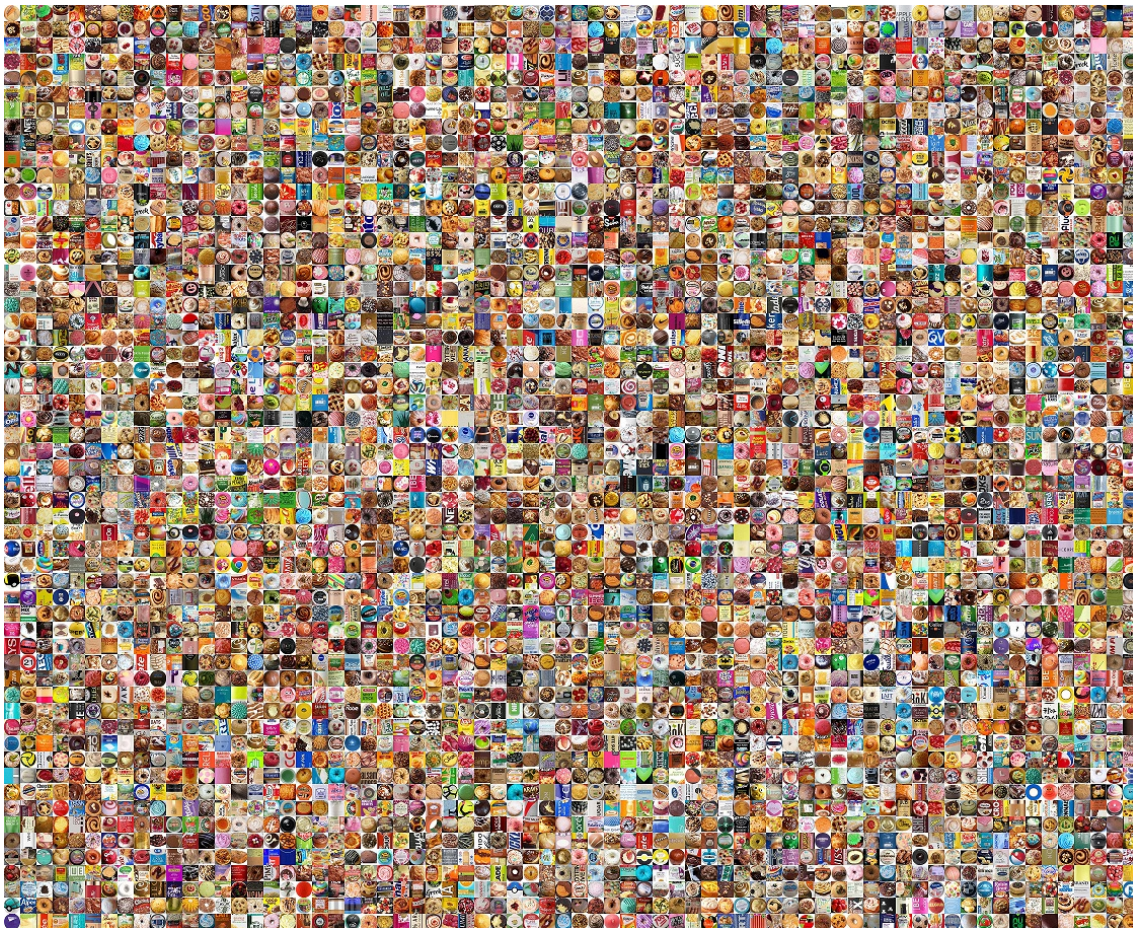
Mikado+

Figure C.1: Overview of the textures used to generate Mikado (top) and Mikado+ (bottom). Best viewed in electronic version

