PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

Learning representations for visually-guided robotics

Soutenue par
Robin Strudel
Le 1er Février 2023

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Vincent Lepetit ENPC ParisTech	*Président du jury* *Examinateur*
Nicolas Mansard LAAS, CNRS	*Rapporteur*
Patrick Pérez Valeo	*Rapporteur*
Ivan Laptev Inria	*Directeur de thèse*
Cordelia Schmid Inria	*Directeur de thèse*

ENS | PSL RESEARCH UNIVERSITY PARIS | Inria

Remerciements

There are so many people that helped me in one way or another and to whom I am deeply grateful. While it would be impossible to mention all of you here, let me give it a try and thank some of you.

I would like to start by warmly thanking my PhD advisors Ivan Laptev and Cordelia Schmid, it was great to start our robotics journey together. I was new to experimental science, and you showed me how to carve out interesting insights out of the mess of raw experiments. I liked the dedication and determination to make progress on difficult problems. I learned a lot from your enthusiasm, patience and willingness to push hard, even when I felt the odds were stacked against us. You pushed me to do my best, sometimes until my limits. I am grateful for the precious time Nicolas Mansard and Patrick Perez took to review my thesis, and Vincent Lepetit for being president of the jury.

These almost five years have been particularly enjoyable thanks to the members of Willow and Sierra teams and the atmosphere of the 4th floor of INRIA Paris. The coffee breaks, la cantine, going for a sandwich at la bonne trad' or a beer at Ground Control, climbing at Arkose Nation and doing retreats at Marseille and Avignon were great moments I was glad to share with you. A special thanks to Loïc Estève for showing me so many tricks, the open-source universe and for the climbing breakaways. While PhD may feel like a lonely path sometimes I was very lucky to share this trip with Alexander Pashevich and Ricardo Garcia. Thank you for welcoming me at Thoth in Grenoble, for the shared excitement and frustrations, for the late-night discussions and for the moments spent together. Aligato Thomas Chabal for our trip to Japan and this memorable karaoke.

I am grateful to have had the chance to spend Summer 2022 at DeepMind Paris with an amazing team and my hosts Rémi Leblond and Laurent Sifre. I am looking forward to opening a new chapter with the team and build cool things together.

J'ai une pensée pour les enseignants qui m'ont inspiré et transmis leur enthousiasme avec tant d'énergie comme Olivier Derquene, Jean Giorno, Nicolas Tozel, Antoine Touzé, Julien Berestycki et Alyosha Efros. Merci d'avoir nourri ma curiosité toutes

ces années. Parmis les expériences qui me marqueront à vie je tiens à remercier Patrice Merrier, ce que tu nous a partagé va bien au delà du volley-ball.

Par les mots qui suivent je tiens à exprimer ma profonde gratitude à tous les gens qui me sont proches. La brièveté de ces propos n'a d'égal que le sentiment que je vous porte.

À vous les copains, les cernois, la coloc de Massy, merci pour tout. Dans un autre monde j'aurais nommé ce manuscrit *Wall-A* ou *Eye-robot*. Que les années devant nous soient all-star danger. Merci pour l'inspiration, merci pour la complicité.

À vous mes parents Jean-Loup et Rita, à vous mes frères Vivien, Alexis, Guillaume et Chris. À vous les Amiel, Freddy, Olivier et Thibauld. À toi la sauvage Mimi. À toi ma belle Chloé, petit volcan à l'énergie débordante, en éruption quasi-permanente. Merci pour votre soutien, merci pour votre amour.

Résumé

Le but de cette thèse est de développer des modèles, représentations et méthodes d'apprentissage pour l'acquisition automatique de compétences robotiques guidées par la vision à partir de démonstrations, ainsi que la localisation d'objets.

Nous présentons tout d'abord une méthode pour l'acquisition de compétences robotiques à partir de démonstrations. Un vocabulaire de compétences élémentaires est appris avec de l'apprentissage de politiques par imitation. Les compétences sont ensuite combinées avec une politique de planification apprise par renforcement afin de réaliser des taches plus complexes. Nous montrons sur plusieurs taches que la politique entrainée en simulation transfère sur un robot réel. Pour ce faire, nous avons développé une méthode qui optimise des séquences d'augmentations de données synthétiques afin de résoudre une tache auxiliaire de localisation d'objets sur des données réelles.

Nous proposons ensuite une méthode de planification de mouvements à partir de capteurs. Notre méthode exploite la connaissance des obstacles environnants pour accélérer la recherche de chemins sans collision. La représentation apprise généralise sur une grande variété d'obstacles et la politique de planification fonctionne sur de nouveaux environnements avec des obstacles se déplaçant de manière dynamique.

Alors que les politiques guidées par la vision apprennent des représentations visuelles à partir du contrôle, une autre approche consiste à apprendre des représentations visuelles centrées sur les objets à manipuler. Une fois que la localisation d'un objet est estimée, elle est ensuite intégrée à des contrôleurs robotiques classiques. Les représentations centrées sur les objets reposent sur des méthodes de segmentation que nous proposons d'améliorer avec les contributions suivantes.

Nous introduisons une méthode de segmentation sémantique basée sur les transformers qui exploite l'information globale contenue dans une image à toutes les couches du modèle. Nous obtenons des résultats état de l'art et montrons l'avantage de notre modèle comparé à des réseaux de convolution. Notre méthode de segmentation présente deux limitations, le modèle localise des objets qui sont prédéfinis et son entrainement nécessite des images annotées pour chaque pixel. Pour remédier à

ces limitations, nous présentons une méthode qui segmente des objets définis à partir d'une description texte et ne nécessite pas de supervision au niveau des pixels. Notre méthode apprend à localiser des objets en utilisant des annotations au niveau de l'image uniquement comme la présence ou l'absence d'un objet dans l'image.

Abstract

The goal of this thesis is to develop models, representations and learning algorithms for the automatic acquisition of visually-guided robotic skills from demonstrations and for object localization.

We first introduce a method to acquire robotic skills from demonstrations by learning a vocabulary of basic skills with behavioral cloning. Skills are then combined with a planning policy learned with reinforcement learning in order to perform more complex tasks. We show successful transfer of multiple tasks from simulation to a real robot by using a method developed in this thesis optimizing a sequence of data augmentations on synthetic data to solve a proxy object localization task on real data.

We then focus on sensor-based motion planning and propose an approach leveraging the knowledge of surrounding obstacles observed with a camera to accelerate the finding of collision-free paths. The learned representation generalizes across a large variety of objects, and the planning policy can handle new environments with dynamically moving obstacles.

While visually-guided policies learn task-centric image representations from control supervision, another line of work consists in learning object-centric representations that can be plugged into classical robotics methods. Object-centric approaches rely on a segmentation backbone for which we propose the following contributions.

Towards this goal we propose a transformer-based semantic segmentation model that leverages global context of the image at every stage of the model and show state-of-the-art results when compared to convolution-based approaches on classical benchmarks. Our segmentation model presents two limitations, it localizes a predefined set objects and requires dense annotations to be trained, which limits its scalability to large datasets. To address these limitations, we propose a method that segments an open set of visual concepts defined by natural language and does not require pixel-level supervision. Our method learns to localize objects by using image-level labels such as the presence of an object in the image.

Contents

Chapter 1

Introduction

Robotics development in the last 10 years led to more agile and dynamic robots capable of acrobatic figures such as humanoid robots performing a backflip or jumping between platforms. While their athletic performance is astonishing, it requires years of work from teams of skilled researchers and engineers. Current robots aslo exhibit limited sensory abilities that prevent their widespread use in open and unstructured environments. There is still a long way to go to develop robots more flexible and easier to use. We would like to develop intelligent systems that evolve autonomously by sensing and acting on the world around them, much like kids learning dexterous abilities by playing with objects or becoming excellent at walking by trying to mimick their parents. Sensing and acting produces a wealth of information about causes and effects and a core motivation of this work is to develop learning methods leveraging the interaction between what a robot can act on, and the resulting effects perceived through their visual sensors.

1.1 Goal

Our first objective in this thesis is to develop models, representations and algorithms for the automatic acquisition of complex robotic manipulation skills, requiring perception, from demonstrations and interactions. Most tasks performed by humans involve sensing such as sight or touch when manipulating objects, performing a surgical operation or playing a Rubik's cube - a long term goal is to acquire similar skills with a robot as depicted in Figure 1.1. Our work focuses on simpler tasks, but we develop methods amenable to this long term goal, paving the way towards acquiring complex perception-guided skills.

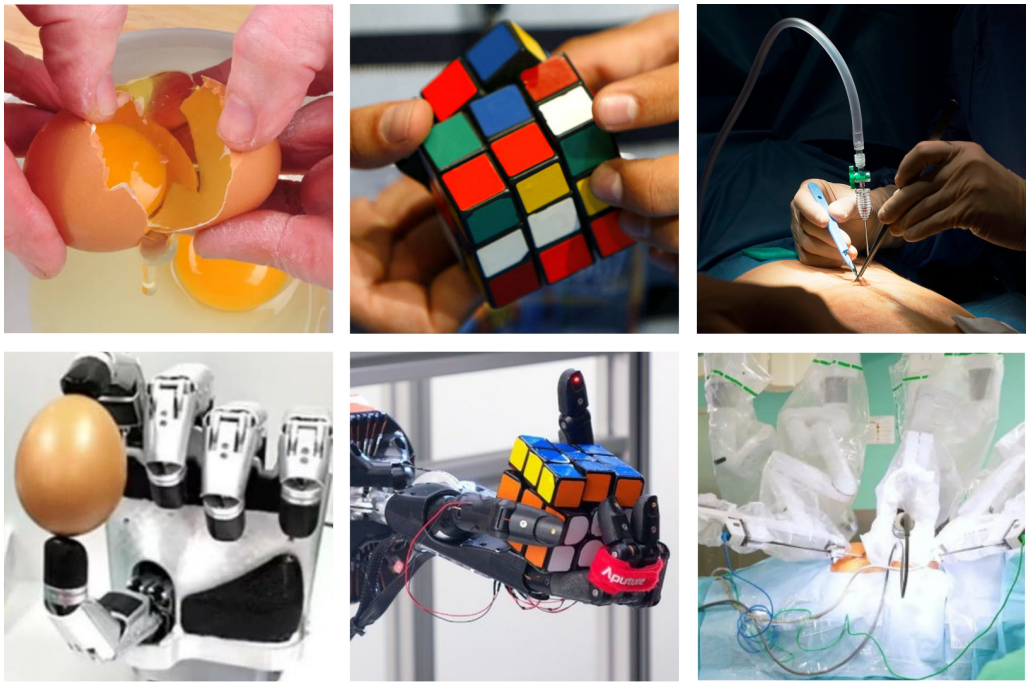


Figure 1.1: Example of complex manipulation tasks such as breaking an egg, playing with a Rubik's cube or performing a surgical operation, that can be performed by a person and are currently daunting to perform with a robot.

The first focus is thus to develop visually-guided policies that can be learned from demonstrations and robot interactions. In Chapter 3, we propose a method that learns a vocabulary of simple skills with behavioral cloning and a high level planning policy combining these skills to perform more complex tasks with reinforcement learning. We wish to avoid learning policies from scratch on a real robot and leverage highly parallelizable simulators. Along with policy learning in simulation, we design a sim-to-real method, presented in Chapter 4 to learn policies that can be deployed on a real robot without the need of costly collection of real data. We also propose a new method to perform visually-guided motion planning in Chapter 5 and show its advantages over standard methods.

Our second objective is related to object-centric methods as we aim at improving and scaling existing methods for object segmentation in images. Segmentation pipelines is the backbone of many visual based systems for robotics and autonomous driving, and improving their performance inherently provides more capable autonomous systems. We first propose to reformulate segmentation as a sequence-to-sequence problem and propose a new approach Segmenter in Chapter 6 leading to state-of-the art results. Segmentation models remain label intensive and require costly pixel-level annotations. To address this issue methods leveraging weaker form of supervision such as image-level labels have been developed in the past. We build on Segmenter

and propose TSEG in Chapter 7, a model extending previous weakly-supervised approach relying on a fixed set of predefined labels to datasets where objects are described with natural language. This allows to scale existing methods to larger datasets.

1.2 Motivation and challenges

1.2.1 Robotics

Figure 1.2: (Left) Robots performing specific tasks, such as wielding pieces of a car, in a constrained environment. Industrial robots are highly specialised and are designed to perform one or a few fixed tasks. Each task requires the development of a new controller by highly skilled engineers. (Right) A collaborative robot, helping a person crafting an object. To get there, we need general controllers using vision to solve a rich variety of tasks and react with changes in the environment.

Robots are seldom witnessed in open environments or interacting with humans as illustrated in Figure 1.2. They are widely used for industrial applications where they are excellent at performing specialized tasks in highly constrained environments. Factory robots are not autonomous, they precisely execute a predefined set of instructions without environment feedback. The lack of sensory feedback inhibits the autonomous abilities of robots. With a few exceptions such as vacuum cleaning robots, existing robots cannot evolve in unknown environments and adapt to surrounding changes or interact with people around them in order to help accomplish a task.

A core motivation of this PhD thesis is to develop visually grounded methods enabling robots to observe then interpret their environment to react according to its changes. A promising direction that we explore in this work is to learn visuo-motor policies directly mapping raw pixels to robot controls. We learn policies by leveraging human designed demonstrations of tasks or from experience, by trial and errors.

Robots that can see and evolve in an open world have a wide range applications, they can be used for collaborative tasks, tasks requiring high precision in a medical surgery room or in a factory, assist someone to perform tasks involving heavy objects, driver a car on Earth or a rover on Mars.

Figure 1.3: Precise manipulation tasks involving sensing and contact. These tasks are challenging because they require millimeter-precise controller with close-loop sensor feedback, a particularly daunting problem as we lack general approaches to design or learn such controllers.

Perception from sensors and partial observability. Learning robot control laws for manipulation directly from raw pixels is very difficult. The signal is high-dimensional, for example a small RGB image with resolution 224×224 contains 150.000 dimensions. Consider a manipulation task such as shown in Figure 1.3 Left where the goal is to empty a sink of dishes and place plates into a dishwasher. From raw pixels, the model has to infer the location of objects of interest in the presence of occlusion, varying illumination condition and with a large variety of objects. To be robust, the model should also ignore irrelevant features of the image corresponding to distractors, the background or sensor noise. Classical robotics approaches typically circumvent this problem by using motion capture systems or QR-code stuck on objects to obtain accurate, low-dimensional representation of objects. A core challenge of vision-based robotics is to develop methods to perform manipulation tasks while relying on simple sensors such as a camera filming the scene coupled with force and touch sensors on the robot end-effector as depicted in Figure 1.3. Current methods rely on heavy laboratory or industrial instrumentation required to control the environment while we would like to develop autonomous systems that can evolve in the wild. An interesting direction of research is to leverage recent progress in computer vision powered by deep learning to learn perception guided controllers incorporating vision and touch that can be deployed on a real robot.

Reasoning about causes and effects. Learning control laws from sensor measurements allows drawing direct sensorimotor connections between a robot and its environment. It produces a wealth of information about causes and effects, about the consequence of actions and what to do in order to perform a task. In this thesis, we explore two families of methods, namely imitation learning and reinforcement learning, whose goal is to develop computational approaches to learning from demonstrations and interaction. Combined with recent deep learning approaches, learning policies operating from sensor measurements is a promising direction that is computationally intensive, it either requires a large body of demonstrations or robot interaction to learn manipulation behaviors that remain limited. Learning a rich set of behaviors in a sample efficient way is a major challenge.

Leveraging simulation and sim-to-real gap. Learning visual representations requires large amount of training data, especially when learning from interaction. Policy learning methods are hard to deploy on a real robot because of safety issues, hardware failure and the speed of data collection, limited by real-time execution. On the other hand, physics simulators provide an attractive alternative due to simple parallelization allowing large scale data collection and access to world state information during training. However, it comes at a cost of a reality gap. It is both hard to synthesize realistic physical interactions and diverse and representative visual appearances. Due to these factors, policies trained in simulation provide low performance when deployed on a real system. A large body of research tries to either close the gap or randomize the parameters of simulation to learn robust policies in simulation transferring to the real world.

1.2.2 Image segmentation

Image segmentation is a challenging computer vision problem with a wide range of applications including autonomous driving, robotics, augmented reality, image editing and medical imaging [Siam, 2017; Hong, 2018; Hesamian, 2019]. Semantic segmentation consists in assigning semantic labels to each pixel of an image. Grouping pixels into meaningful visual entities enables the localization of objects, people, or robots. Segmentation allows to decompose a scene into its core elements and is a key component of many perception pipelines in autonomous driving or robotic manipulation.

Fixed-set segmentation. In the past decade, image segmentation system performances soared with the development of deep learning. The seminal work of DeepLab [Liang-Chieh, 2015] proposed to use fully convolutional networks (FCN) and showed

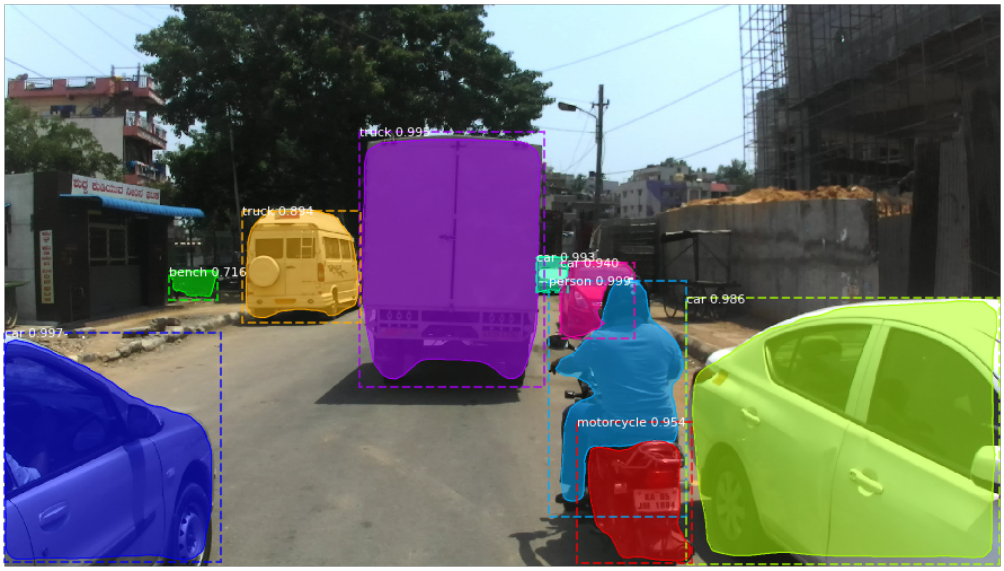


Figure 1.4: Image segmentation for autonomous driving. Masks of meaningful objects for driving such as a car, a motorcycle or a person are overlayed over the image. Segmentation is challenging due to large scene variability coming from change of illumination, viewpoint or weather conditions. Another source of variability comes from intra-class variations as models require to detecta rich sets of cars, motorcycles or road signs in different countries.

their efficiency for segmentation. A steady performance improvement then followed with the use of specialized layers to enhance convolution field-of-views and generate multi-scale features. Despite much effort, image segmentation remains a challenging problem due to rich intra-class variation, context variation and ambiguities originating from occlusion. Even though specialized models allow improvements on specific tasks, the development of generalist models for vision is a current challenge where important progress have occurred recently powered by transformer based models. To train general vision models, there is a need for general architectures but also for a large corpus of data. Manual image annotation of scene regions is extremely costly and classical benchmarks typically contain relatively small number of images and restricted sets of predefined annotated categories.

Open-vocabulary segmentation. An interesting direction of research is to develop more widely applicable models capable of segmenting a large set of objects based on natural language description. To address the use of a fixed set of labels, CLIP [Radford, 2021] proposed to learn separate image and text representations jointly with a contrastive objective and formulate open-set image classification as an image-text retrieval problem. The use of transformer based architecture allows to easily adapt the approach to segmentation and perform segmentation on an open

vocabulary in a zero-shot setting. However, annotating images at the pixel level to obtain training datasets for these approaches is extremely costly.

Figure 1.5: (Top) Attention maps from an approach learning visual features without labels. The model learns to separate foreground from background without supervision. (Bottom) Patch attention maps from the first layer of a transformer trained for segmentation. We display the attention maps of three patches corresponding to sky, person and sidewalk classes.

Weak and self-supervised learning. Relaxing fully-supervised methods to weaker forms of supervision is a major challenge to scale segmentation approaches to much larger datasets. Weak and self-supervised methods aim at learning pixel level information without relying on pixel annotations. Weakly supervised methods leverage image-level information such as labels of objects present in an image or a caption describing the image to learn to localize objects in the image. Self-supervised methods only rely on images as supervision, for example, parts of the image are masked, and the model is trained to reconstruct the missing pixels. Developing these methods to match the performance of strongly supervised method is key to scale open-ended segmentation models to large datasets at a cheap annotation cost and obtain models able to segment new visual entities in the wild.

1.3 Contributions

1.3.1 Manipulation tasks as a composition of primitive skills

Given synthetic demonstrations of a set of basic manipulation skills such as *pushing a cube* or *pouring a cup*, we propose a vision-based method combining behavioral

cloning and reinforcement learning to perform more complex tasks such as *preparing a simple breakfast*. More precisely, we propose to learn a vocabulary of basic skills with behavioral cloning to obtain a variety of visually-guided control policies. Then, we learn a planing policy that compose skills to perform more complex tasks with reinforcement learning and show successful results on a real robot with policies trained in simulation.

To zero-shot transfer from simulation to a real world setting we rely on a sim-to-real method that automatically discovers sequence of random data augmentations. Sequences of data augmentations are discovered by optimizing object localization as a proxy task, and we show that augmentations transfer to policy learning. We can thus leverage simulation to collect rich datasets and interactions with the environment in order to learn policies deployable on a real robot.

1.3.2 Vision-based motion planning

Motion planning is a fundamental robotics problem usually approached with sampling-based methods. These methods typically rely on prior knowledge of the environment the robot is evolving in, however it is often difficult to obtain detailed a priori knowledge about the real state of environments and prevent from working in dynamically changing environments. We propose a neural motion planning approach that is sensor-based and generates motion plans using a policy sensing an unknown and possibly changing environment.

We evaluate the effectiveness of our method on several motion planning tasks including environments containing new objects, dynamically changing obstacles and a classical motion planning problem. Furthermore, we show improved performance over classical methods such as rapidly exploring random trees (RRT), especially in scenarios that are sample expansive.

1.3.3 Transferable segmentation models

Our contribution to the semantic segmentation field is twofold. First, we propose a new method Segmenter that addresses limitations of CNN based methods limited by the local nature of convolutional filters. We formulate semantic segmentation as a sequence-to-sequence problem and propose a transformer architecture based on the recent Vision Transformer [Dosovitskiy, 2021a] to leverage contextual information at every stage of the model. Our approach can capture global interactions of objects in a scene by design, and we show improvements on classical benchmarks of semantic

segmentation.

Second, we propose to extend weakly-supervised approach to an open set of visual concepts described with natural language. Our approach does not require pixel-wise supervision and learns segmentation masks directly from image-level referring expressions, overcoming scaling limitations from previous work that were developed for a fixed set of labels. We build on Segmenter and propose a transformer based approach that computes patch-text similarities to guide the regions corresponding to given natural language expressions. Our approach alleviates the need of manual supervision and paves the way to general methods able to learn segmentation masks on larger datasets than typical benchmarks in semantic segmentation. The model also transfers to new datasets without the need of any dataset specific training.

1.4 Outline

This manuscript is organized in eight chapters, including the introduction.

In Chapter 2, we review previous work in robotics and segmentation most related to this thesis. The robotics review focuses on perception-based methods for robotic manipulation. We review object-centric and task-centric representations that are the two main learning paradigm for sensor-based learning. The segmentation review presents recent methods for semantic segmentation and approaches developed to reduce the need of costly annotations. We present our contributions at the intersection of vision and robotics focused on learning visually-guided policies, then we introduce our contributions related to vision and improving image segmentation methods.

In Chapter 3, we present a method that learns to perform manipulation tasks from demonstrations. Given only demonstrations of primitive skills such as *picking* or *pouring a cup*, our algorithm learns to compose these skills to perform more complex tasks such as preparing a simple breakfast. While only trained on synthetic data, we show successful deployment on a real robot.

In Chapter 4, we develop a sim-to-real method to deploy policies on a real robot. Our method relies on object localization as a proxy task to optimize sequences of data augmentations. This allows to cheaply collect large datasets in simulation critical to the learning of policies which can be then transferred to a real robot without the need to collect real data.

In Chapter 5, we focus on motion planning and propose a method leveraging knowl-

edge of the surrounding environment to accelerate the finding of collision-free paths. We improve existing neural motion planning methods and propose a new approach achieving state-of-the-art performance on several tasks. While trained on a set of fixed obstacle, our method can handle dynamically changing environments.

Chapter 6 introduces the second part of this thesis and focuses on semantic segmentation. We propose a new method, Segmenter, that formulates segmentation as a sequence-to-sequence problem and extend the recent Vision Transformer to semantic segmentation.

In Chapter 7, we extend Segmenter and propose TSEG, a method that learns to localize objects described from natural language without pixel-level supervision. This approach is a step towards scaling segmentation methods to larger datasets without costly annotations.

Finally, in Chapter 8, we provide a summary of our contributions along with their limitations and discuss future directions.

1.5 Publications

This thesis is based on the following five publications:

[Strudel, 2020b]: Robin Strudel, Alexander Pashevich, Igor Kalevatykh, Ivan Laptev, Josef Sivic and Cordelia Schmid. Learning to combine primitive skills: A step towards versatile robotic manipulation. In ICRA, 2020. Presented in Chapter 3.

[Pashevich, 2019]: Alexander Pashevich, Robin Strudel, Igor Kalevatykh, Ivan Laptev and Cordelia Schmid. Learning to Augment Synthetic Images for Sim2Real Policy Transfer. In IROS, 2019. Presented in Chapter 4.

[Strudel, 2020a]: Robin Strudel, Ricardo Garcia, Justin Carpentier, Jean-Paul Laumond, Ivan Laptev and Cordelia Schmid. Learning Obstacle Representations for Neural Motion Planning. In CoRL, 2020. Presented in Chapter 5.

[Strudel, 2021]: Robin Strudel, Ricardo Garcia Pinel, Ivan Laptev and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation. In ICCV, 2021. Presented in Chapter 6.

[Strudel, 2022a]: Robin Strudel, Ivan Laptev and Cordelia Schmid. Weakly-supervised segmentation of referring expressions. ArXiv preprint, 2022. Presented in Chapter 7.

Another work by the author not included in this thesis is described in the following paper:

[Strudel, 2022b]: Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganing, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre and Rémi Leblond. Self-conditioned Embedding Diffusion for Text Generation. ArXiv preprint, 2022.

Chapter 2

Related work

This chapter is organized in two sections presenting related work at the intersection of robotics and vision. In Section 2.1, we present approaches to learn visual representations for robotics, these representations can either be learned by estimating properties of an object such as its pose or a set of keypoints, or by estimating sequences of control to be executed to perform a task. We then present image segmentation methods in Section 2.2, a core component of methods to perform object identification and localization. We describe seminal works in these domains along with their current limitations to motivate the contributions from this thesis.

2.1 Visual representations for robotics manipulation

2.1.1 Object-centric representations

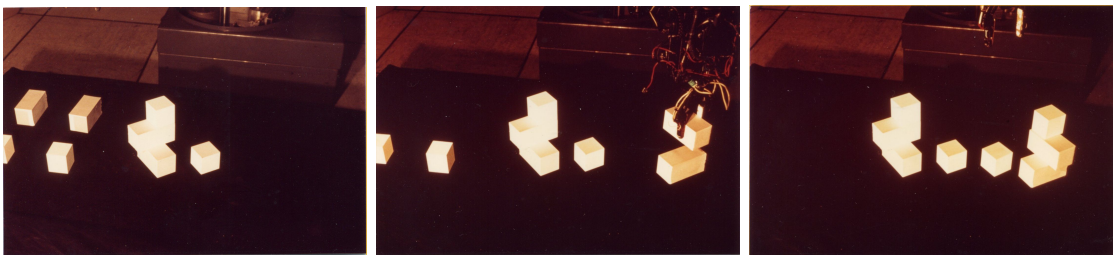


Figure 2.1: Example of the reconstruction of an assembly from the *copy demo* of Winston et al. [Winston, 1970] in 1970. Given an assembly of cubes (left), the robot copies the structure (right).

The problem of manipulating objects with a robot using visual feedback is a long-

standing problem and its first solution probably dates back to the *copy demo* of Patrick Winston et al. in 1970 [Winston, 1970]. Object-centric approaches focus on recovering the poses of known objects in the scene, then plan and manipulate the object according to the recovered poses.

Recent lines of work rely on pose estimation systems using either keypoints [Rothganger, 2006; He, 2020] or templates [Rad, 2017; Li, 2018; Xiang, 2018; Wang, 2019; Deng, 2020; Labbé, 2020] powered by CNNs. Pose estimation has been successfully applied to robotics in order to perform millimeter level assembly tasks [Choi, 2012; Litvak, 2019; Stevsic, 2020] in controlled environments with known 3D objects. It is currently probably one of the best approach to perform vision-based manipulation in highly-controlled industrial robotics environments.

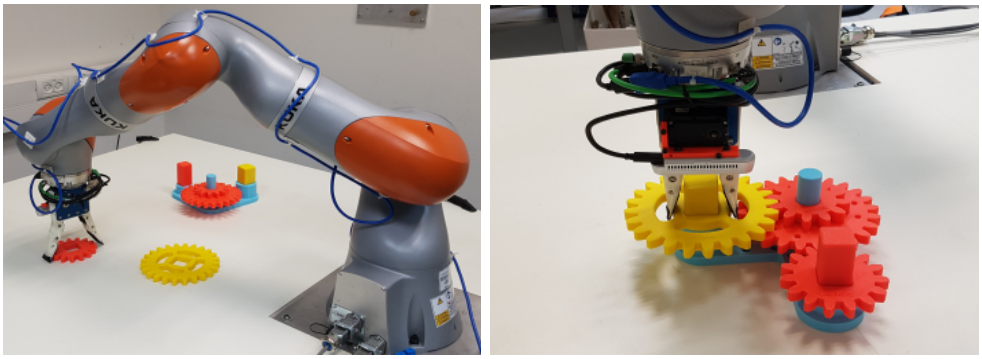


Figure 2.2: Vision-based assembly tasks requiring millimeter precision performed with pose estimation systems using known 3D models [Litvak, 2019] in 2019.

Pose estimation typically assumes access to known 3D models (CAD models) representing the objects to manipulate. Such assumption is limiting for real-world applications where the goal is to perform manipulation in an open-world where the CAD models of objects encountered are not available. The notion of pose of an object is already ambiguous for rigid objects with symmetries and does not easily extend to deformable objects such as ropes, laces or clothes. The notion of pose also breaks for objects that can be cut, when you cut a tomato, should each slice of the tomato be considered as a separate object ? If one wants to move a pile of sand with a robot, what is the atomic object to be detected ?

To circumvent such limitations, another line of work in robotics aims at learning visual representations directly from raw pixels using task demonstrations or by trial-and-errors. We present task-centric representations and policy learning in the next section, the main focus of robotics chapters of this thesis.

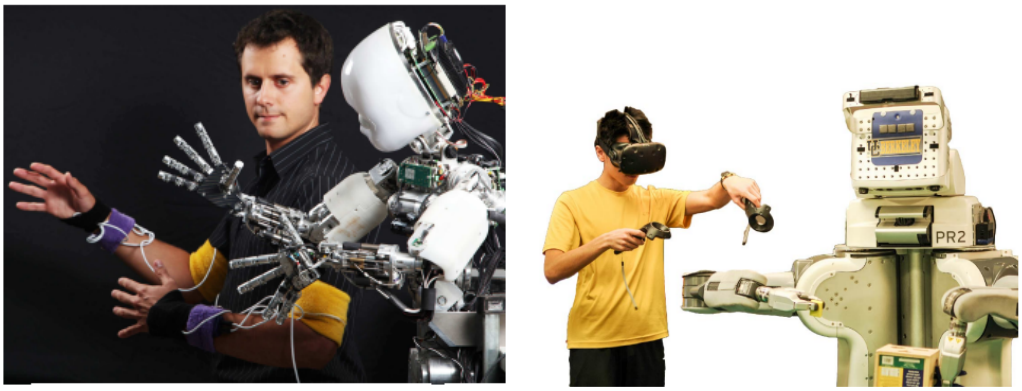


Figure 2.3: Example of human operators [Calinon, 2010; Zhang, 2018c] teleoperating a robot to provide demonstrates of a task. The dataset of demonstrations is then used to train a policy reproducing the task.

2.1.2 Task-centric representations: Imitation learning

One of the first approach to learn visual representations from raw pixels is the ALVINN system [Pomerleau, 1988] in 1988. Developed by Pomerleau at CMU, the approach allows performing autonomous vehicle navigation by processing input images and generating steering wheel controls with a neural network. A model directly mapping an input state to control is called a *policy*, the problem of learning a policy is particularly challenging when the input is an image as the model has to extract information directly from pixels. To learn a vision-based policy ALVINN [Pomerleau, 1988] relies on imitation learning, a class of methods for acquiring skills by observing demonstrations, see [Calinon, 2009; Argall, 2009] for surveys. Two main lines of work of imitation learning are behavioral cloning [Calinon, 2009; Calinon, 2010], that performs supervised learning from observations to actions, and inverse reinforcement learning [Ng, 2000; Abbeel, 2004; Ziebart, 2008] where a reward function is estimated to describe demonstrations as near optimal behaviors. This work focuses on Behavioral Cloning.

Behavioral cloning has been applied to a wide range of robotics tasks such as autonomous driving [Pomerleau, 1988; Bojarski, 2016; Giusti, 2016], autonomous helicopter flight [Abbeel, 2010] and manipulation [Calinon, 2010; Akgün, 2012; Zhang, 2018c; Florence, 2021]. BC was particularly successful in robotics when a low-dimensional representation of environment state is available [Schaal, 2003; Billard, 2004; Calinon, 2010]. However, in many scenarios it is challenging or intractable to extract low-dimension state information and hence it is desirable to learn policies taking as input raw sensory measurements. Thanks to recent progresses in deep learning, policies with a CNNs backbone [Zhang, 2018a; Strudel, 2020b; Florence,

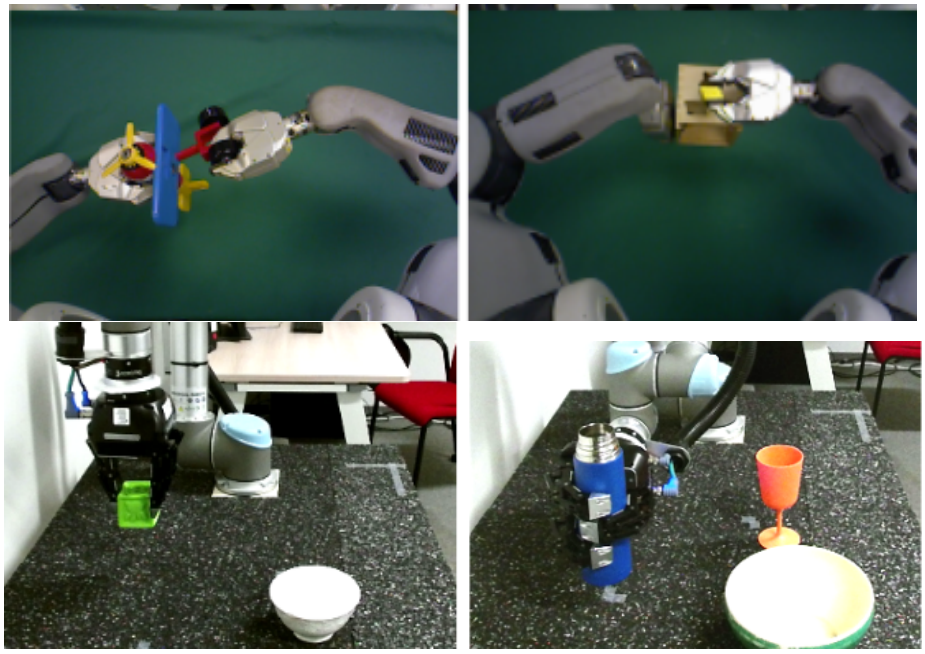


Figure 2.4: Examples of policies learned with behavioral cloning executing diverse tasks such as insertion, grasping or pouring. (Top) Demonstrations are collected by teleoperation [Zhang, 2018c]. (Bottom) Demonstrations are collected in simulation with scripts [Strudel, 2020b].

2021] can operate at the pixel level and learning a manipulation task from only 50 to 500 demonstrations. The use of implicit formulations also enabled to improve policy precision and perform precise manipulation tasks such as oriented block insertion and combinatorially complex sorting tasks.

2.1.3 Task-centric representations: Reinforcement learning

Reinforcement learning (RL) is a class of methods that autonomously learns policies through trial and error. The learner is not specified which actions to take, but instead must discover the sequence of actions yielding the highest return by exploring the environment. In the most interesting and challenging cases, actions may affect not only the immediate reward but also all future ones.

While there have been recent advances in deep RL leading to excellent performance in video games [Mnih, 2015] or Go [Silver, 2016], its application to robotics remains challenging. Indeed, as of today reinforcement learning is much more compute expensive than imitation learning and requires millions of interactions with the environment in order to perform stacking tasks [Zhu, 2018a; Riedmiller, 2018b; Lee, 2021a] or grasping tasks [Pinto, 2016a; Levine, 2018; Bousmalis, 2018; Kalashnikov,

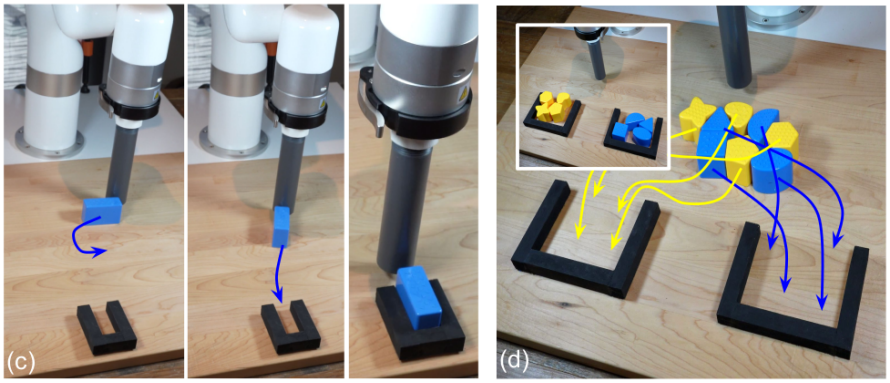


Figure 2.5: Solving precise manipulation tasks with implicit behavioral cloning [Florence, 2021]. (Left) Precise block insertion. (Right) Combinatorially complex sorting task.

2018]. Learning a visuo-motor controller from experience is exceedingly challenging for several reasons. Learning a CNNs handling real noisy sensor inputs from a 1D reward signal is a hard problem as the gradients are typically much noisier than in supervised learning [Sutton, 2018]. The tasks at hand are contact rich [Lee, 2021a; Levine, 2018] and induces complex physical interactions between objects that are hard to reproduce in simulation [Todorov, 2012a]. Finally, the action space of robotics tasks is continuous in contrast to board games [Silver, 2016] and video games [Mnih, 2015], leading to complex exploration problem that are partially circumvented with reward shaping [Lee, 2021a] or the use of demonstrations to guide the search [Zhu, 2018a].

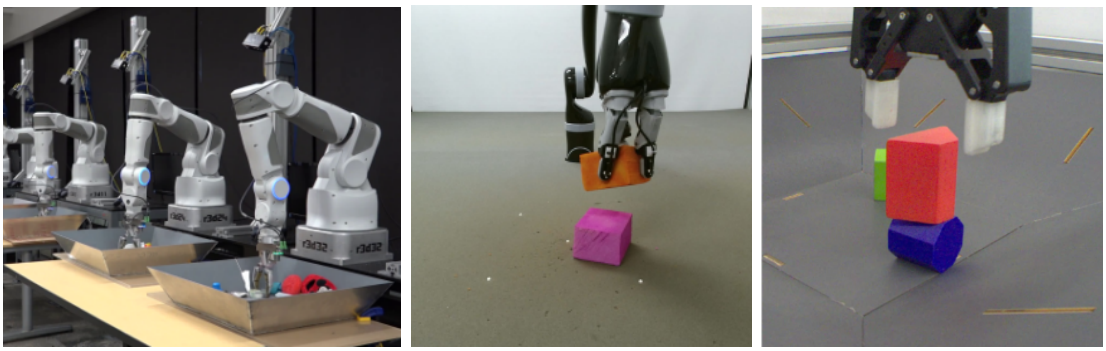


Figure 2.6: (Left) Large-scale data collection setup of 14 robotic manipulators collecting 800,000 grasp attempts to train a CNN grasp prediction model [Levine, 2018]. (Middle) Block stacking on a real robot learned by combining imitation learning and reinforcement learning [Zhu, 2018a]. (Right) Stacking objects of varying shapes involving complex contact dynamics [Lee, 2021a].

Obtaining a large number of interactions on a real robotic system in order to learn a policy is time-consuming or requires a large number of robots that might break due

to mechanical fatigue. Training policies in simulation is an interesting alternative allowing to speed up training thanks to parallelization at a cheap cost but also reproducibility, a key property that is typically not possible on real robotics system due to their complexity. However, there is typically a simulation-to-reality gap and policies trained in simulation transfer poorly as is to real robotic system. This problem has led to the development of sim2real methods which we present in the next section.

2.1.4 Simulation to real-world transfer

Learning visuo-motor policies in simulation and transferring them to the real world is an appealing approach to address difficulties inherent to real robotic systems. Simulation comes however at the cost of discrepancies, the two main sources are the system dynamics that are hard to simulate, particularly for contact-rich tasks, and the visual rendering is lacking sufficient realism and diversity.

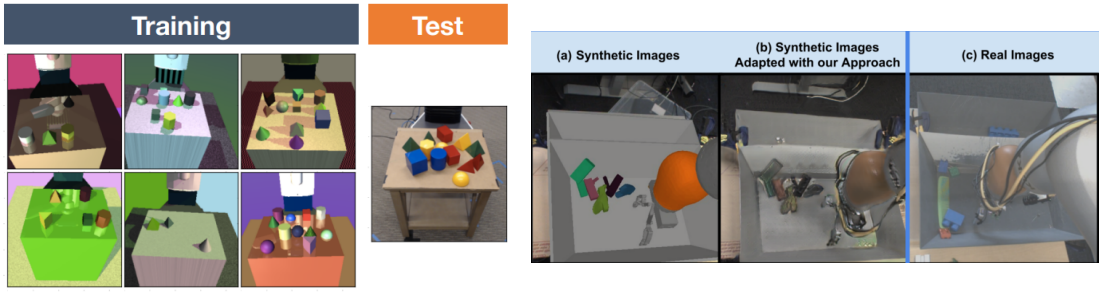


Figure 2.7: (Left) Domain randomization of synthetic images, textures and lighting are randomized to learn models invariant to visual changes [Tobin, 2017]. (Right) Domain adptation aims at mapping synthetic images to realisic looking images using generative models [Bousmalis, 2018].

This PhD focuses on the visual sim-to-real gap that is addressed using either domain randomization [Tobin, 2017; James, 2017; Peng, 2018] or domain adaptation [Bousmalis, 2018; Cubuk, 2019; Li, 2020; Zhao, 2020]. Domain randomization methods aim at randomizing the visual appearance of simulated environments by using random textures, lighting and camera pose [Tobin, 2017]. Instead of aiming at simulating visually realistic environments, the goal is to train a model to be invariant to visual variations which can then transfer to real sensor images. Domain adaptation methods focus on mapping synthetic images to real-looking one's by using generative adversarial networks (GANs) [Bousmalis, 2018]. While GANs are expressive models, they tend to generate artifacts and require adding a lot of manual constraints to preserve the 3D structure of the scene. In comparison, domain randomization preserves the 3D structure by design, a particularly desirable property

for robotics manipulation where preserving the geometry of the manipulator and objects is crucial.

2.2 Image segmentation

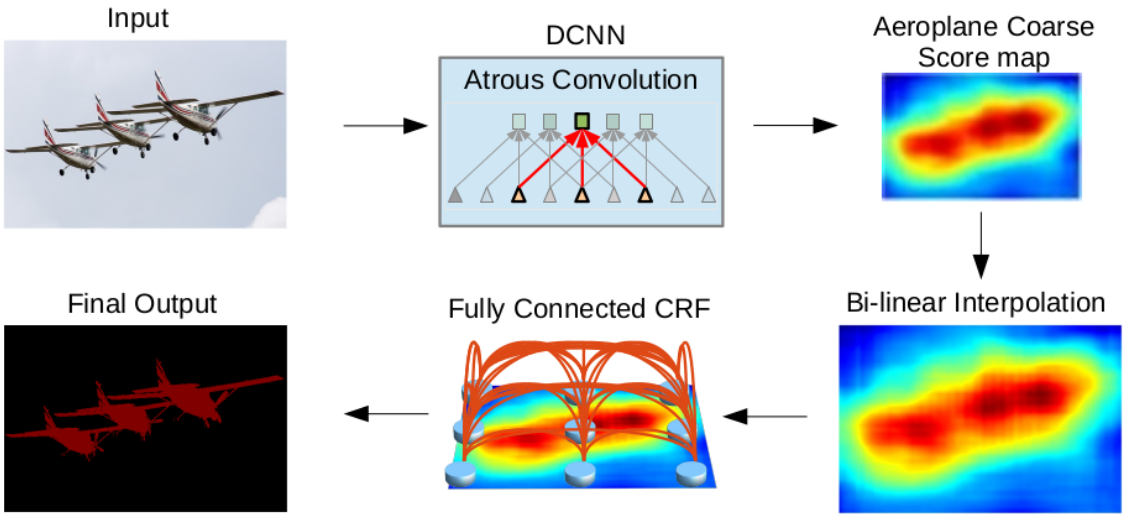


Figure 2.8: Semantic segmentation using DeepLab [Liang-Chieh, 2015], a CNN based approach combined with learnable atrous convolutions and non-learnable conditional random fields.

Most of the semantic segmentation approaches developed in the previous decade rely on hand-crafted features such as boosting [Tu, 2010], random forests [Shotton, 2008] or support vector machines [Fulkerson, 2009]. Many subsequent works focused on improving models by either using richer information from context or structured predictions. However, this body of work relied on handcrafted features that limit their expressiveness.

Deep learning breakthrough for image classification relying on convolutional neural networks in 2012 [Krizhevsky, 2012] has led to consequent improvements in semantic segmentation. Methods based on fully convolutional networks (FCN) [Farabet, 2013; Pinheiro, 2014; Ronneberger, 2015; Long, 2015; Amirul Islam, 2017; Badrinarayanan, 2017; Lin, 2017; Pohlen, 2017] learn task-oriented features by directly optimizing convolutions filters of a CNN. The seminal work of DeepLab [Liang-Chieh, 2015; Chen, 2018b; Chen, 2017; Chen, 2018d] proposed atrous convolutions and spatial pyramid pooling to enlarge the receptive field of convolutional filters for low and high level features.