Mikado

Mikado+

Figure C.2: Overview of the backgrounds used to generate Mikado (top) and Mikado+ (bottom). Best viewed in electronic version
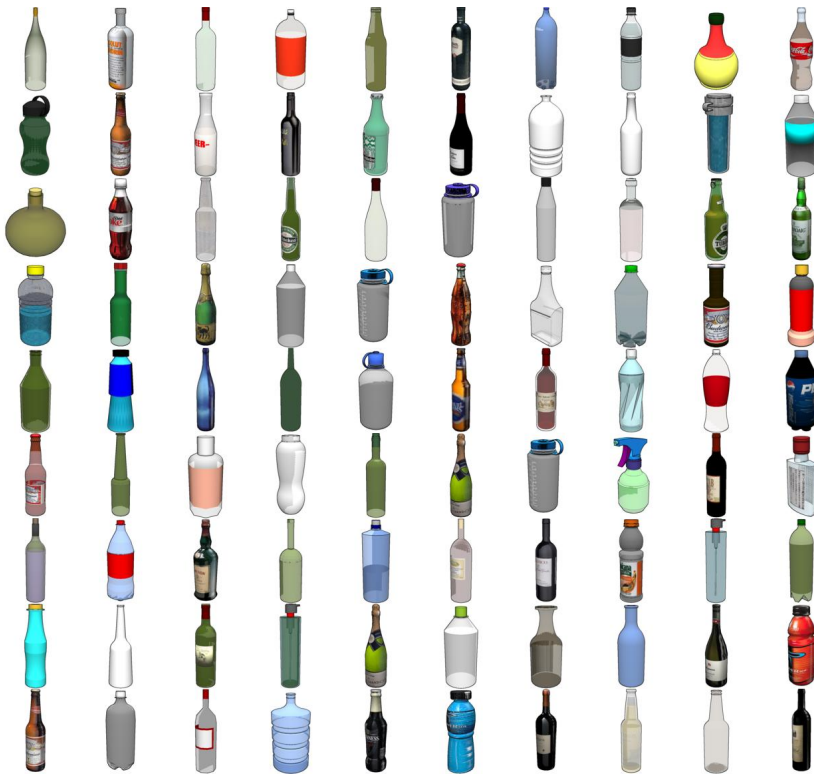
Figure C.3: The bottle models, from ShapeNet [32], used to generate synthetic depth maps of bottle instances piled up in bulk, within the proposed Mikado framework
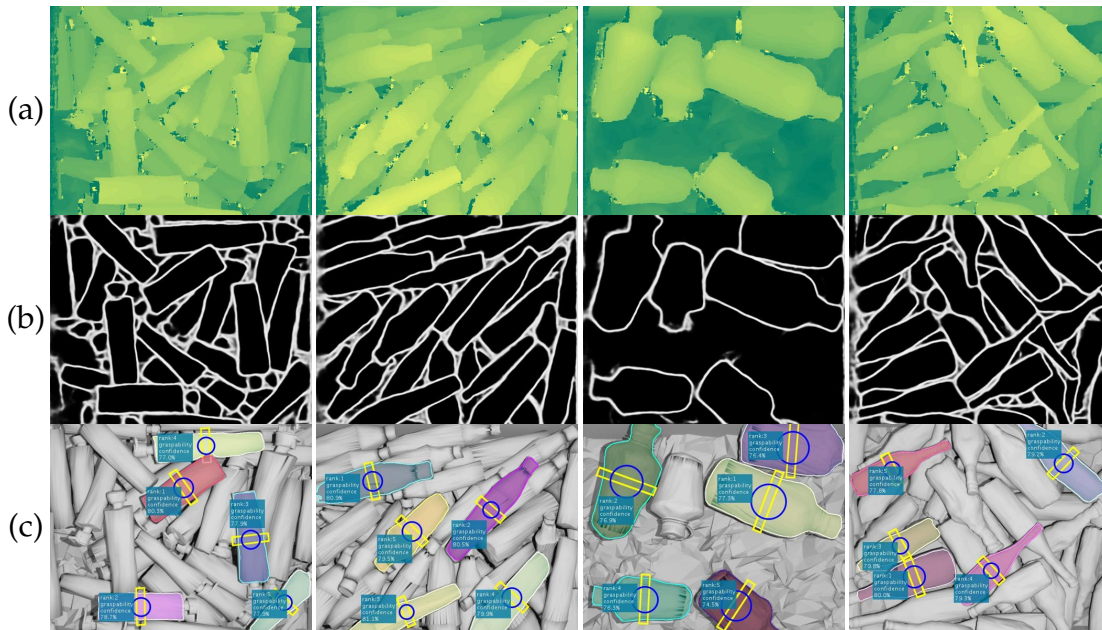

Figure C.4: Qualitative results on synthetic depth images using a synthetically trained bicameral network for jointly detecting boundaries and occlusions from a single depth map. From top to bottom: input (a); detected boundaries (b); result (c). Circles and rectangles represent vacuum suction and parallel-jaw instance-centered grasps. This shows that the proposed bicameral network and Mikado pipeline could be easily adapted to alternative input modality.

C.2   Real-World Experimental Evaluation

This section contains the numerical version of the bin-picking performances presented in Section 5.3 (see Table C.1 for the correspondences). The tables also include the number of observations and bin-picking sequences on which the performances are averaged (*c.f.* Table C.2).