Vision Transformers (ViT) [Dosovitskiy, 2021a] was recently proposed as a pure transformer approach to image classification, outperforming CNNs when pretrained

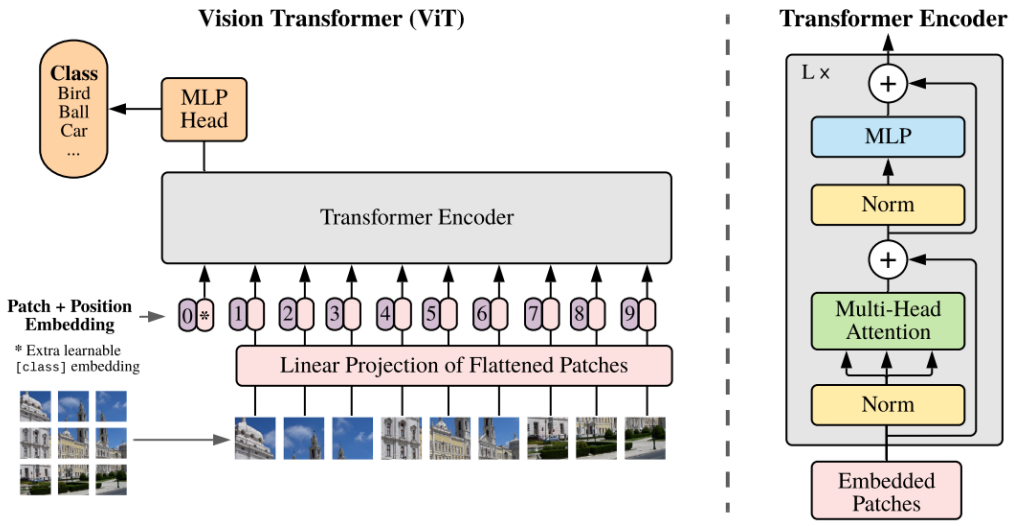


Figure 2.9: Vision Transformer [Dosovitskiy, 2021a] overview. An image is split into fixed-size patches that are projected into tokens and fed to a standard Transformer encoder.

on a large corpus of data. Transformers are based on an attention mechanism that computes individual patch features dependent on all the patches of the images and captures long-range dependencies by design and with learnable parameters. We build on ViT and propose to formulate segmentation as a sequence-to-sequence problem in our work Segmenter [Strudel, 2021] and present a pure transformer model to perform segmentation that does not rely on task specific layers such as atrous convolutions.

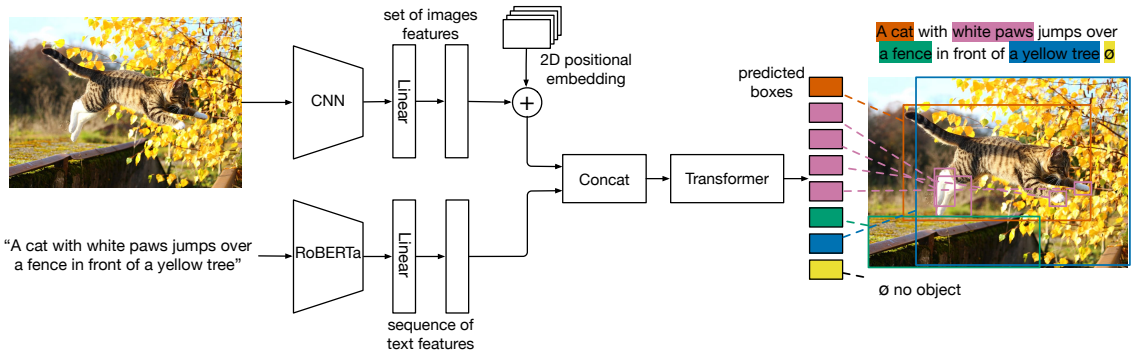


Figure 2.10: MDETR [Kamath, 2021] predicts the bounding boxes of the objects and their grounding in text. MDETR is a text-conditioned open-vocabulary object detector.

The performance of modern segmentation methods relies on dense annotations that are typically extremely costly to obtain. To address costly annotations, weakly supervised segmentation, introduced [Zhou, 2016], learns to localize by relying on image labels as supervision. Class activation maps are introduced to obtain class

masks that are then reduced to a class label with a pooling mechanism. As class activation maps tend to focus on the most discriminative parts of an object [Wei, 2017], many recent methods improve precision and recall by expanding masks using visual cues [Kolesnikov, 2016; Wei, 2017; Ahn, 2018; Fan, 2018; Ahn, 2019; Fan, 2019; Yu, 2019] and are consistently reducing the gap with fully supervised approaches.

Weakly-supervised segmentation is generally studied on datasets of restricted size such as Pascal VOC [Everingham, 2010] that contains 1500 images with 20 object categories. In order to scale these methods, it is necessary to consider the more general problem of localizing an open set of objects defined by natural language instead of a fixed set of labels. Multi-modal relations between texts and images [Frome, 2013] recently led to impressive vision-language models trained at scale such as CLIP [Radford, 2021], MDETR [Kamath, 2021], DALL-E 2 [Ramesh, 2022] or OWL-ViT [Minderer, 2022]. In this thesis, we build on CLIP and propose a new approach TSEG [Strudel, 2022a] that performs weakly-supervised segmentation from referring expressions, a set of short texts describing visual entities present in the image.

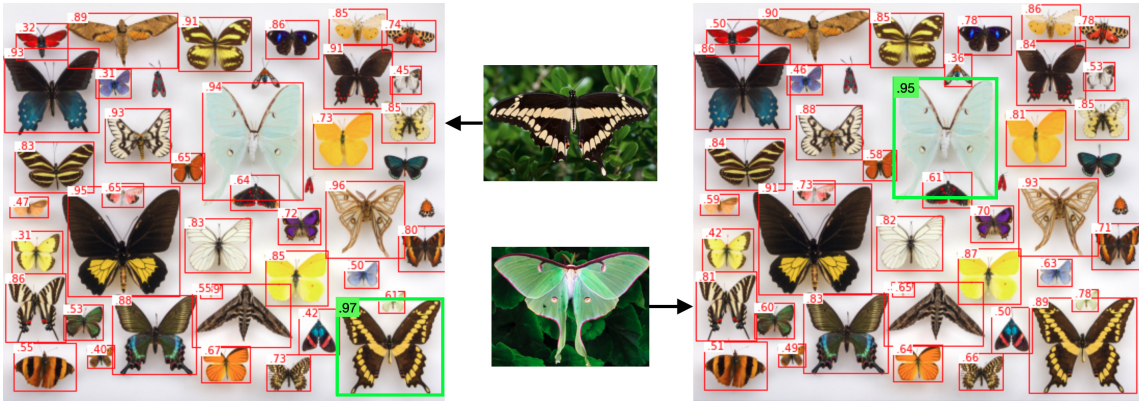


Figure 2.11: OWL-ViT [Minderer, 2022] predicts the bounding box of the object given by an image, this is an example image-conditioned open-vocabulary detector.

Developing open-vocabulary detection and localization models is key to obtain generally capable vision models that can be deployed and reused for many tasks including robotics manipulation of novel objects in the wild. Figure 2.10 is an overview of MDETR [Kamath, 2021], a supervised model trained to detect objects given a caption. Figure 2.11 presents detection performed with OWL-ViT [Minderer, 2022], another model able to perform open-set vocabulary object detection while conditioned on a single image depicting the object or a text description.

Chapter 3

Learning to combine primitive skills for versatile robotic manipulation

In this chapter, we propose an approach to learn manipulation tasks as a composition of primitive skills. We learn vision-based manipulation policies in simulation and deploy them on a real robot. Manipulation tasks such as preparing a meal or assembling furniture remain highly challenging for robotics and vision. Traditional task and motion planning (TAMP) methods can solve complex tasks but require full state observability and are not adapted to dynamic scene changes. Recent learning methods can operate directly on visual inputs but typically require many demonstrations and/or task-specific reward engineering. In this work we aim to overcome previous limitations and propose a reinforcement learning (RL) approach to task planning that learns to combine primitive skills. First, compared to previous learning methods, our approach requires neither intermediate rewards nor complete task demonstrations during training. Second, we demonstrate the versatility of our vision-based task planning in challenging settings with temporary occlusions and dynamic scene changes. Third, we propose an efficient training of basic skills from few synthetic demonstrations by exploring recent CNN architectures and data augmentation. Notably, while all of our policies are learned on visual inputs in simulated environments, we demonstrate the successful transfer and high success rates when applying such policies to manipulation tasks on a real UR5 robotic arm.

3.1 Introduction

We consider visually guided robotics manipulations and aim to learn robust visuo-motor control policies for particular tasks. Autonomous manipulations such as assembling IKEA furniture [Suárez-Ruiz, 2018] remain highly challenging given the complexity of real environments as well as partial and uncertain observations provided by the sensors. Successful methods for task and motion planning (TAMP) [Srivastava, 2014; Lozano-Pérez, 2014; Toussaint, 2015] achieve impressive results for complex tasks but often rely on limiting assumptions such as the full state observability and known 3D shape models for manipulated objects. Moreover, TAMP methods usually complete planning before execution and are not robust to dynamic scene changes.

Recent learning methods aim to learn visuo-motor control policies directly from image inputs. Imitation learning (IL) [Pomerleau, 1988; Ross, 2014; Pinto, 2018; Ng, 2000] is a supervised approach that can be used to learn simple skills from expert demonstrations. One drawback of IL is its difficulty to handle new states that have not been observed during demonstrations. While increasing the number of demonstrations helps to alleviate this issue, an exhaustive sampling of action sequences and scenarios becomes impractical for long and complex tasks.

In contrast, reinforcement learning (RL) requires little supervision and achieves excellent results for some challenging tasks [Mnih, 2015; Silver, 2016]. RL explores previously unseen scenarios and, hence, can generalize beyond expert demonstrations. As full exploration is exponentially hard and becomes impractical for problems with long horizons, RL often relies on careful engineering of rewards designed for specific tasks.

Common tasks such as preparing food or assembling furniture require long sequences of steps composed of many actions. Such tasks have long horizons and, hence, are difficult to solve by either RL or IL methods alone. To address this issue, we propose a RL-based method that learns to combine simple imitation-based policies. Our approach simplifies RL by reducing its exploration to sequences with a limited number of primitive actions, that we call *skills*.

Given a set of pre-trained skills such as "grasp a cube" or "pour from a cup", we train RL with sparse binary rewards corresponding to the correct/incorrect execution of the full task. While hierarchical policies have been proposed in the past [Das, 2018], our approach can learn *composite manipulations using no intermediate rewards and no demonstrations of full tasks*. Hence, the proposed method can be directly applied

to learn new tasks. See Figure 3.1 for an overview of our approach.

Our skills are low-level visuo-motor controllers learned from synthetic demonstrated trajectories with behavioral cloning (BC) [Pomerleau, 1988]. Examples of skills include *go to the bowl*, *grasp the object*, *pour from the held object*, *release the held object*, etc. We automatically generate expert synthetic demonstrations and learn corresponding skills in simulated environments. We also minimize the number of required demonstrations by choosing appropriate CNN architectures and data augmentation methods. Our approach is shown to compare favorably to the state of the art [Pinto, 2018] on the FetchPickPlace test environment [Plappert, 2018]. Moreover, using recent techniques for domain adaptation [Pashevich, 2019] we demonstrate the successful transfer and high accuracy of our simulator-trained policies when tested on a real robot

We compare our approach with two classical methods: (a) an open-loop controller estimating object positions and applying a standard motion planner (b) a closed-loop controller adapting the control to re-estimated object positions. We show the robustness of our approach to a variety of perturbations. The perturbations include dynamic change of object positions, new object instances and temporary object occlusions. The versatility of learned policies comes from both the reactivity of the BC learned skills and the ability of the RL master policy to re-plan in case of failure. Our approach allows computing adaptive control and planning in real-time.

In summary, this work makes the following contributions. (i) We propose to learn robust RL policies that combine BC skills to solve composite tasks. (ii) We present sample efficient training of BC skills and demonstrate an improvement compared to the state of the art. (iii) We demonstrate successful learning of relatively complex manipulation tasks with neither intermediate rewards nor full demonstrations. (iv) We successfully transfer and execute policies learned in simulation to real robot setups. (v) We show successful task completion in the presence of perturbations.

Our simulation environments together with the code and models used in this work is publicly available at `https://www.di.ens.fr/willow/research/rlbc/`.

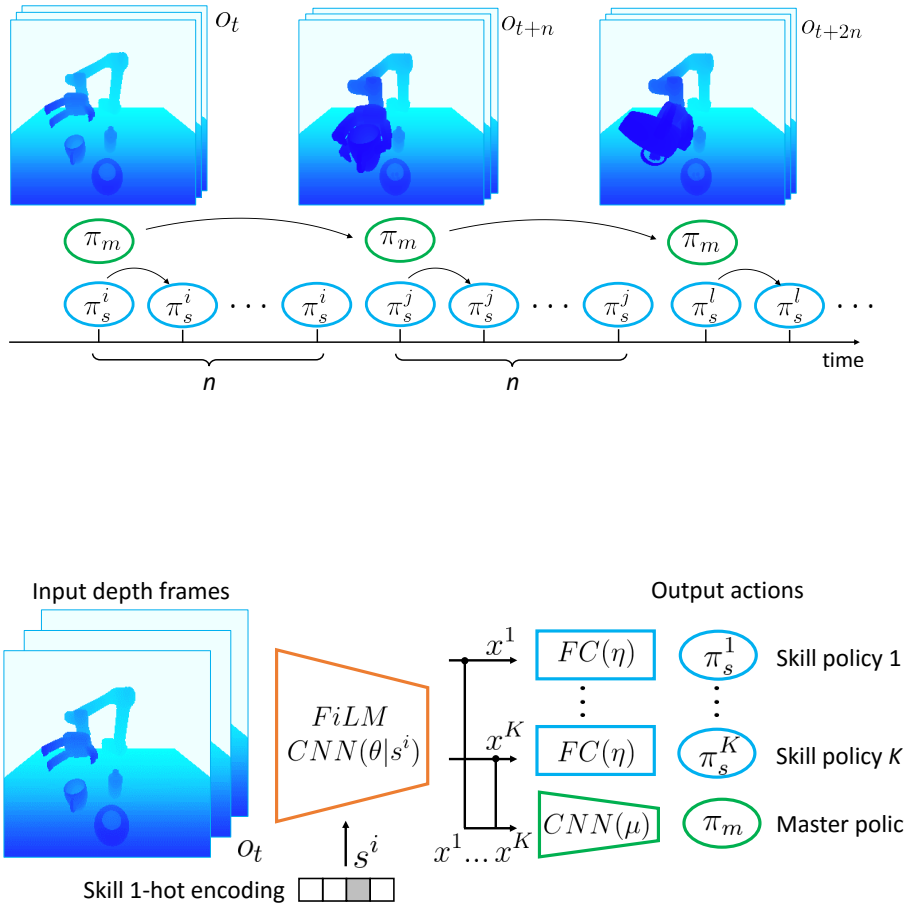


Figure 3.1: Illustration of our approach. (Left): Temporal hierarchy of master and skill policies. The master policy π_m is executed at a coarse interval of n time-steps to select among K skill policies $\pi_s^1 \ldots \pi_s^K$. Each skill policy generates control for a primitive action such as *grasping* or *pouring*. (Right): CNN architecture used for the skill and master policies.

3.2 Related work

Our work is related to robotics manipulation such as grasping [Lampe, 2013], opening doors [Gu, 2016], screwing the cap of a bottle [Levine, 2015] and cube stacking [Popov, 2017]. Such tasks have been addressed by various methods including imitation learning (IL) [Duan, 2017] and reinforcement learning (RL) [Riedmiller, 2018a].

Imitation learning (IL). A neural network is trained to solve a task by observing demonstrations. Approaches include behavioral cloning (BC) [Pomerleau, 1988]

and inverse reinforcement learning [Ng, 2000]. BC learns a function that maps states to expert actions [Ross, 2014; Pinto, 2018], whereas inverse reinforcement learning learns a reward function from demonstrations in order to solve the task with RL [Ho, 2016; Kumar, 2016; Popov, 2017]. BC typically requires a large number of demonstrations and has issues with not observed trajectories. While these problems might be solved with additional expert supervision [Ross, 2014] or noise injection in expert demonstrations [Laskey, 2017], we address them by improving the standard BC framework. We use recent state-of-the-art CNN architectures and data augmentation for expert trajectories. This permits to significantly reduce the number of required demonstrations and to improve performance.

Reinforcement learning (RL). RL learns to solve a task without demonstrations using exploration. Despite impressive results in several domains [Silver, 2016; Mnih, 2015; Kober, 2013; Gu, 2016], RL methods show limited capabilities when operating in complex and sparse-reward environments common in robotics. Moreover, RL methods typically require prohibitively large amounts of interactions with the environment during training. Hierarchical RL (HRL) methods alleviate some of these problems by learning a high-level policy modulating low-level workers. HRL approaches are generally based either on options [Sutton, 1999] or a feudal framework [Dayan, 1993]. The option methods learn a master policy that switches between separate skill policies [Frans, 2018; Lee, 2019c; Bacon, 2017; Florensa, 2017]. The feudal approaches learn a master policy that modulates a low-level policy by a control signal [Haarnoja, 2018a; Nachum, 2018; Vezhnevets, 2017; Kulkarni, 2016; Hausman, 2018]. Our approach is based on options but in contrast to the cited methods, we pretrain the skills with IL. This allows us to solve complex and sparse reward problems using significantly less interactions with the environment during training.

Combining RL and IL. A number of approaches combining RL and IL have been introduced recently. Gao et al. [Gao, 2018] use demonstrations to initialize the RL agent. [Cheng, 2018; Sun, 2018] use RL to improve expert demonstrations, but do not learn hierarchical policies. Demonstrations have been also used to define RL objective functions [Hester, 2018; Nair, 2018] and rewards [Zhu, 2018b]. Das et al. [Das, 2018] combines IL and RL to learn a hierarchical policy. Unlike our method, however, [Das, 2018] requires full task demonstrations and task-specific reward engineering. Moreover, the addressed navigation problem in [Das, 2018] has a much lower time horizon compared to our tasks. [Das, 2018] also relies on pre-trained CNN representations which limits its application domain. Le at al. [Le, 2018] train low-level skills with RL, while using demonstrations to switch between skills. In

a reverse manner, we use IL to learn low-level control and then deploy RL to find appropriate sequences of pre-trained skills. The advantage is that our method can learn a variety of complex manipulations without full task demonstrations. Moreover, [Das, 2018; Le, 2018] learn discrete actions and cannot be directly applied to robotics manipulations that require continuous control.

In summary, none of the methods [Das, 2018; Le, 2018; Cheng, 2018; Sun, 2018] is directly suitable for learning complex robotic manipulations due to requirements of dense rewards [Das, 2018; Sun, 2018] and state inputs [Cheng, 2018; Sun, 2018], limitations to short horizons and discrete actions [Das, 2018; Le, 2018], the requirement of full task demonstrations [Das, 2018; Le, 2018; Cheng, 2018; Sun, 2018] and the lack of learning of visual representations [Das, 2018; Cheng, 2018; Sun, 2018]. Moreover, our skills learned from synthetic demonstrated trajectories outperform RL based methods, see Section 3.5.4.

3.3 Planing with a vocabulary of learned skills

Our RLBC approach aims to learn multi-step policies by combining reinforcement learning (RL) and pre-trained skills obtained with behavioral cloning (BC). We present BC and RLBC in Sections 3.3.1 and 3.3.2. Implementation details are given in Section 3.3.3.

3.3.1 Skill learning with behavioral cloning

Our first goal is to learn basic skills that can be composed into more complex policies. Given observation-action pairs $\mathcal{D} = \{(o_t, a_t)\}$ along expert trajectories, we follow the behavioral cloning approach [Pomerleau, 1988] and learn a function approximating the conditional distribution of the expert policy $\pi_{\mathrm{E}}(a_t|o_t)$ controlling a robot arm. Our observations $o_t \in \mathcal{O} = \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times\mathrm{M}}$ are sequences of the last M depth frames. Actions $a_t = (\mathbf{v}_t, \boldsymbol{\omega}_t, g_t)$, $a_t \in \mathcal{A}^{\mathrm{BC}}$ are defined by the end-effector linear velocity $\mathbf{v}_t \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^3$ as well as the gripper openness state $g_t \in \{0, 1\}$.

We learn the deterministic skill policies $\pi_s : \mathcal{O} \to \mathcal{A}^{\mathrm{BC}}$ approximating the expert policy π_{E}. Given observations o_t with corresponding expert (ground truth) actions $a_t = (\mathbf{v}_t, \boldsymbol{\omega}_t, g_t)$, we represent π_s with a convolutional neural network (CNN) and learn network parameters (θ, η) such that predicted actions $\pi_s(o_t) = (\hat{\mathbf{v}}_t, \hat{\boldsymbol{\omega}}_t, \hat{g}_t)$ minimize the loss where $\lambda \in [0, 1]$ is a scaling factor which we empirically set to 0.9.

Our network architecture is presented in Figure 3.1(right). When training a skill policy π_s^i, such as reaching, grasping or pouring, we condition the network on the skill using the recent FiLM architecture [Perez, 2018]. Given the one-hot encoding s^i of a skill i, we use s^i as input to the FiLM generator which performs affine transformations of the network feature maps. FiLM conditions the network on performing a given skill, which permits learning a shared representation for all skills. Given an observation o_t, the network $CNN(\theta|s^i)$ generates a feature map x_t^i conditioned on skill i. The spatially-averaged x_i is linearly mapped with $FC(\eta)$ to the action of π_s^i.

3.3.2 RLBC approach

We wish to solve composite manipulations without full expert demonstrations and with a single sparse reward. For this purpose we rely on a high-level *master policy* π_m controlling the pre-trained *skill policies* π_s at a coarse timescale. To learn π_m, we follow the standard formulation of reinforcement learning and maximize the expected return $\mathbb{E}_\pi \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ given rewards r_t. Our reward function is sparse and returns 1 upon successful termination of the task and 0 otherwise. The RL master policy $\pi_m : \mathcal{O} \times \mathcal{A}^{\text{RL}} \rightarrow [0, 1]$ chooses one of the K skill policies to execute the low-level control, i.e., the action space of π_m is discrete: $\mathcal{A}^{\text{RL}} = \{1, \ldots, K\}$. Note, that our sparse reward function makes the learning of deep visual representations challenging. We, therefore, train π_m using visual features x_t^i obtained from the BC pre-trained $CNN(\theta|s^i)$. Given an observation o_t, we use the concatenation of skill-conditioned features $\{x_t^1, \ldots, x_t^K\}$ as input for the master $CNN(\mu)$, see Figure 3.1(right).

To solve composite tasks with sparse rewards, we use a coarse timescale for the master policy. The selected skill policy controls the robot for n consecutive time-steps before the master policy is activated again to choose a new skill. This allows the master to focus on high-level task planning rather than low-level motion planning achieved by the skills. We expect the master policy to recover from unexpected events, for example, if an object slips out of a gripper, by re-activating an appropriate skill policy. Our combination of the master and skill policies is illustrated in Figure 3.1(left).

RLBC algorithm. The pseudo-code for the proposed approach is shown in Algorithm 1. The algorithm can be divided into three main steps. First, we collect a dataset of expert trajectories $\mathcal{D}_k$ for each skill policy π_s^k. For each policy, we use an expert script that has access to the full state of the environment. Next, we train a set of skill policies $\{\pi_s^1, \ldots, \pi_s^K\}$. We sample a batch of state-action pairs and update parameters of convolutional layers θ and the skills linear layer parameters

η. Finally, we learn the master π_m using the pretrained skill policies and the frozen parameters θ. We collect episode rollouts by first choosing a skill policy with the master and then applying the selected skill to the environment for n time-steps. We update the master policy weights μ to maximize the expected sum of rewards.

Algorithm 1 RLBC

*** Collect expert data ***
for $k \in \{1, \ldots, K\}$ **do**
    Collect an expert dataset $\mathcal{D}_k$ for the skill policy π_s^k
end for
*** Train $\{\pi_s^1, \ldots, \pi_s^K\}$ by solving: ***
$\theta, \eta = \arg\min_{\theta,\eta} \sum_{k=1}^{K} \sum_{(o_t,a_t)\in\mathcal{D}_k} L_{\text{BC}}(\pi_s^k(o_t), a_t)$
while task is not solved **do**
    *** Collect data for the master policy ***
    $\mathcal{E} = \{\}$ ▷ Empty storage for rollouts
    for episode_id $\in \{1, \ldots,$ ppo_num_episodes$\}$ **do**
        $o_0 =$ new_episode_observation()
        $t = 0$
        while episode is not terminated **do**
            $k_t \sim \pi_m(o_t)$ ▷ Choose the skill policy
            $o_{t+n}, r_{t+n} =$ perform_skill$(\pi_s^{k_t}, o_t)$
            $t = t + n$
        end while
        $\mathcal{E} = \mathcal{E} \cup \{(o_0, k_1, r_1, o_1, k_2, r_2, o_2, \ldots)\}$
    end for
    *** Make a PPO step for the master policy on $\mathcal{E}$ ***
    $\mu =$ ppo_update$(\pi_m, \mathcal{E})$
end while

3.3.3 Approach details

Skill learning with BC. We use ResNet-18 for the $CNN(\theta|s^i)$, which we compare to VGG16 and ResNet-101 in Section 3.5.1. We augment input depth frames with random translations, rotations and crops. We also perform viewpoint augmentation and sample the camera positions on a section of a sphere centered on the robot and with a radius of 1.40 m. We uniformly sample the yaw angle in $[-15°, 15°]$, the pitch angle in $[15°, 30°]$, and the distance to the robot base in $[1.35, 1.50]$ m. The impact of both augmentations is evaluated in Section 3.5.2. We normalize the ground truth of the expert actions to have zero mean and a unit variance and normalize the depth values of input frames to $[-1, 1]$. We learn BC skills using Adam [Kingma, 2015a] with the learning rate 10^{-3} and a batch size 64. We also use Batch Normalization [Ioffe, 2015a].

Task learning with RL. We learn the master policies with the PPO [Schulman, 2017] algorithm using the open-source implementation [Kostrikov, 2018] where we set the entropy coefficient to 0.05, the value loss coefficient to 1, and use 8 episode

rollouts for the PPO update. For the RLBC method, the concatenated skill features $\{x_t^1, \ldots, x_t^K\}$ are processed with the master network $CNN(\mu)$ having 2 convolutional layers with 64 filters of size 3×3. During pre-training of skill policies we update the parameters (θ, η). When training the master policy, we only update μ while keeping (θ, η) parameters fixed. We train RLBC using 8 different random seeds in parallel and evaluate the best one.

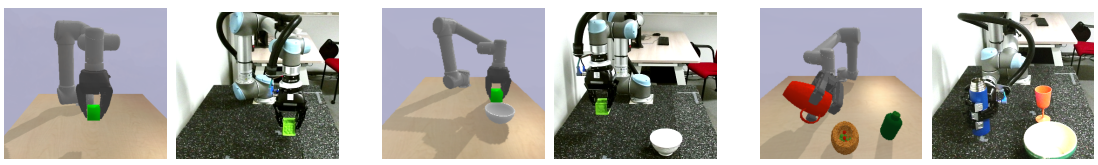
Real robot transfer. To apply our method on the real robot, we use a state-of-the-art technique of learning sim2real transfer based on data augmentation with domain randomization [Pashevich, 2019]. This method uses a proxy task of cube position prediction and a set of basic image transformations to learn a sim2real data augmentation function for depth images. We augment the depth frames from synthetic expert demonstrations with this method and, then, train skill policies. Once the skill policy is trained on these augmented simulation images, it is directly used on the real robot.

3.4 Experimental setup

This section describes the setup used to evaluate our approach. First, we present the robot environment and the different tasks in Sections 3.4.1 and 3.4.2. Next, we describe the synthetic dataset generation and skill definition for each task in Sections 3.4.3 and 3.4.4.

3.4.1 Robot and agent environment

For our experiments we use a 6-DoF UR5 robotic arm with a 3 finger Robotiq gripper, see Figure 3.2. In simulation, we model the robot with the `pybullet` physics simulator [Courmans, 2016]. For observation, we record depth images with the Microsoft Kinect 2 placed in front of the arm. The agent takes as input the three last depth frames $o_t \in \mathbb{R}^{224 \times 224 \times 3}$ and commands the robot with an action $a_t \in \mathbb{R}^7$. The control is performed at 10 Hz frequency.



(a) UR5-Pick (b) UR5-Bowl (c) UR5-Breakfast

Figure 3.2: UR5 tasks used for evaluation: (a) task of picking up the cube, (b) task of bringing the cube to the bowl, (c) task of pouring the cup and the bottle into the bowl. (Left) simulation, (right) real robot.

3.4.2 UR5 tasks

For evaluation, we consider 3 tasks: UR5-Pick, UR5-Bowl and UR5-Breakfast. The **UR5-Pick** task picks up a cube of a size between 3.5 cm and 8.0 cm and lifts it up, see Figure 3.2a. In **UR5-Bowl** the robot has to grasp the cube and place it in the bowl, see Figure 3.2b. The **UR5-Breakfast** task contains a cup, a bottle and a bowl as shown in Figure 3.2c. We use distinct ShapeNet [Chang, 2015a] object instances for the training and test sets (27 bottles, 17 cups, 32 bowls in each set). The robot needs to pour ingredients from the cup and the bottle in the bowl. In all tasks, the reward is positive if and only if the task goal is reached. The maximum episode lengths are 200, 600, and 2000 time-steps for UR5-Pick, UR5-Bowl, and UR5-Breakfast correspondingly.

3.4.3 Synthetic datasets

We use the simulated environments to create a synthetic training and test set. For all our experiments, we collect trajectories with random initial configurations where the objects and the end-effector are allocated within a workspace of $80 \times 40 \times 20\,\mathrm{cm}^3$. The synthetic demonstrations are collected using an expert script designed for each skill. The script has access to the full state of the system including the states of the robot and the objects. To generate synthetic demonstrations, we program end-effector trajectories and use inverse kinematics (IK) to generate corresponding trajectories in the robot joints space. Each demonstration consists of multiple pairs of the three last camera observations and the robot control command performed by the expert script. For UR5-Pick, we collect 1000 synthetic demonstrated trajectories for training. For UR5-Bowl and UR5-Breakfast, we collect a training dataset of 250 synthetic demonstrations. For evaluation of each task, we use 100 different initial configurations in simulation and 20 trials on the real robot.

3.4.4 Skill definition

UR5-Pick task is defined as a single skill. For UR5-Bowl and UR5-Breakfast, we consider a set of skills defined by expert scripts. For UR5-Bowl, we define four skills: (a) go to the cube, (b) go down and grasp, (c) go up, and (d) go to the bowl and open the gripper. For UR5-Breakfast, we define four skills: (a) go to the bottle, (b) go to the cup, (c) grasp an object and pour it to the bowl, and (d) release the held object. We emphasize that the expert dataset does not contain full task demonstrations and that all our training is done in simulation. When training the RL master, we execute selected skills for 60 consecutive time-steps for the UR5-Bowl task and 220

time-steps for the UR5-Breakfast task.

3.5 Evaluation of BC skill learning

This section evaluates the different parameters of the BC skill training for the UR5-Pick task and a comparison with the state of the art. First, we evaluate the impact of the CNN architecture and data augmentation on the skill performance in Sections 3.5.1 and 3.5.2. Then, we show that the learned policies transfer to a real-robot in Section 3.5.3. Finally, we compare the BC skills with the state of the art in Section 3.5.4.

3.5.1 CNN architecture for BC skill learning

Given the simulated UR5-Pick task illustrated in Figure 3.2a(left), we compare BC skill networks trained with different CNN architectures and varying number of expert demonstrations. Table 3.1 compares the success rates of policies with VGG and ResNet architectures. Policies based on the VGG architecture [Simonyan, 2014] obtain success rate below 40% with 100 training demonstrations and reach 95% with 1000 demonstrations. ResNet [He, 2016a] based policies have a success rate above 60% when trained on a dataset of 100 demonstrations and reach 100% with 1000 demonstrations. Overall ResNet-101 has the best performance closely followed by ResNet-18 and outperforms VGG significantly. To conclude, we find that the network architecture has a fundamental impact on the BC performance. In the following experiments we use ResNet-18 as it presents a good trade-off between performance and training time.

When examining why VGG-based BC has a lower success rate, we observe that it has higher validation errors compared to ResNet. This indicates that VGG performs worse on the level of individual steps and is hence expected to result in higher compounding errors.

Demos	VGG16-BN	ResNet-18	ResNet-101
20	1%	1%	0%
50	9%	5%	5%
100	37%	65%	86%
1000	95%	**100%**	**100%**

Table 3.1: Evaluation of BC skills trained with different CNN architectures and number of demonstrations on the UR5-Pick task in simulation.

Demos	None	Standard	Viewpoint	Standard & Viewpoint
20	1%	49%	39%	75%
50	5%	81%	79%	93%
100	65%	97%	**100%**	**100%**

Table 3.2: Evaluation of ResNet-18 BC skills trained with different data augmentations on UR5-Pick task in simulation.

3.5.2 Evaluation of data augmentation

We evaluate the impact of different types of data augmentations in Table 3.2. We compare training without data augmentation with 3 variants: (1) random translations, rotations and crops, as is standard for object detection, (2) record each expert synthetic demonstration from 10 varying viewpoints and (3) the combination of (1) and (2).

Success rates for UR5-Pick on datasets with 20, 50 and 100 demonstrations are reported in Table 3.2. We observe that data augmentation is particularly important when only a few demonstrations are available. For 20 demonstrations, the policy trained with no augmentation performs at 1% while the policy trained with standard and viewpoint augmentations together performs at 75%. The policy trained with a combination of both augmentation types performs the best and achieves 93% and 100% success rate for 50 and 100 demonstrations respectively. In summary, data augmentation allows a significant reduction in the number of expert trajectories required to solve the task.

3.5.3 Real robot experiments

We evaluate our method on the real-world UR5-Pick illustrated in Figure 3.2a(right). We collect demonstrated trajectories in simulation and train the BC skills network applying standard, viewpoint and sim2real augmentations. We show that our approach transfers well to the real robot using no real images. The learned policy manages to pick up cubes of 3 different sizes correctly in 20 out of 20 trials.

3.5.4 Comparison with state-of-the-art methods

One of the few test-beds for robotic manipulation is FetchPickPlace from OpenAI Gym [Plappert, 2018] implemented in `mujoco` [Todorov, 2012b], see Figure 3.3. The goal for the agent is to pick up a cube and to move it to the red target (see Figure 3.3). The agent observes the three last RGB-D images from a camera placed

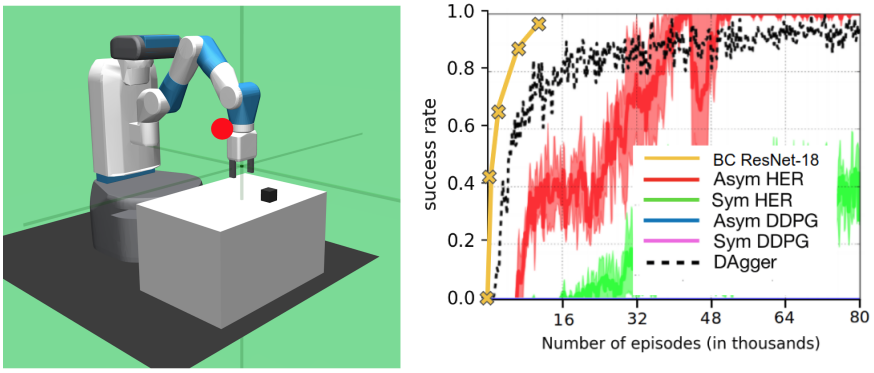


Figure 3.3: Comparison of BC ResNet-18 with state of the art [Pinto, 2018] on the FetchPickPlace task. BC ResNet-18 results are reported for 200 different initial configurations.

in front of the robot $o_t \in \mathbb{R}^{100\times100\times4\times3}$. The positions of the cube and the target are set at random for each trial. The reward of the task is a single sparse reward of success. The maximum length of the task is set to 50 time-steps.

For a fair comparison with [Pinto, 2018], we do not use any data augmentation. We report the success rate of ResNet-18 policy in Figure 3.3. We follow [Pinto, 2018] and plot the success rate of both RL and IL methods with respect to the number of episodes used (either trial episodes or demonstrations). Our approach outperforms the policies trained with an imitation learning method DAgger [Ross, 2014] in terms of performance and RL methods such as HER [Andrychowicz, 2017a] and DDPG [Lillicrap, 2016] in terms of data-efficiency. According to [Pinto, 2018], DAgger does not reach 100% even after $8 * 10^4$ demonstrations despite the fact that it requires an expert during training. HER reaches the success rate of 100% but requires about $4 * 10^4$ trial episodes. Our approach achieves the 96% success rate using 10^4 demonstrations.

Our policies differ from [Pinto, 2018] mainly in the CNN architecture. Pinto et al. [Pinto, 2018] use a simple CNN with 4 convolutional layers while we use ResNet-18. Results of this section confirm the large impact of the CNN architecture on the performance of visual BC policies, as was already observed in Table 3.1.

3.6 Evaluation of RLBC

This section evaluates the proposed RLBC approach and compares it to baselines introduced in Section 3.6.1. First, we evaluate our method on UR5-Bowl in Section 3.4.2. We then test the robustness of our approach to various perturbations

UR5-Bowl perturbations	Detect & Plan	Detect & Replan	BC-ordered	RLBC
No perturbations	17/20	16/20	17/20	**20/20**
Moving objects	0/20	12/20	13/20	**20/20**
Occlusions	17/20	10/20	2/20	**18/20**
New objects	16/20	14/20	15/20	**18/20**

Table 3.3: Comparison of RLBC with 3 baselines on the real-world UR5-Bowl task with dynamic changes of the cube position, dynamic occlusions and new object instances.

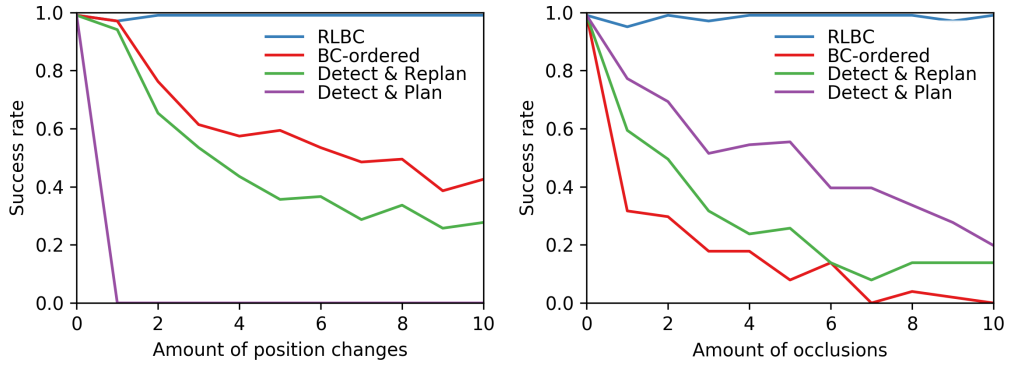
such as dynamic changes of object positions, dynamic occlusions, unseen object instances and the increased probability of collisions due to small distances between objects. We show that RLBC outperforms the baselines on those scenarios both in simulation and on a real robot. Note, that our real robot experiments are performed with skills and master policies that have been trained exclusively in simulation using sim2real augmentation [Pashevich, 2019]. We use the same policies for all perturbation scenarios. Qualitative results of our method are available in the **supplementary video**.

3.6.1 Baseline methods

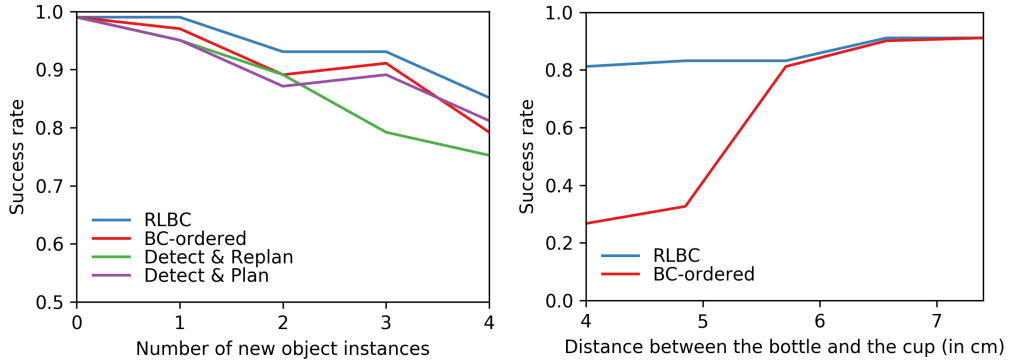
We compare RLBC with 3 baselines: (a) a fixed sequence of BC skills following the manually pre-defined correct order (BC-ordered); (b) an open-loop controller estimating positions of objects and executing an expert script (Detect & Plan); (c) a closed-loop controller performing the same estimation-based control and replanning in case if object positions change substantially (Detect & Replan). We use the same set of skills for RLBC and BC-ordered. We train the position estimation network using a dataset of 20.000 synthetic depth images with randomized object positions. All networks use ResNet-18 architecture and are trained with the standard, viewpoint and sim2real augmentations described in Section 3.5.2.

3.6.2 Results on UR5-Bowl with no perturbations

We first evaluate RLBC and the three baselines on the UR5-Bowl task (see Figure 3.2b). When tested in simulation, all the baselines and RLBC manage to perfectly solve the task. On the real-world UR5-Bowl task, BC-ordered and Detect & Plan baselines sometimes fail to grasp the object which leads to task failures (see Table 3.3, first row). On the contrary, RLBC solves the task in all 20 episodes given its ability to re-plan the task in the cases of failed skills.



(a) UR5-Bowl (b) UR5-Bowl



(c) UR5-Bowl (d) UR5-Breakfast

Figure 3.4: Performance of RLBC and baseline methods in simulated environments under perturbations: (a) Dynamic changes of cube position; (b) Dynamic occlusions; (c) Replacing the cube by unseen objects; (d) Decreasing the distance between objects.

We have also attempted to solve the simulated UR5-Bowl task without skills by learning an RL policy performing low-level control. We have used ImageNet pre-trained ResNet-18 to generate visual features. The features were then used to train low-level RL control policy with PPO. Whereas such a low-level RL policy did not solve the task a single time after 10^4 episodes, RLBC reaches 100% after 400 episodes.

3.6.3 Robustness to perturbations

Robustness to dynamic changes in object position. We evaluate RLBC against the baselines in the UR5-Bowl scenario where the cube is moved several times during the episode. We plot success rates evaluated in simulation with respect to the number of position changes in Figure 3.4a. We observe the stability of RLBC and the fast degradation of all baselines. As both RLBC and BC-ordered use the same set of skills, the stability of RLBC comes from the learned skill combination. The "Moving objects" row in Table 3.3 reports results for 3 moves of the cube evaluated on the real robot. Similar to simulated results, we observe excellent results of RLBC and the degraded performance for all the baselines.

Robustness to occlusions. We evaluate the success of UR5-Bowl task under occlusions. Each occlusion lasts 3 seconds and covers a large random part of the workspace by a cardboard. Figure 3.4b shows success rates with respect to the number of occlusions in the simulated UR5-Bowl environment. Similar to perturbation results in Figure 3.4a, RLBC demonstrates high robustness to occlusions while the performance of other methods quickly degrades. The "Occlusions" row in Table 3.3 reports results for a single occlusion performed during the real-robot evaluation. Baseline methods are strongly influenced by occlusions except Detect & Plan which performs well unless occlusion happens during the first frames. Our HRBC policy performs best compared to other methods.

Robustness to new object instances. We evaluate the robustness of methods to the substitution of a cube by other objects not seen during the training of UR5-Bowl task. Figure 3.4c shows the success rate of RLBC and other methods with respect to the number of new objects in simulation. The novel objects are ordered by their dissimilarity with the cube. The difficulty of grasping unseen objects degrades the performance of grasping skills. In contrast to other methods RLBC is able to automatically recover from errors by making several grasping attempts. Table 3.3 reports corresponding results on a real robot where the cube has been replaced by 10 unseen objects. Similar to other perturbations we observe superior performance

of RLBC.

Impact of the distance between objects. We vary the distance between a bottle and a cup in the UR5-Breakfast task. The smaller distance between objects A and B implies higher probability of collision between a robot and A when grasping B behind A. The choice of the grasping order becomes important in such situations. While our method is able to learn the appropriate grasping order to maximize the chance of completing the task, the BC-ordered and other baselines use pre-defined order. Figure 3.4d demonstrates the performance of RLBC and BC-ordered for different object distances in the simulated UR5-Breakfast task. As expected, RLBC learns the correct grasping order and avoids most of the collisions. The performance of BC-ordered strongly degrades with the decreasing distance. In the real-world evaluation, both RLBC and ordered skills succeed in 16 out of 20 episodes when the distance between objects is larger than 10 cm. However, the performance of BC-ordered drops to 8/20 when the cup and the bottle are at 4cm from each other. In contrast, RLBC chooses the appropriate object to avoid collisions and succeeds in 16 out of 20 trials.

3.7 Conclusion

This chapter presents a method to learn combinations of primitive skills. Our method requires no full-task demonstrations nor intermediate rewards and shows excellent results in simulation and on a real robot. We demonstrate the versatility of our approach in challenging settings with dynamic scene changes.

Chapter 4

Optimizing image augmentations for sim2real policy transfer

The previous chapter relies on a sim-to-real method to transfer policies learned on synthetic data to a real robot. In this method we present the sim-to-real method we developed and how we used a proxy pose estimation task to optimize our data augmentation and finally applied it to policy learning.

Vision and learning have made significant progress that could improve robotics policies for complex tasks and environments. Learning deep neural networks for image understanding, however, requires large amounts of domain-specific visual data. While collecting such data from real robots is possible, such an approach limits the scalability as learning policies typically requires thousands of trials.

In this chapter, we attempt to learn manipulation policies in simulated environments. Simulators enable scalability and provide access to the underlying world state during training. Policies learned in simulators, however, do not transfer well to real scenes given the domain gap between real and synthetic data. We follow recent work on domain randomization and augment synthetic images with sequences of random transformations. Our main contribution is to *optimize* the augmentation strategy for sim2real transfer and to enable domain-independent policy learning. We design an efficient search for depth image augmentations using object localization as a proxy task. Given the resulting sequence of random transformations, we use it to augment synthetic depth images during policy learning. Our augmentation strategy is policy-independent and enables policy learning with no real images. We demonstrate our approach to significantly improve accuracy on three manipulation tasks evaluated

on a real robot.

4.1 Introduction

In particular, recent progress in computer vision and deep learning motivates new methods combining learning-based vision and control. Successful methods in computer vision share similar neural network architectures, but learn *task-specific* visual representations, e.g. for object detection, image segmentation or human pose estimation. Guided by this experience, one can assume that successful integration of vision and control will require learning of policy-specific visual representations for particular classes of robotics tasks.

Learning visual representations requires large amounts of training data. Previous work has addressed policy learning for simple tasks using real robots e.g., in [Levine, 2016; Pinto, 2016b; Zhang, 2018c]. Given the large number of required attempts (e.g. 800,000 grasps collected in [Levine, 2016]), learning with real robots might be difficult to scale to more complex tasks and environments. On the other hand, physics simulators and graphics engines provide an attractive alternative due to the simple parallelization and scaling to multiple environments as well as due to access to the underlying world state during training.

Learning in simulators, however, comes at the cost of the reality gap. The difficulty of synthesizing realistic interactions and visual appearance typically induces biases and results in low performance of learned policies in real scenes. Among several approaches to address this problem, recent work proposes domain randomization [Tobin, 2017] by augmenting synthetic data with random transformations such as random object shapes and textures. While demonstrating encouraging results for transferring policies trained in simulations to real environments ("sim2real" transfer), the optimality and generality of proposed transformations remains open.