Figure	Table
5.13	C.3
5.16	C.4
5.20	C.5
5.23	C.6

Table C.1: Correspondances between the figures in Section 5.3 and the tables in this section

	Metric	Measures the amount of
N_O	Observations	Number of observations
N_S	Sequences	Number of bin-picking sequences
S_R	Real success	Successful extractions, *i.e.* whose instance is taken away
M_V	Virtual margin	Failed extractions of well detected instances
S_V	Virtual success	Extractions either successful, or failed but whose grasp is centered on a non-occluded instance
S_C	Centered success	Successful extractions whose grasp is centered
S_{NO}	Non-occluded success	Successful extractions whose instance is unoccluded
F_{NC}	Not-centered failure	Failed extractions because of not centered grasps
F_O	Occluded failure	Failed extractions because of occluded instances

Table C.2: Definition of our real-world performance metrics

		N_O	N_S	S_R	M_V	S_V	S_C	S_{NO}	F_{NC}	F_O
compote		75	8	.74	.42	.85	.96	.89	.37	.32
cookie		107	4	.81	.00	.81	.92	.78	.60	.55
crepe		90	7	.74	.09	.77	.88	.84	.77	.23
donut		102	7	.60	.24	.70	.92	.97	.42	.44
haribo		91	5	.52	.68	.85	1.0	.91	.14	.21
madeleine		126	5	.65	.07	.67	.88	.90	.75	.25
nem		111	4	.84	.17	.86	.92	.78	.78	.28
sauce		417	19	.77	.25	.83	.95	.84	.50	.51
tartelette		97	5	.84	.13	.86	.86	.79	.73	.40
tea		140	8	.77	.41	.86	.78	.87	.50	.16
overall		1,356	72	.74	.27	.81	.91	.85	.51	.36

Table C.3: Per-product performances averaged over all observations, using a bicameral network trained on Mikado+

		N_O	N_S	S_R	M_V	S_V	S_C	S_{NO}	F_{NC}	F_O
nem	Mikado	144	4	.50	.07	.54	.85	.78	.92	.35
	Mikado+	111	4	.84	.17	.86	.92	.78	.78	.28
sauce	Mikado	90	4	.75	.05	.76	.88	.86	.77	.50
	Mikado+	417	19	.77	.25	.83	.95	.84	.50	.51
tea	Mikado	145	7	.64	.36	.77	.58	.79	.62	.16
	Mikado+	140	8	.77	.41	.86	.78	.87	.50	.16

Table C.4: Per-product performances averaged over all observations, using a bicameral network trained on Mikado or Mikado+

		N_O	N_S	S_R	M_V	S_V	S_C	S_{NO}	F_{NC}	F_O
madeleine	Kamido	129	2	.50	.11	.55	.55	.77	.75	.38
	Ours (Mikado+)	111	4	.84	.17	.86	.92	.78	.78	.29
sauce	Kamido	108	2	.49	.12	.55	.37	.69	.87	.35
	Ours (Mikado+)	417	19	.77	.25	.83	.95	.84	.50	.51

Table C.5: Comparison of the real-world performances with the current industrial approach

		N_O	N_S	S_R	M_V	S_V	S_C	S_{NO}	F_{NC}	F_O
nem	F25	122	4	.80	.32	.86	.95	.76	.48	.28
	F100	111	4	.84	.17	.86	.92	.78	.78	.28
sauce	F25	109	4	.71	.03	.72	.95	.86	.53	.84
	F100	417	19	.77	.25	.83	.95	.84	.50	.51
cookie	F25	107	4	.84	.12	.86	.92	.70	.59	.65
	F100	107	4	.81	.00	.81	.92	.78	.60	.55
madeleine	F25	129	4	.68	.15	.73	.86	.94	.54	.49
	F100	126	5	.65	.07	.67	.88	.90	.75	.25

Table C.6: Per-product performances, using bicameral networks trained on Mikado+, with respect to the number of convolutional filters

AUTORISATION DE SOUTENANCE

Vu les dispositions de l'arrêté du 25 mai 2016,

Vu la demande du directeur de thèse

Monsieur L. CHEN

et les rapports de

M. T. CHATEAU
Professeur - Directeur de l'Ecole Doctorale des Sciences - Université Clermont-Auvergne -
Institut Pascal - Campus Universitaire des Cézeaux - 7 avenue Blaise Pascal
Bâtiment Physique 3 - TSA 60026 / CS 60026 - 63178 Aubière cedex

et de

M. D. SAMARAS
Professeur - Directeur du Computer Vison Lab - Stony Brook University
Computer Science Department - 263 New Computer Science - Stony Brook - NY 11794-2424
USA

Monsieur GRARD Matthieu

est autorisé à soutenir une thèse pour l'obtention du grade de **DOCTEUR**

Ecole doctorale InfoMaths

Fait à Ecully, le 14 mai 2019

P/Le directeur de l'E.C.L.
La directrice des Etudes

Le Directeur
de l'Ecole centrale de Lyon
Frank DEBOUCK

M-A. GALLAND