In this work we follow the domain randomization approach and propose to learn transformations optimizing sim2real transfer. Given two domains, our method finds policy-independent sequences of random transformations that can be used to learn multiple tasks. While domain randomization can be applied to different stages of a simulator, our goal here is the efficient learning of visual representations for manipulation tasks. We therefore learn parameters of random transformations to bridge the domain gap between synthetic and real images. We here investigate the transfer of policies learned for depth images. However, our method should generalize to RGB inputs. Examples of our synthetic and real depth images used to train and

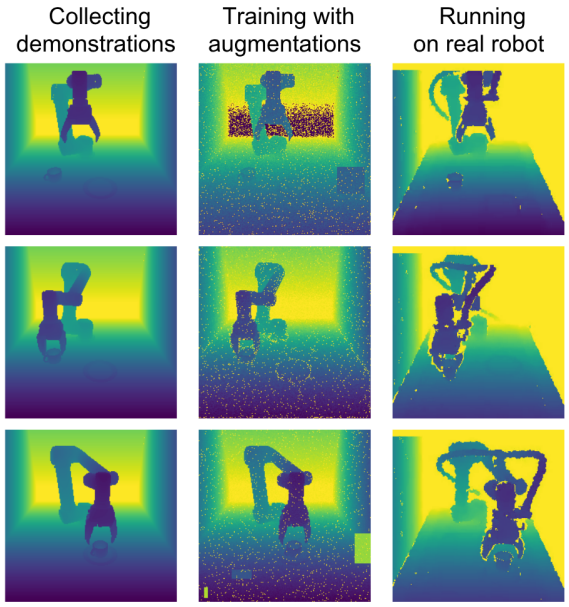


Figure 4.1: Example depth images for the task "Cup placing". Synthetic depth maps (first column) are augmented with random transformations during policy training (second column). The resulting policy is applied to depth maps from the real robot scene (third column).

test the "Cup placing" policy are illustrated in Fig. 4.1.

In more details, our method uses a proxy task of predicting object locations in a robot scene. We synthesize a large number of depth images with objects and train a CNN regressor estimating object locations after applying a given sequence of random transformations to synthetic images. We then score the parameters of current transformations by evaluating CNN location prediction on pre-recorded real images. Inspired by the recent success of AlphaGo [Silver, 2016] we adopt Monte-Carlo Tree Search (MCTS) as an efficient search strategy for transformation parameters.

We evaluate the optimized sequences of random transformations by applying them to simulator-based policy learning. We demonstrate the successful transfer of such policies to real robot scenes while using no real images for policy training. Our method is shown to generalize to multiple policies. The overview of our approach is illustrated in Fig. 4.2. The code of the project is publicly available at the project website[1].

[1] http://pascal.inrialpes.fr/data2/sim2real

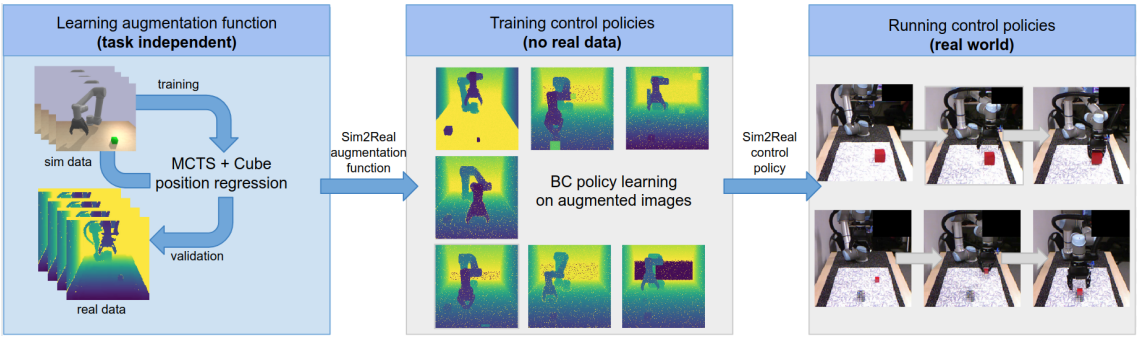


Figure 4.2: Overview of the method. Our contribution is the policy-independent learning of depth image augmentations (left). The resulting sequence of augmentations is applied to synthetic depth images while learning manipulation policies in a simulator (middle). The learned policies are directly applied to real robot scenes without finetuning on real images.

4.2 Related work

Robotics tasks have been addressed by various learning methods including imitation learning [Duan, 2017] and reinforcement learning [Riedmiller, 2018a]. Despite mastering complex simulated tasks such as Go [Silver, 2016], the addressed robotics tasks remain rather simple [Zhang, 2018c; Riedmiller, 2018a; Gu, 2016]. This difference is mainly caused by the real world training cost. Learning a manipulation task typically requires a large amount of data [Levine, 2016; Pinto, 2016b]. Thus, robot interaction time becomes much longer than in simulation [Gu, 2016] and expert guidance is non-trivial [Zhang, 2018c].

Learning control policies in simulation and transferring them to the real world is a potential solution to address these difficulties. However, the visual input in simulation is significantly different from the real world and therefore requires adaptation. Recent attempts to bridge the gap between simulated and real images can be generally divided into two categories: domain adaptation [Bousmalis, 2018] and domain randomization [Tobin, 2017]. Domain adaptation methods either map both image spaces into a common one [James, 2019; Müller, 2018] or map one into the other [Lee, 2019b]. Domain randomization methods add noise to the synthetic images [Pinto, 2018; Sadeghi, 2018], thus making the control policy robust to different textures and lighting. The second line of work is attractive due to its effectiveness and simplicity. Yet, it was so far only shown to work with RGB images. While depth images are well suited for many robotics tasks [Litvak, 2019], it is not obvious what type of randomization should be used in the case of depth data. Here, we explore a learning based approach to select appropriate transformations and show that this allows us to close the gap between simulated and real visual data.

Domain randomization is also referred to as data augmentation in the context of image classification and object detection. Data augmentation is known to be an important tool for training deep neural networks and in most cases it is based on a manually designed set of simple transformations such as mirroring, cropping and color perturbations. In general, designing an effective data augmentation pipeline requires domain-specific knowledge [Dvornik, 2018]. Depth images might be augmented by adding random noise [Handa, 2015], noise patterns typical for real sensors [Eitel, 2015] or by compensating missing information [N Yang, 2012]. Learning to augment is a scalable and promising direction that has been explored for visual recognition in [Paulin, 2014]. Recent attempts to automatically find the best augmentation functions propose to use reinforcement learning and require several hundreds of GPUs [Cubuk, 2019]. Given the prohibitive cost of executing thousands of policies in a real-robot training loop, we propose to optimize sequences of augmentations within a proxy task by predicting object locations in pre-recorded real images using the Monte-Carlo tree search. A related idea of learning the rendering parameters of a simulator has been recently proposed for a different task of semantic image segmentation in [Ruiz, 2019].

4.3 Approach

We describe the proposed method for learning depth image augmentations in Sections 4.3.2 and 4.3.3. Our method builds on Behaviour Cloning (BC) policy learning [Pomerleau, 1988; Ross, 2011] which we overview in Section 4.3.1.

4.3.1 Behaviour cloning in simulation

Given a dataset $\mathcal{D}^{\text{expert}} = \{(o_t, a_t)\}$ of observation-action pairs along with the expert trajectories in *simulation*, we learn a function approximating the conditional distribution of the expert policy $\pi_{\text{expert}}(a_t|o_t)$ controlling a robotic arm. Here, the observation is a sequence of the three last depth frames, $o_t = (I_{t-2}, I_{t-1}, I_t) \in \mathcal{O} = \mathbb{R}^{H\times W\times 3}$. The action $a_t \in \mathcal{A} = \mathbb{R}^7$ is the robot command controlling the end-effector state. The action $a_t = (\mathbf{v}_t, \boldsymbol{\omega}_t, g_t)$ is composed of the end-effector linear velocity $\mathbf{v}_t \in \mathbb{R}^3$, end-effector angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^3$ and the gripper openness state $g_t \in \{0, 1\}$. We learn the deterministic control policy $\pi : \mathcal{O} \to \mathcal{A}$ approximating the expert policy π_{expert}. We define π by a Convolutional Neural Network (CNN) parameterized by a set of weights θ and learned by minimizing the L_2 loss for the velocity controls $(\mathbf{v}_t, \boldsymbol{\omega}_t)$ and the cross-entropy loss L_{CE} for the binary grasping signal g_t. Given the state-action pair (s_t, a_t) and the network prediction $\pi_\theta(s_t) = (\hat{\mathbf{v}}_t, \hat{\boldsymbol{\omega}}_t, \hat{g}_t)$,

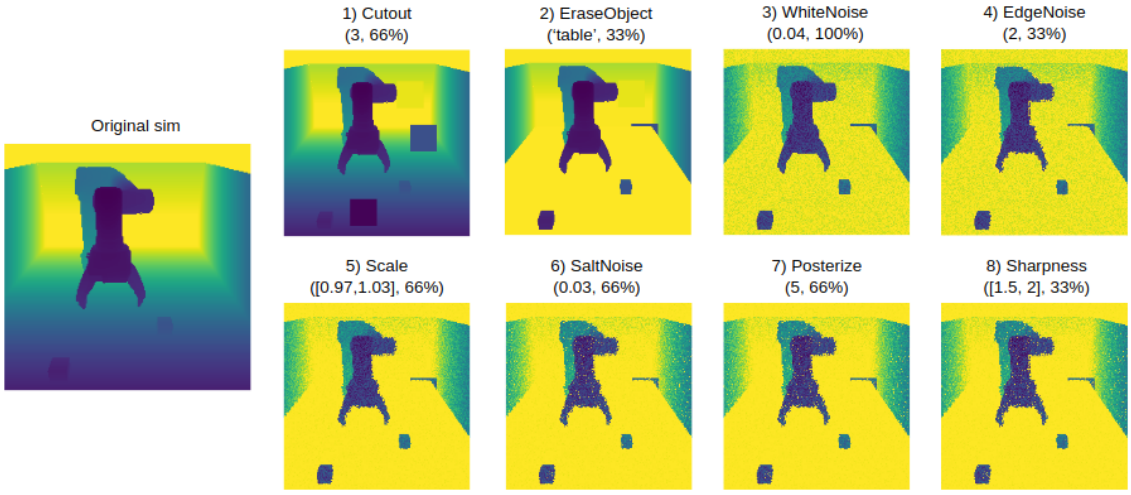


Figure 4.3: The original synthetic depth image on the left is augmented by the sequence of eight random transformations learned by our method.

we minimize the loss:

$$\lambda L_2\left([\hat{\mathbf{v}}_\mathbf{t}, \hat{\boldsymbol{\omega}}_t], [\mathbf{v}_t, \boldsymbol{\omega}_t]\right) + (1-\lambda) L_{\text{CE}}\left(\hat{g}_t, g_t\right), \tag{4.1}$$

where $\lambda \in [0, 1]$ is a scaling factor of the cross-entropy loss which we experimentally set to 0.9.

4.3.2 Sim2Real transfer

Given a stochastic augmentation function f, we train a CNN h to predict the cube position on a simulation dataset $\mathcal{D}^{\text{sim}} = \left\{(I_i^{\text{sim}}, p_i^{\text{sim}})\right\}$. Given an image I_i^{sim}, the function f sequentially applies N primitive transformations, each with a certain probability. This allows for bigger variability during the training. We minimize the L_2 loss between the cube position p_i^{sim} and the network prediction given an augmented depth image $h\left(f(I_i^{\text{sim}})\right)$:

$$\sum_k \mathbb{E}\, L_2\left(h\left(f(I_k^{\text{sim}})\right), p_k^{\text{sim}}\right). \tag{4.2}$$

We evaluate augmentation functions by computing the average error of network prediction on a pre-recorded real-world dataset, $\mathcal{D}^{\text{real}} = \left\{(I_i^{\text{real}}, p_i^{\text{real}})\right\}$ as

$$e^{real} = \frac{1}{n}\sum_{k=1}^{n} L_2\left(h(I_k^{\text{real}}), p_k^{\text{real}}\right). \tag{4.3}$$

The optimal augmentation function f^* should result in a network h with the smallest real-world error e^{real}. We assume that the same augmentation function will produce optimal control policies. We re-apply the learned stochastic function f^* on individual frames of $\mathcal{D}^{\text{expert}}$ at every training epoch and learn π^{sim2real}. We use π^{sim2real} to control the real robot using the same control actions as in the simulation, i.e., $a_t^{\text{real}} = (\mathbf{v}_t^{\text{real}}, \boldsymbol{\omega}_t^{\text{real}}, g_t^{\text{real}}) = \pi^{\text{sim2real}}(I_t^{\text{real}})$, see Fig. 4.2.

4.3.3 Augmentation space

We discretize the search space of augmentation functions by considering sequences of N transformations from a predefined set. We then apply the selected sequence of transformations in a given order to each image. The predefined set of transformations consists of the depth-applicable standard transformations from PIL, a popular Python Image Library, as well as Cutout [Devries, 2017], white (uniform) noise and salt (Bernoulli) noise. We also take advantage of segmentation masks provided by the simulator and define two object-aware transformations, i.e., boundary noise and object erasing (see Section 4.4.2 for details). The identity (void) transformation is included in the set to enable the possibility of reducing N. The full set of our eleven transformations is listed in Table 4.1. Each transformation is associated with a magnitude and a probability of its activation. The magnitude defines the transformation-specific parameter. For each transformation we define two possible magnitudes and three probabilities. With $N = 8$ in our experiments, our search space roughly includes $(11 \times 2 \times 3)^8 \approx 3.6 * 10^{14}$ augmentation functions. We reduce the search space by restricting each transformation, except identity, to occur only once in any augmentation sequence.

4.3.4 Real robot control

Algorithm 2 Sim2Real policy transfer algorithm

*** Given datasets $\mathcal{D}^{\text{sim}}, \mathcal{D}^{\text{real}}, \mathcal{D}^{\text{expert}}_{\text{sim}}$ ***
MCTS = init_mcts()
repeat
    f = MCTS.sample_path()
    CNN = train_cube_prediction($f, \mathcal{D}^{\text{sim}}$) ▷ Eq.4.2
    e^{real} = compute_error(CNN, $\mathcal{D}^{\text{real}}$) ▷ Eq.4.3
    MCTS.update(e^{real}) ▷ Backpropagate the error
until the smallest e^{real} is constant for 500 iterations
f^* = MCTS.select_best_path()
π^{sim2real} = train_BC_policy($f^*, \mathcal{D}^{\text{expert}}_{\text{sim}}$) ▷ Eq.4.1
return π^{sim2real}

To find an optimal sequence of augmentations, we use Monte Carlo Tree Search (MCTS) [Coulom, 2006] which is a heuristic tree search algorithm with a trade-off between exploration and exploitation. Our search procedure is defined in Algorithm 2. The algorithm iteratively explores the Monte Carlo tree (lines 3-8) by sampling sequences of transformations (line 4), training a cube position prediction network on an augmented simulation dataset (line 5), evaluating the trained network on the real dataset (line 6) and backpropagating the evaluation error through the Monte Carlo tree (line 7). Once the smallest error on the real dataset stays constant for 500 iterations, we choose the best augmentation function according to MCTS (line 9) and train sim2real control policies using the simulation dataset of augmented expert trajectories on the tasks of interest (line 10). Once trained, the control policies can be directly applied in real robot scenes without fine-tuning. The sequence of eight transformations found by MCTS is illustrated in Fig. 4.3.

4.4 Experimental evaluation

This section evaluates the transfer of robot control policies from simulation to real. First, we describe our tasks and the experimental setup in Section 4.4.1. We evaluate independently each of predefined basic transformations on the cube position prediction task in Section 4.4.2. In Section 4.4.3, we compare our approach of learning augmentation functions with baselines. Finally, we demonstrate the policy transfer to the real-world robotics tasks in Section 4.4.4.

4.4.1 Experimental setup

Our goal is to learn a policy to control a UR5 6-DoF robotic arm with a 3 finger Robotiq gripper for solving manipulation tasks in the real world. The policy takes as input 3 depth images $o_t \in \mathbb{R}^{H \times W \times 3}$ from the Kinect-1 camera positioned in front of the arm. We scale the values of depth images to the range $[0, 1]$. The policy controls the robot with an action $a_t \in \mathbb{R}^7$. The control is performed at a frequency of 10 Hz. All the objects and the robotic gripper end-effector are initially allocated within the area of $60 \times 60\,\mathrm{cm}^2$ in front of the arm. The simulation environment is built with the `pybullet` physics simulator [Courmans, 2016] and imitates the real world setup. We consider three manipulation tasks. **Cube picking task:** The goal of the task is to pick up a cube of size 4.7 cm and to lift it. In simulation, the cube size is randomized between 3 and 9 cm. **Cubes stacking task:** The goal of the task is to stack a cube of size 3.5 cm on top of a cube of size 4.7 cm. We randomize the sizes of cubes in simulation between 3 and 9 cm. **Cup placing task:** The goal of the task

is to pick up a cup and to place it on a plate. In simulation, we randomly sample 43 plates from ModelNet [Wu, 2015] and 134 cups from ShapeNet [Chang, 2015b]. We use three cups and three plates of different shapes in our real robot experiments.

Transformation	Error in sim	Error in real
Identity	0.63 ± 0.50	6.52 ± 5.04
Affine	**0.59 ± 0.45**	4.83 ± 4.42
Cutout	1.19 ± 0.87	**1.86 ± 2.45**
Invert	0.88 ± 0.60	4.63 ± 3.28
Posterize	0.66 ± 0.48	5.54 ± 4.59
Scale	0.67 ± 0.47	6.00 ± 4.37
Sharpness	0.83 ± 0.49	5.48 ± 3.84
WhiteNoise	0.68 ± 0.54	3.60 ± 2.33
SaltNoise	0.72 ± 0.50	2.42 ± 1.16
BoundaryNoise	0.88 ± 0.66	2.06 ± 1.17
EraseObject	0.64 ± 0.47	1.93 ± 1.01

Table 4.1: Cube prediction error (in cm) for synthetic and real depth images evaluated separately for each of eleven transformations considered in this paper. The errors are averaged over 200 pairs of images and cube positions.

	Augmentation	Error in sim	Error in real
i	None	**0.63 ± 0.50**	6.52 ± 5.04
ii	Random (8 operations)	6.56 ± 4.05	5.77 ± 3.12
iii	Handcrafted (4 operations)	0.99 ± 0.68	2.35 ± 1.36
iv	Learned (1 operation)	1.19 ± 0.87	1.86 ± 2.45
v	Learned (4 operations)	1.21 ± 0.78	1.17 ± 0.71
vi	Learned (8 operations)	1.31 ± 0.90	**1.09 ± 0.73**

Table 4.2: Cube prediction error (in cm) on synthetic and real depth images using different types of depth data augmentation. More augmentations increase the error for synthetic images and decrease the error for real images as expected from our optimization procedure. The errors are averaged over 200 pairs of images and cube positions.

4.4.2 Evaluation of individual transformations

Before learning complex augmentation functions, we independently evaluate each transformation from the set of transformations defined in Section 4.3.3. In this section, we describe each transformation and the associated values of magnitude. **Affine** randomly translates and rotates the image in the range of $[-9, 9]$ pixels and $[-5, 5]$ degrees, or alternatively by $[-16, 16]$ pixels and $[-10, 10]$ degrees. **Cutout** [Devries, 2017] samples one or three random rectangles in the image and

sets their values to a random constant in $[0, 1]$. **Invert** function inverts each pixel value by applying the operation $x \mapsto 1 - x$ and does not have any parameters. **Posterize** reduces the number of bits for each pixel value to be either 5 or 7. **Scale** randomly multiplies the image with a constant in one of the two ranges: $[0.95, 1.05]$ or $[0.97, 1.03]$. **Sharpness** increases the image sharpness either randomly in the range between 50% and 100% or by 100%. **WhiteNoise** adds uniform noise to each pixel with a magnitude of 0.04 or 0.08. **SaltNoise** sets each pixel value to 1 with the probability 0.01 or 0.03. **BoundaryNoise** uses the semantics mask of the simulator and removes patches of pixels located at the boundary between different objects. For BoundaryNoise, we remove either 2 or 4 pixels along the boundaries. **EraseObject** removes either the table or the walls behind the robot using the semantics segmentation mask. All the above transformations are associated with a probability in the set $\{33\%, 66\%, 100\%\}$. For instance, the probability 33% means that the transformation is applied 1 out of 3 times and 2 our of 3 times it acts as the identity function.

As explained in Section 4.3.2, we evaluate each augmentation function by computing the prediction error of the cube position in real depth images after training the position regressor on simulated data. We collect 2000 pairs of simulated depth images and cube positions for training and 200 real depth images for evaluation. To be robust to viewpoint changes in real scenes, each simulated scene is recorded from five random viewpoints. We randomize the camera viewpoint in simulation around the frontal viewpoint by sampling the camera yaw angle in $[-15, 15]$ degrees, pitch angle in $[15, 30]$ degrees, and distance to the robot base in $[1.35, 1.50]$ m. We only require the viewpoint of the real dataset to be within the simulated viewpoints distribution. Moreover, we allow to move the camera between different real robot experiments. We treat each transformation in the given set as a separate augmentation function (sequence of length 1). We use each of them independently to augment the simulation dataset. Next, we train a CNN based on ResNet-18 architecture [He, 2016a] to predict the cube position given a depth image. During the network training, we compute the prediction error (4.3) on two validation datasets, 200 simulated images and 200 real images. To evaluate each augmentation function more robustly, we always start the CNN training at the same initial position. For each transformation, we report the cube position prediction error in Table 4.1. With no data augmentation, the trained network performs well in simulation (error of 0.63 cm) but works poorly on real images (error of 6.52 cm). Cutout transformation reduces the regression error by more than 4 cm and indicates that it should be potentially combined with other transformations.

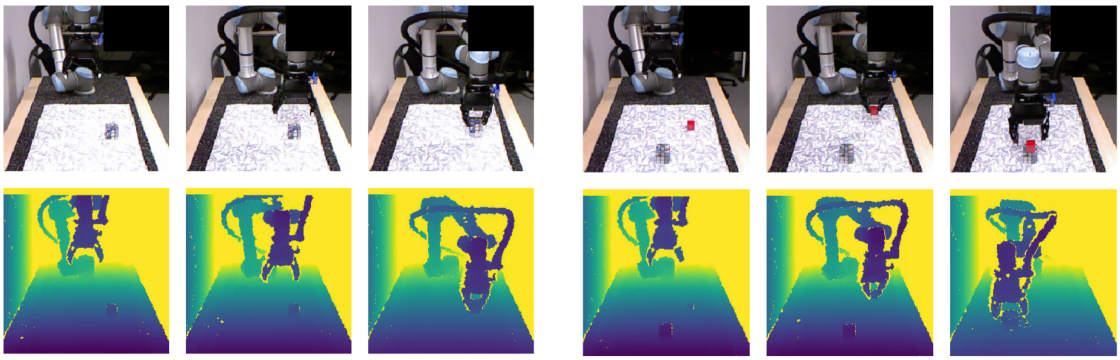
4.4.3 Augmentation function learning

We compare the augmentation function learned by our approach to several baselines on the task of estimating the cube position in Table 4.2. The baselines include training the network on synthetic images (i) without any data augmentation, (ii) with augmentation sequences composed of 8 random transformations (average over 10 random sequences), (iii) with a handcrafted augmentation sequence of four transformations built according to our initial intuition: Scale, WhiteNoise, EraseObjects and SaltNoise, (iv) with the best single transformation from Section 4.4.2 and (v) with the learned augmentation composed of 4 transformations. To have a robust score and exclude outliers, we compute the median error over evaluations of 10 training epochs.

Baselines (i)-(iii) with no augmentation learning demonstrate worst results on the real dataset. Results for learned transformations (iv)-(vi) show that more transformations with different probabilities help to improve the domain transfer. Table 4.2 also demonstrates the trade-off between the performance in different domains: the better augmentation works on the real dataset, the worse it performs in the simulation. Effectively, the learned augmentation shifts the distribution of simulated images towards the distribution of real images. As a consequence, the network performs well on the real images that are close to the training set distribution and works worse on the original simulated images that lie outside the training set distribution. The best augmentation sequence found by our method is illustrated in Fig. 4.3 and contains the following transformations: Cutout, EraseObject, WhiteNoise, EdgeNoise, Scale, SaltNoise, Posterize, Sharpness. Learning an augmentation function of length 8 takes approximately 12 hours on 16 GPUs. The vast majority of this time is used to train the position estimation network while MCTS path sampling, evaluation and MCTS backpropagation are computationally cheap. We iteratively repeat the training and evaluation routine until the error does not decrease for a sufficiently long time (500 iterations). Once the sim2real augmentation is found, it takes approximately an hour to train the BC control policy.

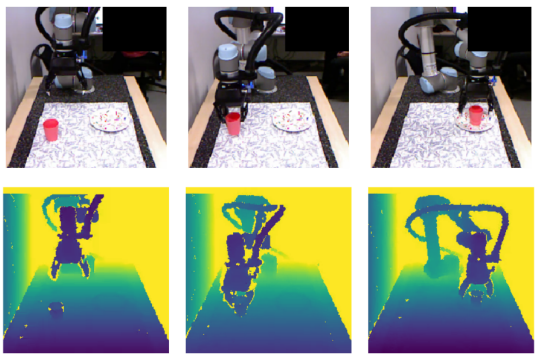
4.4.4 Real robot control

In this section we demonstrate that the data augmentation learned for a proxy task transfers to other robotic control tasks. We collect expert demonstrations where the full state of the system is known, and an expert script can easily be generated at training time. We augment the simulated demonstrations with the learned data augmentation and train BC policies without any real images. Moreover, we show

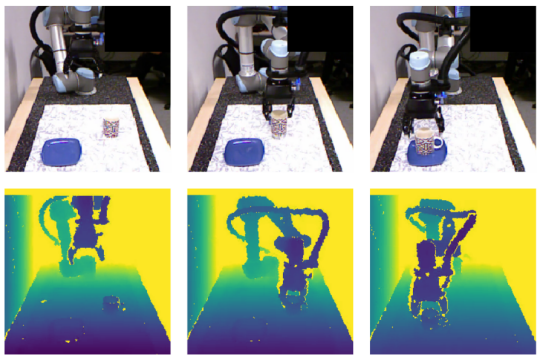


(a) Cube picking

(b) Cube stacking



(c) Cup placing



(d) Cup placing

Figure 4.4: Frame sequences of real world tasks performed with a policy learned on a simulation dataset augmented with our approach. The tasks are: a) picking up a cube, b) stacking two cubes, c) placing a cup on top of a plate, d) placing another cup on top of another plate.

that our augmentation is not object specific and transfers to tasks with new object instances not present in the set of expert demonstrations. For each task, we compare the learned augmentation function with 3 baselines: no augmentation, handcrafted augmentation and best single transformation. Each evaluation consists of 20 trials with random initial configurations. The results are reported in Table 4.3.

Cube picking task. Success rates for policies learned with different augmentation functions are strongly correlated with results for cube position estimation in Table 4.2. The policy without augmentation has a success rate 3/20. Single transformation and handcrafted augmentation have 9/20 and 8/20 successful trials respectively. The sim2real policy learned with our method succeeds 19 out of 20 times.

Cube stacking task. Given a more difficult task where more precision is required, the baseline approaches perform poorly and achieve the success rate of only 2/20 for the handcrafted augmentation. We observe most of the failure cases due to impre-

Augmentation	Pick	Stack	Cup Placing
None	3/20	1/20	0/20
Handcrafted (4 operations)	9/20	2/20	6/20
Learned (1 operation)	8/20	1/20	1/20
Learned (8 operations)	**19/20**	**18/20**	**15/20**

Table 4.3: Success rates for control policies executed on a real robot (20 trials per experiment). Results are shown for three tasks and alternative depth image augmentations.

cise grasping and stacking. We successfully tested the learned data augmentation function on cubes of varying sizes which indicates high control precision. Overall, our method was able to stack cubes in 18 runs out of 20.

Cup placing. Solving the Cup placing task requires both precision and the generalization to previously unseen object instances. The policies are trained over a distribution of 3D meshes and thus leverage the large dataset available in the simulation. All baselines fail to solve the Cup placing except for the handcrafted augmentation which succeeds 6 times out of 20. Our approach is able to solve the task with the success rate of 15/20 despite the presence of three different instances of cups and plates never seen during training. These results confirm our hypothesis that the augmentations learned for a proxy task of predicting the cube position, generalize to new objects and tasks.

4.5 Conclusion

In this chapter, we introduce a method to learn augmentation functions for sim2real policy transfer. To evaluate the transfer, we propose a proxy task of object position estimation that requires only a small amount of real world data. Our evaluation of data augmentation shows significant improvement over the baselines. We also show that the performance on the proxy task strongly correlates with the final policy success rate. Our method does not require any real images for policy learning and can be applied to various manipulation tasks. We apply our approach to solve three real world tasks including the task of manipulating previously unseen objects.

Chapter 5

Learning obstacle representations for neural motion planning

The two previous chapters focused on learning manipulation tasks in environments where there are no obstacles to avoid and no clutter on the table. Ideally we would like to scale these methods to more complex manipulation environments where with clutter both motion planning and manipulation of objects is needed to solve tasks. In this chapter we specifically focus on vision-based motion planning and propose a method leveraging knowledge of the surrounding environment and able to adapt to dynamically changing environments.

Motion planning and obstacle avoidance is a key challenge in robotics applications. While previous work succeeds to provide excellent solutions for known environments, sensor-based motion planning in new and dynamic environments remains difficult. In this chapter we address sensor-based motion planning from a learning perspective. Motivated by recent advances in visual recognition, we argue the importance of learning appropriate representations for motion planning. We propose a new obstacle representation based on the PointNet architecture [Qi, 2017] and train it jointly with policies for obstacle avoidance. We experimentally evaluate our approach for rigid body motion planning in challenging environments and demonstrate significant improvements of the state of the art in terms of accuracy and efficiency.

5.1 Introduction

Motion planning is a fundamental robotics problem [Latombe, 1991; LaValle, 2006] with numerous applications in mobile robot navigation [Fox, 1997], industrial robotics

[Laumond, 2006], humanoid robotics [Harada, 2014] and other domains. Sampling-based methods such as Rapidly Exploring Random Trees (RRT) [Jr, 2000] and Probabilistic Roadmaps (PRM) [Kavraki, 1996] have been shown successful for finding a collision-free path in complex environments with many obstacles. Such methods are able to solve the so-called piano mover problem [Schwartz, 1986] and typically assume static environments and prior knowledge about the shape and location of obstacles. In many practical applications, however, it is often difficult or even impossible to obtain detailed a-priori knowledge about the real state of environments. It is therefore desirable to design methods relying on partial observations obtained from sensor measurements and enabling motion planning in unknown and possibly dynamic environments. Moreover, given the high complexity devoted to exploration in sampling-based methods, it is also desirable to design more efficient methods that use prior experience to quickly find solutions for motion planning in new environments.

To address the above challenges, several works [Glasius, 1995; Yang, 2000; Pfeiffer, 2017; Ichter, 2018; Jurgenson, 2019; Qureshi, 2019] adopt neural networks to learn motion planning from previous observations. Such Neural Motion Planning (NMP) methods either improve the exploration strategies of sampling-based approaches [Ichter, 2018] or learn motion policies with imitation learning [Pfeiffer, 2017; Qureshi, 2019] and reinforcement learning [Jurgenson, 2019]. In this work we follow the NMP paradigm and propose a new learnable obstacle representation for motion planning.

Motivated by recent advances in visual recognition [Qi, 2017; Krizhevsky, 2012; He, 2016b], we argue that the design and learning of obstacle representations plays a key role for the success of learning-based motion planning.

In this work, we proposed a new representation based on point clouds which are first sampled from visible surfaces of obstacles and then encoded with a PointNet [Qi, 2017] neural network architecture, see Fig. 5.1. We learn our obstacle representation jointly with the motion planing policies in the SAC (Soft Actor Critic) reinforcement learning framework [Haarnoja, 2018b]. In particular, while using environments composed of 3D-box shapes during training, we demonstrate the generalization of our motion planning policies to complex environments with objects of previously unseen shapes. We also show our method to seamlessly generalize to new object constellations and dynamic scenes with moving obstacles. We evaluate our method in challenging simulated environments and validate our representation through ablation studies and comparison to the state of the art. Our method significantly improves

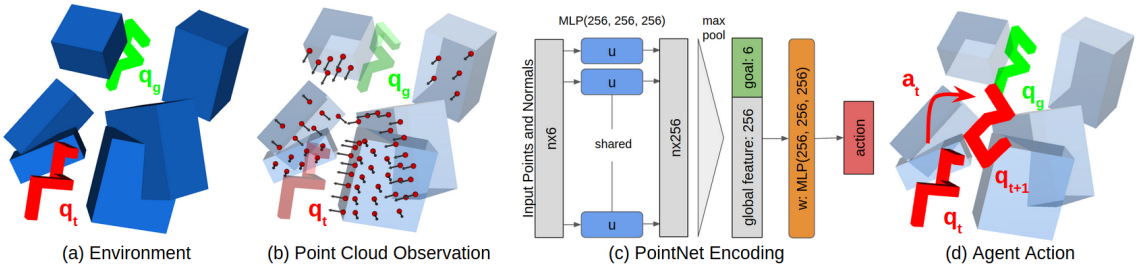


Figure 5.1: Overview of our approach. (a) We aim to find a collision-free path for a rigid body from its current configuration q_t to the goal configuration q_g. (b) We assume no prior knowledge about the scene and represent obstacles by points and normals sampled on object surfaces. (c) Our neural network learns the PointNet encoding of observed points and normals together with the motion policy. (d) The learned network generates actions that move the body towards the goal configuration along a collision-free path.

the accuracy of previous NMP methods while being one order of magnitude more computationally efficient compared to close competitors.

The contributions of our work are threefold. First, we propose a new learnable obstacle representation based on point clouds and the PointNet neural architecture [Qi, 2017]. Second, we learn our representation jointly with policies for rigid body motion planning using reinforcement learning. Finally, we experimentally evaluate our approach and demonstrate significant improvements of the state of the art in motion planning in terms of accuracy and efficiency. Code and qualitative results along with a video are available on the project webpage [Strudel,].

5.2 Related work

Sampling-based motion planning. Sampling-based methods such as RRT [Jr, 2000] have been extensively studied for motion planning [Latombe, 1991; Latombe, LaValle, 2006; Jr, 2000; Kavraki, 1996]. Such methods can deal with complex environments, however, they typically assume a complete knowledge of the robot workspace and static obstacles. Given the complexity of sampling-based methods, recent work proposes efficient exploration schemes [Jetchev, 2010; Lien, 2010; Branicky, 2008; Martin, 2007], e.g., by reusing previously discovered paths in similar scenarios [Branicky, 2008] or biasing RRT search to promising regions [Martin, 2007]. Our work does not assume any a-priori knowledge about the environment and can deal with moving obstacles. Moreover, while our method performs an extensive exploration during training, the learned policies directly generate feasible paths during testing in new environments.

Neural motion planning. Learning-based methods for motion planning have been introduced in [Glasius, 1995; Yang, 2000]. Motivated by the success of deep learning, a number of more recent methods explore neural networks for motion planning and obstacle avoidance. [Ichter, 2018] learns to sample robot configurations close to the target RRT solutions. [Qureshi, 2019] and [Pfeiffer, 2017] learn a policy with imitation learning using RRT solutions as demonstrations. While improving the efficiency of RRT, [Pfeiffer, 2017; Ichter, 2018; Qureshi, 2019] still require sampling at test time and are not easily applicable in scenes with moving obstacles. Our work is most related to [Jurgenson, 2019] who explore reinforcement learning (RL) to learn motion policies. Similar to [Jurgenson, 2019] we use RL and learn to avoid obstacles using a negative collision reward. While [Jurgenson, 2019] presents results in relatively simple 2D environments, we propose a new learnable obstacle representation that generalizes to complex 3D scenes. We experimentally compare our method to [Ichter, 2018; Jurgenson, 2019; Qureshi, 2019] and demonstrate improved accuracy.

Visual Representation. The NMP methods [Jurgenson, 2019; Qureshi, 2019; Ichter, 2018] assume full knowledge of the workspace and use an obstacle encoding either based on a 2D image of obstacles encoded with a convolutional neural network (CNN) [Jurgenson, 2019], or an occupancy grid [Ichter, 2018] or a volumetric point cloud [Qureshi, 2019] encoded with a multi-layer perceptron (MLP). We show that obstacles representation is critical to solve complex problems with rich workspace variations and propose to rely on a point cloud representation of obstacles coupled with PointNet [Qi, 2017]. [Qi, 2017] demonstrated the performance of PointNet to classify and segment point clouds, in this work we propose to use it to encode obstacles.

5.3 Method

5.3.1 Overview of the method

In this work we consider rigid robots with 2 to 6 degrees of freedom (DoF). Let $\mathcal{W} \subseteq \mathbb{R}^3$ be the workspace of the robot, containing a set of obstacles $\mathcal{V}$. We denote $\mathcal{C}_{free}$ the open set of configurations where the robot does not collide with the obstacles $\mathcal{V}$ and $\mathcal{C}_{collision} = \mathcal{C} \setminus \mathcal{C}_{free}$. Given a start configuration $q_0 \in \mathcal{C}_{free}$ and a goal configuration $q_g \in \mathcal{C}_{free}$, motion planning aims at finding a *collision-free* path which connects the start configuration to the goal configuration. A continuous function $\tau : [0, 1] \to \mathcal{C}$ is a solution if $\tau([0, 1])$ is a subset of $\mathcal{C}_{free}$, $\tau(0) = q_0$ and $\tau(1) = q_g$. A motion

planning problem is thus defined by a start configuration, a goal configuration and a set of obstacles $\mathcal{V}$.

In this work, we represent obstacles $\mathcal{V}$ by sets of points (point clouds) and the corresponding normals sampled on the surface of obstacles. This representation is valid for arbitrary shapes and can be obtained with a depth sensor. We consider the robot as an embodied agent which senses surrounding obstacles with a panoramic camera (Fig.5.1a). The point cloud, expressed in the robot local coordinate frame (Fig.5.1b), is then processed with a PointNet architecture [Qi, 2017] to encode the obstacles (Fig.5.1c). The goal vector q_g is expressed in the robot local coordinate frame and is concatenated to the PointNet obstacle encoding. This vector is processed by a MLP that generates actions, bringing the robot closer to the goal while avoiding obstacles (Fig.5.1d). We learn the policy jointly with the PointNet encoding of the obstacles (Fig.5.1c) in an end-to-end fashion with reinforcement learning. In the next two sections, we first give details on the obstacle representation and then describe policy learning.

5.3.2 Obstacle representation for motion planning

We aim to learn a function that encodes the obstacles and the goal configuration as a vector enabling subsequent motion planning. While many parametric functions could be used as an encoder, we follow advances in visual recognition [Qi, 2017; Krizhevsky, 2012; He, 2016b] and define obstacle representations by a neural network learned jointly with the task of motion planning. We experimentally demonstrate the significant impact of the encoder on the performance of motion planning in Section 5.4.

In previous works [Ichter, 2018; Jurgenson, 2019; Qureshi, 2019], obstacles have been represented by occupancy grids encoded with a MLP [Ichter, 2018; Qureshi, 2019] or images encoded with a CNN [Jurgenson, 2019] assuming global workspace knowledge. In our work we use points sampled on the surface of obstacles along with their oriented normals. Such measurements can be obtained using a depth sensor either placed on the robot or at other locations. A set of obstacles $\mathcal{V}$ is represented by a finite set $S_{normals} = \{(x_i, n_i)\}_{i=1,\dots,N} \in \mathbb{R}^{N \times 2d}$ where $d = 2, 3$ and the points x_i and normals n_i are expressed in the robot local coordinate frame. We denote by α_i the couple (x_i, n_i). We define the goal g as the displacement to reach the goal configuration from the current robot configuration.

To process a point cloud, we use a PointNet [Qi, 2017] like network (Fig.5.1c) reduced to its core layers for computation efficiency which we describe below. The

idea of PointNet is to use a function which is symmetric with respect to the input to get a model invariant to input permutations. Based on this idea we build a network composed of two MLPs u and w with Exponential Linear Units (ELUs) activation [Clevert, 2016] as shown in Fig.5.1c. The first MLP u is shared across point cloud elements α_i (Fig.5.1c blue block) and is used to generate local features of each element $u(\alpha_i)$. Then, a max operation is applied to obtain a global feature v encoding the point cloud. The global feature v is further concatenated with the goal g (Fig.5.1c green block) and passed through a second MLP w (Fig.5.1c orange block):

$$\begin{aligned} v(\alpha_1, ..., \alpha_N) &= \max_{i=1,...,N} [u(\alpha_1), ..., u(\alpha_N)] \\ f(\alpha_1, ..., \alpha_N, g) &= w(v(\alpha_1, ..., \alpha_N), g) \end{aligned} \tag{5.1}$$

The max operator is a max-pooling operation: given a list of feature vectors of size $N \times d$, it outputs a vector of size d by taking the maximum over the N points in each coordinate.

The PointNet encoding of obstacles u is trained jointly with the policy w. For the training of u and w we compare imitation learning and reinforcement learning schemes as described in the next section.

5.3.3 Learning policies for motion planning

We cast the motion planning problem as a Markov decision process (MDP) [Sutton, 2018]. The state space $\mathcal{S}$ is the configuration space of the robot $\mathcal{C}$, $\mathcal{A}$ is the space of valid velocities at the current configuration and $\mathcal{O}$ is a representation of the workspace along with the goal configuration g. Given a robot configuration $q \in \mathcal{C}_{free}$ and v an admissible velocity vector, we denote $q(v)$ as the configuration reached by applying the velocity vector v to the robot for a fixed time. As the robot can hit into obstacles, we consider $q_{free}(v)$ which returns the last collision free configuration on the path connecting q to $q(v)$. Then the dynamics p is defined as $p(q, v) = q_{free}(v)$.

We aim at learning policies to solve motion planning problems. For that purpose, we explore and compare policies trained with imitation learning (behavioral cloning) and reinforcement learning. To train a policy with imitation learning [Schaal, 2003], we collect a dataset $\mathcal{D} = \{(o_t, a_t)\}$ of observation-action pairs along expert trajectories generated with Bi-RRT [Jr, 2000] and follow the behavioral cloning approach [Pomerleau, 1988]. Given a learnable policy π, we minimize the L^2 loss, $\mathcal{L}(\pi) = \|a_t - \pi(o_t)\|_2$, between the expert action a_t and the policy applied to the expert observation o_t.

To train a policy with reinforcement learning [Sutton, 2018], we define a reward function as follows. Given a goal configuration $g \in \mathcal{C}$, we define $r_{velocity}(q, v) = -\|v\|_2$ and

$$r_{task}(q, v, g) = \begin{cases} r_{goal} & \text{if } \|q_{free}(v) - g\| \leq \varepsilon, \\ r_{free} & \text{if } [q, q(v)] \subset \mathcal{C}_{free}, \\ r_{collision} & \text{else.} \end{cases} \tag{5.2}$$

with $r_{goal} > 0$, $r_{free} < 0$ and $r_{collision} < 0$. The reward function is then defined as $r(q, v, g) = r_{velocity}(q, v) + r_{task}(q, v, g)$. r_{task} rewards actions reaching g and penalizes actions which lead to a collision. Given two collision-free paths leading to the goal, the total reward $r(q, v, g)$ is highest for the shortest path. Maximizing the reward enables the policy path to be collision free and as short as possible. Note that the dynamics p depends only on the robot and the workspace, and the reward function r depends additionally on the goal configuration to be reached. An episode is terminated when reaching the goal or the maximum number of steps.

The reward r_{task} defined above is sparse with respect to the goal: it is only positive if the agent reaches the goal during one episode, which may have a low probability in challenging environments. Hindsight Experience Replay (HER) [Andrychowicz, 2017b] is a technique to improve the sample efficiency of off-policy RL algorithms in the sparse and goal-conditioned setting which we use extensively in this work. After collecting one rollout $s_0, ..., s_T$ which may or may not have reached the goal g, it consists in sampling one of the states as the new goal g' for this rollout. The rollout may not have reached g but in hindsight, it can be considered as a successful rollout to reach g'. HER can be seen as a form of implicit curriculum learning which accelerates the learning of goal-conditioned policies.

5.4 Experimental evaluation

Below we describe our experimental setup and implementation details in Sections 5.4.1 and 5.4.2. Section 5.4.3 evaluates alternative obstacle representations while Sections 5.4.4 and 5.4.5 compare our approach to the state of the art in challenging environments.

5.4.1 Environments

We evaluate our method in a number of different environments, namely, 2D and 3D environments used in [Ichter, 2018; Qureshi, 2019], our own 3D environments and

an environment based on a classic motion planning problem [Latombe,] where a S-shape with 6 DoF should go through a thin slot. We consider rigid body robots which are either a 2D/3D sphere with 2/3 DoF or a S-shape body with 6 DoF. We use distinct workspaces for training and evaluation so that policies are evaluated on workspaces unseen during training. We evaluate the success rate of a policy over 400 rollouts. At the end of a rollout, the environment is reset: a new random workspace is sampled along with a start and goal configuration. A rollout is considered successful if it reaches a configuration near the goal defined by an epsilon neighborhood before the maximum number of steps is reached. We describe details for each of our environments below.

2D-Narrow [Ichter, 2018]: we generate 2D environments from [Ichter, 2018] using publicly available code [Ichter, 2020]. The environment contains a sphere robot navigating in 2D workspaces composed of 3 randomly generated narrow gaps as shown in Fig.5.2 and random start and goal configurations. We set the maximum number of policy steps to 50.
3D-Qureshi [Qureshi, 2019]: 3D workspaces with a sphere robot from [Qureshi, 2019] contain axis-aligned boxes with fixed center and varying sizes. We used open-source code from [Qureshi, 2020] to generate the workspaces.
3D-Boxes: our environment composed of 3 to 10 static boxes generated with random sizes, positions and orientations as illustrated in Fig.5.4(a). The maximum number of steps is set to 80.
3D-Synthethic: a variant of 3D-Boxes composed of unseen synthetic obstacles such as capsules, cylinders and spheres instead of boxes as illustrated in Fig.5.4(b).
3D-YCB: a variant of 3D-Boxes composed of real objects from the YCB dataset [Çalli, 2015] recorded with a RGB-D camera, see Fig.5.4(c).
3D-Dynamic: a variant of 3D-Boxes with dynamic obstacles moving in real time. For each box two placements are sampled uniformly, and the current box placement is interpolated between the two.
Slot: S-shaped rigid 6 DoF robot that should pass through a thin slit of varying width 2-8 times wider than the smallest robot link. This is a classic problem in motion planning [Latombe,] illustrated in Fig.5.4(d).

5.4.2 Implementation details

Below we describe implementation details for our own and other methods used for comparison. To train policies with reinforcement learning, we use Soft Actor Critic (SAC) [Haarnoja, 2018b] with automatic entropy tuning, a learning rate of 3.10^{-4}

with Adam optimizer [Kingma, 2015b], a batch size of 256 and a replay buffer of 10^6 steps combined with Hindsight Experience Replay (HER) [Andrychowicz, 2017b] with 80% of the trajectories goal relabeled, as in the original papers. We train a policy on 2.10^6 environment steps before reporting results. We use the open-source implementation of [Pong, 2020]. The policy π and the Q-function are implemented as separate networks and trained jointly with alternate optimization following the actor-critic scheme. We use a PointNet based policy with 3 hidden layers of size 256 for the point encoder u and the global feature network w respectively as shown in Fig.5.1c. The Q-function has a similar structure and the action is concatenated to each point of the point cloud.

For the comparison of representations in Table 5.1 we use a CNN policy with 64×64 image inputs composed of 3 convolutional layers with 32 filters of size 3×3 followed by a max-pooling operator and 3 hidden layers of size 256. The current configuration and goal are concatenated after max-pooling. For the Q-function, the action is also concatenated after max-pooling. The MLP policy processing point clouds is composed of 3 hidden layers of size 256 and the goal is concatenated to the list of points. For the Q-function, the action is also concatenated to the input.

To train policies with imitation learning, we use a learning rate of 10^{-3} with Adam optimizer [Kingma, 2015b] and a batch size of 256. For Bi-RRT, once a solution is found, we shorten its length by randomly sampling two points along the solution, if the shortcut made out of these two points is collision free we modify the solution to include it. The maximal size of an edge in RRT corresponds to the maximal size of a policy step for RL or BC, in this way computing collision checking on an edge has the same cost for both.

For [Ichter, 2018], we used the code provided in [Ichter, 2020] along with the dataset to train the conditional variational auto-encoder. Once trained we combined the learned sampling with Bi-RRT to report the results. For [Qureshi, 2019], we adjust the implementation provided by [Qureshi, 2020] to our environments. We followed the training procedure described in [Qureshi, 2019]. For Qureshi neural replanning (NR), we run neural planning for 50 steps then use neural replanning recursively for 10 iterations each comprising 50 steps maximum. For Qureshi hybrid replanning (HR), if neural replanning fails to find a solution after 10 iterations, Bi-RRT is used to find a path between states where there is still a collision. For [Jurgenson, 2019], we adapted the open-source implementation [Jurgenson, 2020] to run on our environment. For training, we use the same parameters as [Jurgenson, 2019], image based workspace representation and reward definition. We follow their DDPG-MP

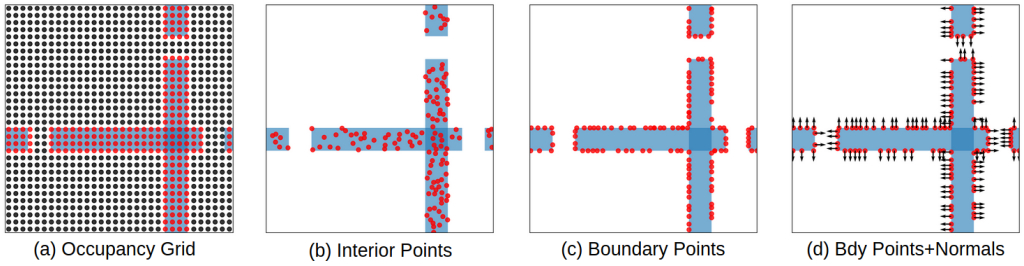


Figure 5.2: Illustration of different obstacles representations for 2D-Narrow. See Table 5.1 for results.

method. The training dataset consists of 10^4 workspaces. For pre-training, we use 400 random steps per workspace and for training, we use Bi-RRT generated 10 expert trajectories per workspace. Finally, the rigid bodies and obstacles are modeled using Pinocchio [Carpentier, 2015–2019; Carpentier, 2019] and collision checks are computed by FCL [Pan, 2012].

5.4.3 Comparison of obstacle representations and learning approaches

In this section we compare different policies on the 2D-Narrow environment [Ichter, 2018]. Policies are trained with Behavioral Cloning (BC) or Reinforcement Learning (RL) using different obstacles representations presented in Fig.5.2. The occupancy grid is a 64×64 image (Fig. 5.2a), the point clouds are composed of 128 points either sampled on the interior of obstacles (Fig. 5.2b) or at their boundary (Fig. 5.2c, 5.2d). 128 points is a good trade-off between speed and accuracy as adding more points did not improve our results. In Table 5.1 we report the success rate of RL policies after 1000 epochs which corresponds to 10^6 interactions with the environment, a stage at which the policies performance have plateaued. For BC, we collect a dataset of solutions using Bi-RRT, containing 10^6 steps in total and train for 200 epochs. This allows for a fair comparison of BC and RL trained on the same dataset size. A simple baseline connecting the start and the goal by a straight line has success rate of 45.5%.

Table 5.1 shows that the choice of obstacles representation greatly impacts the policy success rate both for BC and RL. Using a representation based on an occupancy grid encoded with a CNN yields poor performance for both BC and RL. Similarly, if interior points are encoded with a MLP the performance is low for both BC and RL. In both case results with RL are below BC. For BC, the training set is composed of expert demonstrations which is fixed for every representation. In contrast, as the RL

	Behavioral Cloning	Reinforcement Learning
64×64 Image - CNN	64.5%	44.5%
Interior points - MLP	42.0%	29.0%
Interior points - PointNet	76.0%	83.0%
Boundary points - PointNet	85.0%	93.8%
Boundary points and normals - PointNet	86.5%	**99.5%**

Table 5.1: Comparison of different obstacles representations and policies training methods on the 2D-Narrow environment.

training set is generated by the learning policy, the obstacles representation impacts the quality of the training set. Encoding interior points with PointNet results in a significant gain, +34% for BC and +54% for RL. Using PointNet in combination with boundary points and normals increases the performance by +10% for BC and +16% for RL. We can also observe that normals improve the performance over boundary points alone. Furthermore, with a more stable obstacle encoding, RL outperforms BC by a margin, +13% in the case of boundary points and normals.

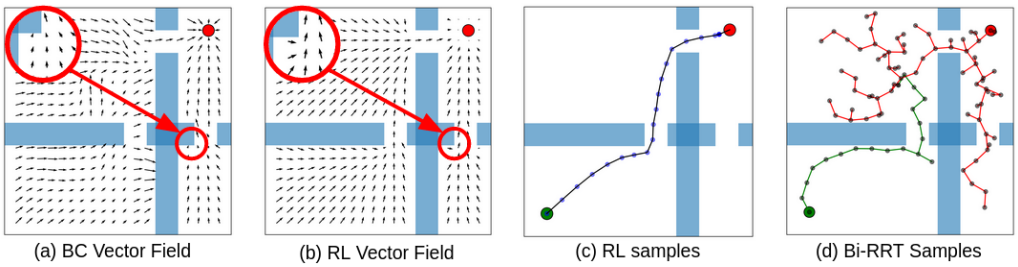


Figure 5.3: (a) Vector field of a policy trained with behavioral cloning. (b) Vector field of a policy trained with reinforcement learning. (c) RL path samples. (d) RRT samples generated to find a path.

The difference between policies trained with BC or RL is illustrated in Fig.5.3(a,b). For each policy, a vector field has been generated by using a grid covering the environment and computing the policy output at each point given a fixed goal plotted in red. We observe that policies trained with BC can fail close to edges of obstacles. This is a typical problem of imitation learning [Ross, 2011] which is limited to perfect, expert trajectories in the training set and does not observe failure cases. In contract, RL explores the environment during training and generates actions pointing away from obstacles as it has been trained explicitly to avoid collisions.

We also evaluate alternative obstacle representations on a collision classification task in the 2D-Narrow environment (see Fig.5.2). Given a configuration and an action, the goal is to predict if the robot configuration after executing the action

Method	Success Rate	Nodes	Path Length
Bi-RRT [Jr, 2000]	100%	358	0.65
Ichter $\lambda = 0.5$ [Ichter, 2018]	100%	232	1.30
Ichter $\lambda = 0.9$ [Ichter, 2018]	100%	207	1.22
Qureshi (NR) [Qureshi, 2019]	68.0%	102	0.52
Qureshi (HR) [Qureshi, 2019]	95.0%	950	0.69
Jurgenson [Jurgenson, 2019]	47.0%	12	0.56
Ours	99.5%	10	0.63

Table 5.2: Comparison to the state of the art on the 2D-Narrow environment.

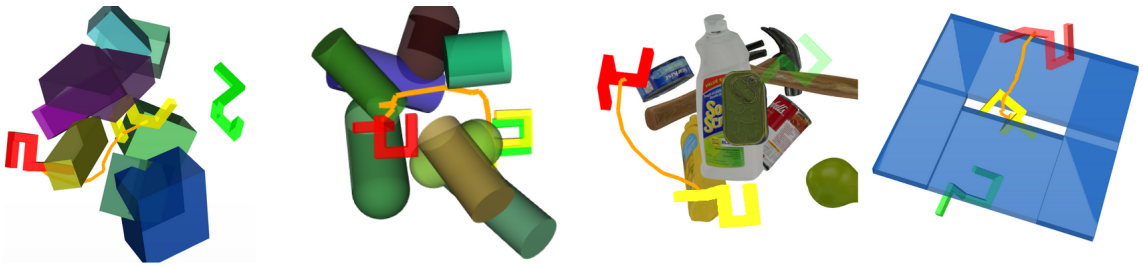
is colliding with any obstacle or not. We compare classifiers trained on top of (a) 64x64 images encoded with CNNs, (b) interior points, (c) boundary points and (d) boundary points with normals, where (b), (c) and (d) are encoded with PointNet. The resulting accuracies for collision prediction are: (a) 80%, (b) 94%, (c) 97% and (d) 98.5%. Consistently with results in Table 5.1, we observe that (i) PointNet representations outperform occupancy grid with CNN and (ii) boundary points with normals encoding give best results. Further analysis of failure cases has shown inaccuracy of CNN collision predictions at locations near the boundaries of obstacles and in regions with narrow passages.

We compare our approach to state-of-the-art neural motion planning approaches [Ichter, 2018; Jurgenson, 2019; Qureshi, 2019] and Bi-RRT [Jr, 2000] in Table 5.2. We compare in terms of success rate, number of configurations (nodes) explored before finding a successful path and in terms of length of the found solution. We chose to compare the number of configurations explored to find a solution because connecting two configurations requires to perform collision checking which represents 95% of time spent by motion planning algorithms [Ratliff, 2009]. Bi-RRT achieves a success rate of 100% as a solution is found if the algorithm is run long enough but it requires 35 times more nodes than RL to find a solution and shorten the path. [Ichter, 2018], which is also based on RRT, requires 20 times more nodes than RL to find a solution and yields longer solutions overall. The neural replanning (NR) and hybrid replanning (HR) approaches of [Qureshi, 2019] allow to find short paths at the price of extensive use of collision checking, which is limiting in scenarios with time constraints. [Jurgenson, 2019] uses an image representation of obstacles, yielding a low success rate which correlates with the results of Table 5.1, it mostly solves problems with straight solutions which explains the short path length.

In Fig.5.3(c,d) we illustrate the number of nodes required to find a path. While RL outputs short paths leading directly to the goal (Fig.5.3c), Bi-RRT explores the

space in several directions before finding a suboptimal path which then needs to be shortened (Fig.5.3d).

5.4.4 Towards a realistic setup: 3D environments with local observability



(a) 3D-Boxes S Shape (b) 3D-Synthetic S Shape (c) 3D-YCB S Shape (d) Slot

Figure 5.4: 3D environments with S-shaped robot composed of: (a) boxes obstacles, (b) sphere, cylinder and cone obstacles, (c) YCB dataset [Çalli, 2015] obstacles, (d) a thin slot. We plot the start configuration in red, the current configuration in yellow and the goal configuration in green.

We compare agents trained with either global observations, where points are sampled for all obstacles surface as in Section 5.4.3, or local observations where obstacles are only observed locally, through a camera which is closer to a realistic setup. For the local observation, we consider agents equipped with a panoramic camera observing the local geometry (Fig.5.1b). To model such agents, we use ray tracing and cast rays from the center of the robot in every direction by uniformly sampling points on the sphere. Each ray hitting an obstacle provides a point with its normal and the agent observation consists of the point cloud formed by the union of these points. For local observations (`Local`), we use a point cloud of 64 points, corresponding to a density of 60 points per meter squared. For global observations (`Global`), we use 256 points sampled uniformly on the obstacles surface which corresponds to the same point density as `Local`. The two observations thus have the same geometry resolution.

We consider 3D environments and compare agents controlling either a sphere robot with 3DoF or a S-shaped robot with 6DoF as shown in Fig.5.4 and report results in Table 5.3. All the policies are trained on the 3D-Boxes environment (Fig.5.4a) exclusively. The S-shape problem is harder to solve than the sphere one but our approach still yields good results with a performance of 97% on 3D-Boxes for the local observation. Overall `Local` and `Global` policies yield similar performances which shows that our approach still works on harder problems where only local knowledge of obstacles is available. When tested on 3D-Boxes, `Local` yields better

results than `Global` while the generalization performance are better for `Global` when tested on unseen environments (Fig.5.4b,c). While solely trained on problems with static obstacles from 3D Boxes (Fig.5.4a), the policies generalize to unseen scenes containing new set of synthetic obstacles (3D-Synthetic) and real obstacles recorded with depth sensor from the YCB dataset (3D-YCB). The policies trained with our approach also solve challenging scenarios with dynamic obstacles moving in real time (3D-Dynamic). This highlights the advantage of using a policy which directly computes the next action instead of RRT-based approaches relying on offline path planning [Jr, 2000; Qureshi, 2019; Ichter, 2019].

	3D-Boxes	3D-Synthetic	3D-YCB	3D-Dynamic
Sphere - `Global`	95.8%	91.3%	91.2%	94.7%
S Shape - `Global`	89.0%	85.3%	88.7%	86.0%
Sphere - `Local`	99.3%	87.0%	87.7%	95.8%
S Shape - `Local`	97.0%	87.3%	96.0%	79.7%

Table 5.3: Comparison of sphere and S-shaped agents trained with global or local obstacle observation.

	3D-Qureshi [Qureshi, 2019]	3D-Boxes	3D-Synthetic
Qureshi (NR) [Qureshi, 2019] - MLP	96.0%	84.0%	81.0%
Qureshi (HR) [Qureshi, 2019] - MLP	99.5%	88.0%	84.5%
Sphere - `Global`	**100%**	**95.8%**	**91.3%**

Table 5.4: Comparison of our approach to Qureshi [Qureshi, 2019] on Qureshi's environment and our environment.

We also compare our approach to [Qureshi, 2019]. For a fair comparison we use global obstacles representation for our approach as [Qureshi, 2019] uses full obstacles knowledge and report results in Table 5.4. On 3D-Qureshi, we show that our approach successfully solves the proposed problems. On 3D-Boxes and 3D-Synthetic we show that our approach has better generalization abilities while only needing 20 nodes on average to solve problems where [Qureshi, 2019] requires more than 400 nodes.

5.4.5 S-shape motion planning

We consider a challenging problem in motion planning composed of a S-shaped robot and a thin gate it has to go through, introduced by [Latombe,] and shown

in Fig.5.4(d). The width of the gate determines the difficulty of the problem. We consider problems with a gate of random width, sampled to be 2 times to 8 times wider than the smallest dimension of each robot link, as a comparison, the gate of [Latombe,] is 2.5 times wider. We compare the performance of an agent trained with `Local` observations to Bi-RRT with an allocated budget of 50000 nodes for each problem. The learned policy has a success rate of 97.7% whereas Bi-RRT has a success rate of 62.5% on average when tested on 400 planning problems. In contrast with experiments of Section 5.4.3, Bi-RRT does not succeed to solve every problem with the allocated nodes. While our trained policy provides solutions in real-time composed of 40 nodes on average, the computational burden of RRT is increasing significantly as the number of explored configurations increases which is typically the case for this environment where many nodes need to be expanded to find a solution. We have also noted that our policy adapts its behavior to minimize the path length according to the slot size. Indeed, when the slot is thin, e.g. 2 times wider than the smallest robot link, the motions are quite constrained, the policy inserts a link by translating the S-shape, rotates by 90 degrees and translates again which was also the only solution found when using RRT. When the slot is wider we observe that the policy uses the wider space provided by the diagonal of the slot to reduce overall motion.

5.5 Conclusion

This chapter introduces a new framework for neural motion planning. Obstacles are represented by point clouds and encoded by a PointNet architecture. PointNet encoding and motion policy are trained jointly with either behavioral cloning or reinforcement learning. We show that PointNet encoding outperforms state-of-the-art representations based on image-based CNNs and latent representations.

Chapter 6

Segmenter: transformer for semantic segmentation

Image segmentation is often ambiguous at the level of individual image patches and requires contextual information to reach label consensus. In this chapter we introduce Segmenter, a transformer model for semantic segmentation. In contrast to convolution-based methods, our approach allows modeling global context already at the first layer and throughout the network. We build on the recent Vision Transformer (ViT) and extend it to semantic segmentation. To do so, we rely on the output embeddings corresponding to image patches and obtain class labels from these embeddings with a point-wise linear decoder or a mask transformer decoder. We leverage models pre-trained for image classification and show that we can fine-tune them on moderate sized datasets available for semantic segmentation. The linear decoder allows obtaining excellent results already, but the performance can be further improved by a mask transformer generating class masks. We conduct an extensive ablation study to show the impact of the different parameters, in particular the performance is better for large models and small patch sizes. Segmenter attains excellent results for semantic segmentation. It outperforms the state of the art on both ADE20K and Pascal Context datasets and is competitive on Cityscapes.

6.1 Introduction

Semantic segmentation is a challenging computer vision problem with a wide range of applications including autonomous driving, robotics, augmented reality, image editing, medical imaging and many others [Hesamian, 2019; Hong, 2018; Siam, 2017].

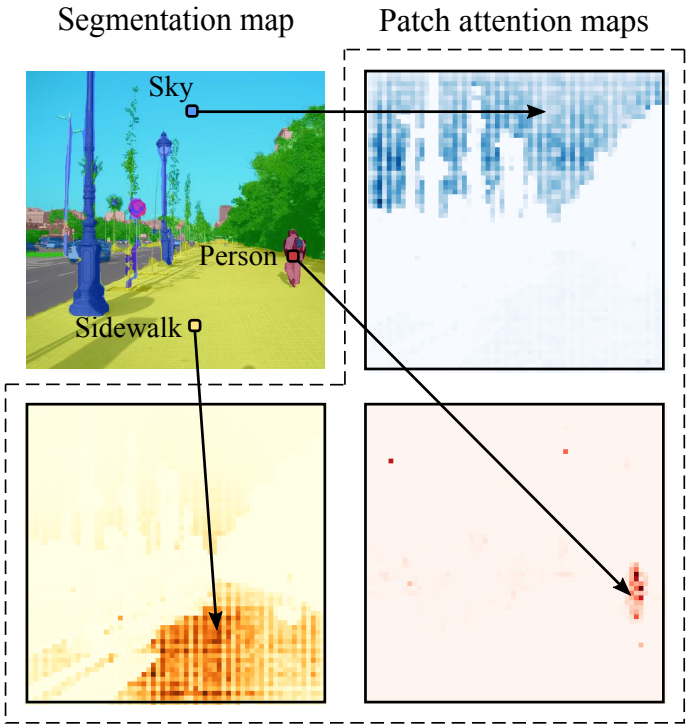


Figure 6.1: Our approach for semantic segmentation is purely transformer based. It leverages the global image context at every layer of the model. Attention maps from the first Segmenter layer are displayed for three 8×8 patches and highlight the early grouping of patches into semantically meaningful categories. The original image (top-left) is overlayed with segmentation masks produced by our method.

The goal of semantic segmentation is to assign each image pixel to a category label corresponding to the underlying object and to provide high-level image representations for target tasks, e.g. detecting the boundaries of people and their clothes for virtual try-on applications [Hsieh, 2019]. Despite much effort and large progress over recent years [Chen, 2018d; Fu, 2020; Huang, 2019; Minaee, 2021; Sultana, 2020; Zhao, 2017; Zhao, 2018], image segmentation remains a challenging problem due to rich intra-class variation, context variation and ambiguities originating from occlusions and low image resolution.

Recent approaches to semantic segmentation typically rely on convolutional encoder-decoder architectures where the encoder generates low-resolution image features and the decoder upsamples features to segmentation maps with per-pixel class scores. State-of-the-art methods deploy Fully Convolutional Networks (FCN) [Shelhamer, 2017] and achieve impressive results on challenging segmentation benchmarks [Chen, 2018d; Fu, 2019b; Yin, 2020; Yu, 2020; Yuan, 2020; Zhang, 2019; Zhao, 2018]. These methods rely on learnable stacked convolutions that can capture semantically rich information and have been highly successful in computer vision. The local nature

of convolutional filters, however, limits the access to the global information in the image. Meanwhile, such information is particularly important for segmentation where the labeling of local patches often depends on the global image context. To circumvent this issue, DeepLab methods [Chen, 2018b; Chen, 2017; Chen, 2018d] introduce feature aggregation with dilated convolutions and spatial pyramid pooling. This allows to enlarge the receptive fields of convolutional networks and to obtain multi-scale features. Following recent progresses in NLP [Vaswani, 2017], several segmentation methods explore alternative aggregation schemes based on channel or spatial [Fu, 2020; Fu, 2019b; Yuan, 2018] attention and point-wise [Zhao, 2018] attention to better capture contextual information. Such methods, however, still rely on convolutional backbones and are, hence, biased towards local interactions. An extensive use of specialised layers to remedy this bias [Chen, 2018b; Chen, 2018d; Fu, 2020; Yu, 2020] suggests limitations of convolutional architectures for segmentation.

To overcome these limitations, we formulate the problem of semantic segmentation as a sequence-to-sequence problem and use a transformer architecture [Vaswani, 2017] to leverage contextual information at every stage of the model. By design, transformers can capture global interactions between elements of a scene and have no built-in inductive prior, see Figure 6.1. However, the modeling of global interactions comes at a quadratic cost which makes such methods prohibitively expensive when applied to raw image pixels [Chen, 2020a]. Following the recent work on Vision Transformers (ViT) [Dosovitskiy, 2021a; Touvron, 2020], we split the image into patches and treat linear patch embeddings as input tokens for the transformer encoder. The contextualized sequence of tokens produced by the encoder is then upsampled by a transformer decoder to per-pixel class scores. For decoding, we consider either a simple point-wise linear mapping of patch embeddings to class scores or a transformer-based decoding scheme where learnable class embeddings are processed jointly with patch tokens to generate class masks. We conduct an extensive study of transformers for segmentation by ablating model regularization, model size, input patch size and its trade-off between accuracy and performance. Our Segmenter approach attains excellent results while remaining simple, flexible and fast. In particular, when using large models with small input patch size the best model reaches a mean IoU of 53.63% on the challenging ADE20K [Zhou, 2019] dataset, surpassing all previous state-of-the-art convolutional approaches by a large margin of 5.3%. Such improvement partly stems from the global context captured by our method at every stage of the model as highlighted in Figure 6.1.

In summary, our work provides the following four contributions: (i) We propose a

novel approach to semantic segmentation based on the Vision Transformer (ViT) [Dosovitskiy, 2021a] that does not use convolutions, captures contextual information by design and outperforms FCN based approaches. (ii) We present a family of models with varying levels of resolution which allows trading off between precision and run-time, ranging from *state-of-the-art* performance to models with fast inference and good performances. (iii) We propose a transformer-based decoder generating class masks which outperforms our linear baseline and can be extended to perform more general image segmentation tasks. (iv) We demonstrate that our approach yields *state-of-the-art* results on both ADE20K [Zhou, 2019] and Pascal Context [Mottaghi, 2014] datasets and is competitive on Cityscapes [Cordts, 2016].

6.2 Related work

Semantic segmentation. Methods based on Fully Convolutional Networks (FCN) combined with encoder-decoder architectures have become the dominant approach to semantic segmentation. Initial approaches [Farabet, 2013; Long, 2015; Pinheiro, 2014] rely on a stack of consecutive convolutions followed by spatial pooling to perform dense predictions. Consecutive approaches [Amirul Islam, 2017; Badrinarayanan, 2017; Lin, 2017; Pohlen, 2017; Ronneberger, 2015] upsample high-level feature maps and combine them with low-level feature maps during decoding to both capture global information and recover sharp object boundaries. To enlarge the receptive field of convolutions in the first layers, several approaches [Chen, 2018b; Liang-Chieh, 2015; Yu, 2016a] have proposed dilated or atrous convolutions. To capture global information in higher layers, recent work [Chen, 2017; Chen, 2018d; Zhao, 2017] employs spatial pyramid pooling to capture multi-scale contextual information. Combining these enhancements along with atrous spatial pyramid pooling, Deeplabv3+ [Chen, 2018d] proposes a simple and effective encoder-decoder FCN architecture. Recent work [Fu, 2020; Fu, 2019b; Yin, 2020; Yu, 2020; Yuan, 2018; Zhao, 2018] replace coarse pooling by attention mechanisms on top of the encoder feature maps to better capture long-range dependencies.

While recent segmentation methods are mostly focused on improving FCN, the restriction to local operations imposed by convolutions may imply inefficient processing of global image context and suboptimal segmentation results. Hence, we propose a pure transformer architecture that captures global context at every layer of the model during the encoding and decoding stages.

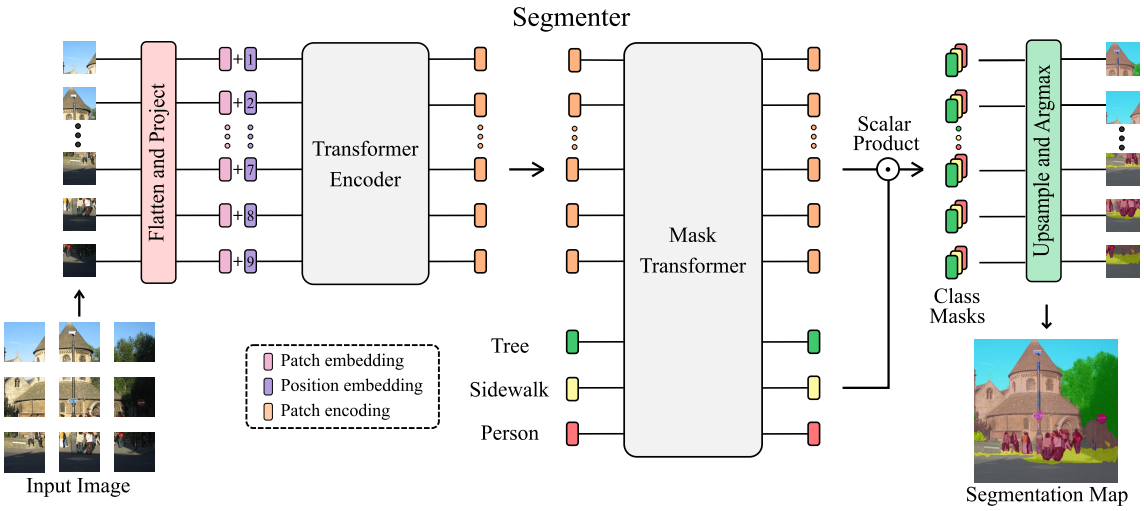


Figure 6.2: Overview of our approach Segmenter. (Left) Encoder: The image patches are projected to a sequence of embeddings and then encoded with a transformer. (Right) Decoder: A mask transformer takes as input the output of the encoder and class embeddings to predict segmentation masks. See text for details.

Transformers for vision. Transformers [Vaswani, 2017] are now state of the art in many Natural Language Processing (NLP) tasks. Such models rely on self-attention mechanisms and capture long-range dependencies among tokens (words) in a sentence. In addition, transformers are well suited for parallelization, facilitating training on large datasets. The success of transformers in NLP has inspired several methods in computer vision combining CNNs with forms of self-attention to address object detection [Carion, 2020], semantic segmentation [Wang, 2020a], panoptic segmentation [Wang, 2020b], video processing [Wang, 2018] and few-shot classification [Doersch, 2020].

Recently, the Vision Transformer (ViT) [Dosovitskiy, 2021a] introduced a convolution-free transformer architecture for image classification where input images are processed as sequences of patch tokens. While ViT requires training on very large datasets, DeiT [Touvron, 2020] proposes a token-based distillation strategy and obtains a competitive vision transformer trained on the ImageNet-1k [Deng, 2009] dataset using a CNN as a teacher. Concurrent work extends this work to video classification [Arnab, 2021a; Bertasius, 2021] and semantic segmentation [Liu, 2021b; Zheng, 2020]. In more detail, SETR [Zheng, 2020] uses a ViT backbone and a standard CNN decoder. As our approach, concurrent works also use ViT [Dosovitskiy, 2021a] as a backbone for semantic segmentation. Swin Transformer [Liu, 2021b] uses a variant of ViT, composed of local windows, shifted between layers and Upper-Net as a pyramid FCN decoder.

Here, we propose Segmenter, a transformer encoder-decoder architecture for semantic image segmentation. Our approach relies on a ViT backbone and introduces a mask decoder inspired by DETR [Carion, 2020]. Our architecture does not use convolutions, captures global image context by design and results in competitive performance on standard image segmentation benchmarks.

6.3 Our approach: Segmenter

Segmenter is based on a fully transformer-based encoder-decoder architecture mapping a sequence of patch embeddings to pixel-level class annotations. An overview of the model is shown in Figure 6.2. The sequence of patches is encoded by a transformer encoder described in Section 6.3.1 and decoded by either a point-wise linear mapping or a mask transformer described in Section 6.3.2. Our model is trained end-to-end with a per-pixel cross-entropy loss. At inference time, argmax is applied after upsampling to obtain a single class per pixel.

6.3.1 Encoder

An image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is split into a sequence of patches $\mathbf{x} = [x_1, ..., x_N] \in \mathbb{R}^{N \times P^2 \times C}$ where (P, P) is the patch size, $N = HW/P^2$ is the number of patches and C is the number of channels. Each patch is flattened into a 1D vector and then linearly projected to a patch embedding to produce a sequence of patch embeddings $\mathbf{x_0} = [\mathbf{E}x_1, ..., \mathbf{E}x_N] \in \mathbb{R}^{N \times D}$ where $\mathbf{E} \in \mathbb{R}^{D \times (P^2 C)}$. To capture positional information, learnable position embeddings $\mathbf{pos} = [\text{pos}_1, ..., \text{pos}_N] \in \mathbb{R}^{N \times D}$ are added to the sequence of patches to get the resulting input sequence of tokens $\mathbf{z_0} = \mathbf{x_0} + \mathbf{pos}$.

A transformer [Vaswani, 2017] encoder composed of L layers is applied to the sequence of tokens $\mathbf{z}_0$ to generate a sequence of contextualized encodings $\mathbf{z}_L \in \mathbb{R}^{N \times D}$. A transformer layer consists of a multi-headed self-attention (MSA) block followed by a point-wise MLP block of two layers with layer norm (LN) applied before every block and residual connections added after every block:

$$\mathbf{a_{i-1}} = \text{MSA}(\text{LN}(\mathbf{z_{i-1}})) + \mathbf{z_{i-1}}, \tag{6.1}$$

$$\mathbf{z_i} = \text{MLP}(\text{LN}(\mathbf{a_{i-1}})) + \mathbf{a_{i-1}}, \tag{6.2}$$

where $i \in \{1, ..., L\}$. The self-attention mechanism is composed of three point-wise linear layers mapping tokens to intermediate representations, queries $\mathbf{Q} \in \mathbb{R}^{N \times d}$,

keys $\mathbf{K} \in \mathbb{R}^{N \times d}$ and values $\mathbf{V} \in \mathbb{R}^{N \times d}$. Self-attention is then computed as follows

$$\text{MSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}. \tag{6.3}$$

The transformer encoder maps the input sequence $\mathbf{z_0} = [z_{0,1}, ..., z_{0,N}]$ of embedded patches with position encoding to $\mathbf{z_L} = [z_{L,1}, ..., z_{L,N}]$, a contextualized encoding sequence containing rich semantic information used by the decoder. In the following section we introduce the decoder.

6.3.2 Decoder

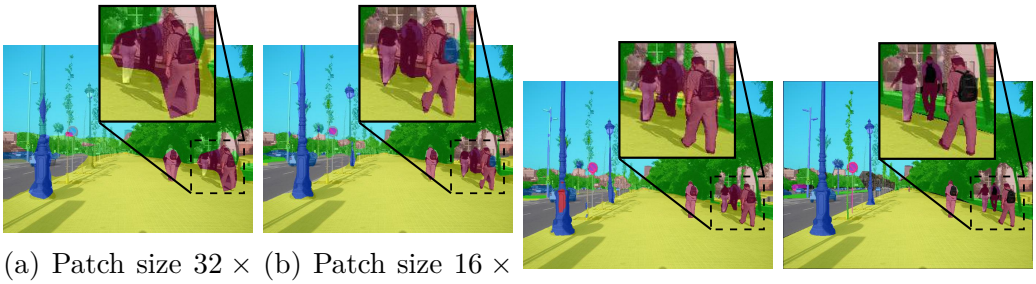
The sequence of patch encodings $\mathbf{z_L} \in \mathbb{R}^{N \times D}$ is decoded to a segmentation map $\mathbf{s} \in \mathbb{R}^{H \times W \times K}$ where K is the number of classes. The decoder learns to map patch-level encodings coming from the encoder to patch-level class scores. Next these patch-level class scores are upsampled by bilinear interpolation to pixel-level scores. We describe in the following a linear decoder, which serves as a baseline, and our approach, a mask transformer, see Figure 6.2.

Linear. A point-wise linear layer is applied to the patch encodings $\mathbf{z_L} \in \mathbb{R}^{N \times D}$ to produce patch-level class logits $\mathbf{z_{lin}} \in \mathbb{R}^{N \times K}$. The sequence is then reshaped into a 2D feature map $\mathbf{s_{lin}} \in \mathbb{R}^{H/P \times W/P \times K}$ and bilinearly upsampled to the original image size $\mathbf{s} \in \mathbb{R}^{H \times W \times K}$. A softmax is then applied on the class dimension to obtain the final segmentation map.

Mask Transformer. For the transformer-based decoder, we introduce a set of K learnable class embeddings $\mathbf{cls} = [\text{cls}_1, ..., \text{cls}_K] \in \mathbb{R}^{K \times D}$ where K is the number of classes. Each class embedding is initialized randomly and assigned to a single semantic class. It will be used to generate the class mask. The class embeddings **cls** are processed jointly with patch encodings $\mathbf{z_L}$ by the decoder as depicted in Figure 6.2. The decoder is a transformer encoder composed of M layers. Our mask transformer generates K masks by computing the scalar product between L2-normalized patch embeddings $\mathbf{z'_M} \in \mathbb{R}^{N \times D}$ and class embeddings $\mathbf{c} \in \mathbb{R}^{K \times D}$ output by the decoder. The set of class masks is computed as follows

$$\text{Masks}(\mathbf{z'_M}, \mathbf{c}) = \mathbf{z'_M}\mathbf{c}^T \tag{6.4}$$

where $\text{Masks}(\mathbf{z'_M}, \mathbf{c}) \in \mathbb{R}^{N \times K}$ is a set of patch sequence. Each mask sequence is then reshaped into a 2D mask to form $\mathbf{s_{mask}} \in \mathbb{R}^{H/P \times W/P \times K}$ and bilinearly upsampled to the original image size to obtain a feature map $\mathbf{s} \in \mathbb{R}^{H \times W \times K}$. A softmax is then



(a) Patch size 32 × 32 (b) Patch size 16 × 16 (c) Patch size 8 × 8 (d) Ground Truth

Figure 6.3: Impact of the model patch size on the segmentation maps.

applied on the class dimension followed by a layer norm to obtain pixel-wise class score forming the final segmentation map. The masks sequences are softly exclusive to each other i.e. $\sum_{k=1}^{K} s_{i,j,k} = 1$ for all $(i, j) \in H \times W$.

Our mask transformer is inspired by DETR [Carion, 2020], Max-DeepLab [Wang, 2020c] and SOLO-v2 [Wang, 2020d] which introduce object embeddings [Carion, 2020] to produce instance masks [Wang, 2020c; Wang, 2020d]. However, unlike our method, MaxDeep-Lab uses a hybrid approach based on CNNs and transformers and splits the pixel and class embeddings into two streams because of computational constraints. Using a pure transformer architecture and leveraging patch level encodings, we propose a simple approach that processes the patch and class embeddings jointly during the decoding phase. Such approach allows producing dynamical filters, changing with the input. While we address semantic segmentation in this work, our mask transformer can also be directly adapted to perform panoptic segmentation by replacing the class embeddings by object embeddings.

6.3.3 Datasets and metrics

ADE20K [Zhou, 2019]. This dataset contains challenging scenes with fine-grained labels and is one of the most challenging semantic segmentation datasets. The training set contains 20,210 images with 150 semantic classes. The validation and test set contain 2,000 and 3,352 images respectively.

Pascal Context [Mottaghi, 2014]. The training set contains 4,996 images with 59 semantic classes plus a background class. The validation set contains 5,104 images.

Cityscapes [Cordts, 2016]. The dataset contains 5,000 images from 50 different cities with 19 semantic classes. There are 2,975 images in the training set, 500 images in the validation set and 1,525 images in the test set.

Metrics. We report Intersection over Union (mIoU) averaged over all classes.

Model	Backbone	Layers	Token size	Heads	Params
Seg-Ti	ViT-Ti	12	192	3	6M
Seg-S	ViT-S	12	384	6	22M
Seg-B	ViT-B	12	768	12	86M
Seg-B†	DeiT-B	12	768	12	86M
Seg-L	ViT-L	24	1024	16	307M

Table 6.1: Details of Transformer variants.

6.3.4 Implementation details

Transformer models. For the encoder, we build upon the vision transformer ViT [Dosovitskiy, 2021a] and consider "Tiny", "Small", "Base" and "Large" models described in Table 6.1. The parameters varying in the transformer encoder are the number of layers and the token size. The head size of a multi-headed self-attention (MSA) block is fixed to 64, the number of heads is the token size divided by the head size and the hidden size of the MLP following MSA is four times the token size. We also use DeiT [Touvron, 2020], a variant of the vision transformer. We consider models representing the image at different resolutions and use input patch sizes 8×8, 16×16 and 32×32. In the following, we use an abbreviation to describe the model variant and patch size, for instance Seg-B/16 denotes the "Base" variant with 16×16 input patch size. Models based on DeiT are denoted with a †, for instance Seg-B†/16.

ImageNet pre-training. Our Segmenter models are pre-trained on ImageNet, ViT is pre-trained on ImageNet-21k with strong data augmentation and regularization [Steiner, 2021] and its variant DeiT is pre-trained on ImageNet-1k. The original ViT models [Dosovitskiy, 2021a] have been trained with random cropping only, whereas the training procedure proposed by [Steiner, 2021] uses a combination of dropout [Srivastava, 2014] and stochastic depth [Huang, 2016] as regularization and Mixup [Zhang, 2018b] and RandAugment [Cubuk, 2020] as data augmentations. This significantly improves the ImageNet top-1 accuracy, i.e., it obtains a gain of +2% on ViT-B/16. We fine-tuned ViT-B/16 on ADE20K with models from [Dosovitskiy, 2021a] and [Steiner, 2021] and observe a significant difference, namely a mIoU of 45.69% and 48.06% respectively. In the following, all the Segmenter models will be initialized with the improved ViT models from [Steiner, 2021]. We use publicly available models provided by the image classification library timm [Wightman, 2020] and Google research [Dosovitskiy, 2021b]. Both models are pre-trained at an image resolution of 224 and fine-tuned on ImageNet-1k at a resolution of 384, except for ViT-B/8 which has been fine-tuned at a resolution of 224. We keep the patch size

fixed and fine-tune the models for the semantic segmentation task at higher resolution depending on the dataset. As the patch size is fixed, increasing resolution results in longer token sequences. Following [Dosovitskiy, 2021a], we bilinearly interpolate the pre-trained position embeddings according to their original position in the image to match the fine-tuning sequence length. The decoders, described in Section 6.3.2 are initialized with random weights from a truncated normal distribution [Hanin, 2018].

Data augmentation. During training, we follow the standard pipeline from the semantic segmentation library MMSegmentation [Contributors, 2020], which does mean subtraction, random resizing of the image to a ratio between 0.5 and 2.0 and random left-right flipping. We randomly crop large images and pad small images to a fixed size of 512×512 for ADE20K, 480×480 for Pascal-Context and 768×768 for Cityscapes. On ADE20K, we train our largest model Seg-L-Mask/16 with a resolution of 640×640, matching the resolution used by the Swin Transformer [Liu, 2021b].

Optimization. To fine-tune the pre-trained models for the semantic segmentation task, we use the standard pixel-wise cross-entropy loss without weight rebalancing. We use stochastic gradient descent (SGD) [Robbins, 1951] as the optimizer with a base learning rate γ_0 and set weight decay to 0. Following the seminal work of DeepLab [Liang-Chieh, 2015] we adopt the "poly" learning rate decay $\gamma = \gamma_0 \, (1 - \frac{N_{iter}}{N_{total}})^{0.9}$ where N_{iter} and N_{total} represent the current iteration number and the total iteration number. For ADE20K, we set the base learning rate γ_0 to 10^{-3} and train for 160K iterations with a batch size of 8. For Pascal Context, we set γ_0 to 10^{-3} and train for 80K iterations with a batch size of 16. For Cityscapes, we set γ_0 to 10^{-2} and train for 80K iterations with a batch size of 8. The schedule is similar to DeepLabv3+ [Chen, 2018d] with learning rates divided by a factor 10 except for Cityscapes where we use a factor of 1.

Inference. To handle varying image sizes during inference, we use a sliding-window with a resolution matching the training size. For multi-scale inference, following standard practice [Chen, 2018d] we use rescaled versions of the image with scaling factors of (0.5, 0.75, 1.0, 1.25, 1.5, 1.75) and left-right flipping and average the results.

6.3.5 Ablation study

In this section, we ablate different variants of our approach on the ADE20K validation set. We investigate model regularization, model size, patch size, model

		Stochastic Depth		
		0.0	0.1	0.2
Dropout	0.0	45.01	**45.37**	45.10
	0.1	42.02	42.30	41.14
	0.2	36.49	36.63	35.67

Table 6.2: Mean IoU comparison of different regularization schemes using Seg-S/16 on ADE20K validation set.

Method	Backbone	Patch size	Im/sec	ImNet acc.	mIoU (SS)
Seg-Ti/16	ViT-Ti	16	396	78.6	39.03
Seg-S/32	ViT-S	32	1032	80.5	40.64
Seg-S/16	ViT-S	16	196	83.7	45.37
Seg-B†/16	DeiT-B	16	92	85.2	47.08
Seg-B/32	ViT-B	32	516	83.3	43.07
Seg-B/16	ViT-B	16	92	86.0	48.06
Seg-B/8	ViT-B	8	7	85.7	49.54
Seg-L/16	ViT-L	16	33	87.1	50.71

Table 6.3: Performance comparison of different Segmenter models with varying backbones and input patch sizes on ADE20K validation set.

performance, training dataset size, compare Segmenter to convolutional approaches and evaluate different decoders. Unless stated otherwise, we use the baseline linear decoder and report results using single-scale inference.

Regularization. We first compare two forms of regularization, dropout [Srivastava, 2014] and stochastic depth [Huang, 2016], and show that stochastic depth consistently improves transformer training for segmentation. CNN models rely on batch normalization [Ioffe, 2015b] which also acts as a regularizer. In contrast, transformers are usually composed of layer normalization [Ba, 2016] combined with dropout as a regularizer during training [Devlin, 2019; Dosovitskiy, 2021a]. Dropout randomly ignores tokens given as input of a block and stochastic depth randomly skips a learnable block of the model during the forward pass. We compare regularizations on Seg-S/16 based on ViT-S/16 backbone. Table 6.2 shows that stochastic depth set to 0.1, dropping 10% of the layers randomly, consistently improves the performance, with 0.36% when the dropout is set to 0 compared to the baseline without regularization. Dropout consistently hurts performances, either alone or when combined with stochastic depth. This is consistent with [Touvron, 2020] which observed the negative impact of dropout for image classification. From now on, all the models

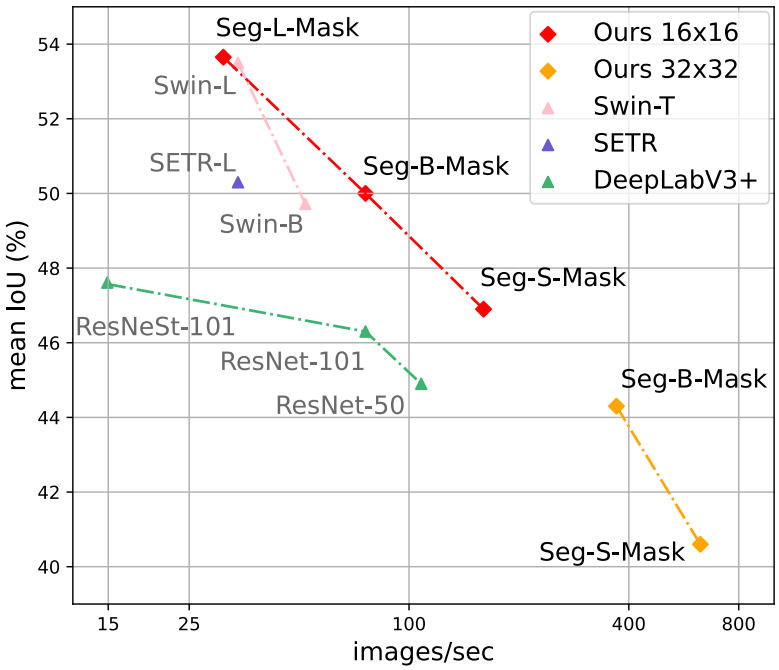


Figure 6.4: Images per second and mean IoU for our approach compared to other methods on ADE20K validation set. Segmenter models offer a competitive trade-off in terms of performance and precision.

will be trained with stochastic depth set to 0.1 and without dropout.

Transformer size. We now study the impact of transformers size on performance by varying the number of layers and the tokens size for a fixed patch size of 16. Table 6.3 shows that performance scales nicely with the backbone capacity. When doubling the token dimension, from Seg-S/16 to Seg-B/16, we get a 2.69% improvement. When doubling the number of layers, from Seg-B/16 to Seg-L/16, we get an improvement of 2.65%. Finally, our largest Segmenter model, Seg-L/16, achieves a strong mIoU of 50.71% with a simple decoding scheme on the ADE20K validation dataset with single scale inference. The absence of tasks-specific layers vastly used in FCN models suggests that transformer based methods provide more expressive models, well suited for semantic segmentation.

Patch size. Representing an image with a patch sequence provides a simple way to trade-off between speed and accuracy by varying the patch size. While increasing the patch size results in a coarser representation of the image, it results in a smaller sequence that is faster to process. The third and backbones and varying patch sizes. We observe that the patch size is a key factor for semantic segmentation performance. It is similarly important to the model size. Indeed, going from a patch size 32 to 16 we observe an improvement of 5% for Seg-B. For Seg-B, we also

Method	Decoder	Small	Medium	Large	mIoU (SS)
DeepLab RNeSt-101	UNet	37.85	50.89	50.67	46.47
Seg-B/32	Linear	31.95	47.82	49.44	43.07
Seg-B-Mask/32	Mask	32.29	49.44	50.82	44.19
Seg-B†/16	Linear	38.31	50.91	52.08	47.10
Seg-B†-Mask/16	Mask	40.49	51.37	54.24	48.70
Seg-B/16	Linear	39.57	51.32	53.28	48.06
Seg-B-Mask/16	Mask	40.16	52.61	52.66	48.48
Seg-B/8	Linear	41.43	54.35	52.85	49.54
Seg-L/16	Linear	42.08	54.67	55.39	50.71
Seg-L-Mask/16	Mask	42.02	54.83	57.06	51.30

Table 6.4: Evaluation with respect to the object size on ADE20k validation set (mean IoU). Comparison of DeepLabv3+ ResNeSt-101 to Segmenter models with a linear or a mask transformer decoder.

Dataset Size	4k	8k	12k	16k	20k
mIoU (SS)	38.31	41.87	43.42	44.61	45.37

Table 6.5: Performance comparison of Seg-S/16 models trained with increasing dataset size and evaluated on ADE20K validation set.

report results for a patch size of 8 and report an mIoU of 49.54%, reducing the gap from ViT-B/8 to ViT-L/16 to 1.17% while requiring substantially fewer parameters. This trend shows that reducing the patch size is a robust source of improvement which does not introduce any parameters but requires to compute attention over longer sequences, increasing the compute time and memory footprint. If it was computationally feasible, ViT-L/8 would probably be the best performing model. Going towards more computation and memory efficient transformers handling larger sequence of smaller patches is a promising direction.

To further study the impact of patch size, we show segmentation maps generated by Segmenter models with decreasing patch size in Figure 6.3. We observe that for a patch size of 32, the model learns a globally meaningful segmentation but produces poor boundaries, for example the two persons on the left are predicted by a single blob. Reducing the patch size leads to considerably sharper boundaries as can be observed when looking at the contours of persons. Hard to segment instances as the thin streetlight pole in the background are only captured at a resolution of 8. In Table 6.4, we report mean IoU with respect to the object size and compare Segmenter to DeepLabv3+ with ResNeSt backbone. To reproduce DeepLabv3+

results, we used models from the MMSegmentation library [Contributors, 2020]. We observe how Seg-B/8 improvement over Seg-B/16 comes mostly from small and medium instances with a gain of 1.27% and 1.74% respectively. Also, we observe that overall the biggest improvement of Segmenter over DeepLab comes from large instances where Seg-L-Mask/16 shows an improvement of 6.39%.

Decoder variants. In this section, we compare different decoder variants. We evaluate the mask transformer introduced in Section 6.3.2 and compare it to the linear baseline. The mask transformer has 2 layers with the same token and hidden size as the encoder. Table 6.4 reports the mean IoU performance. The mask transformer provides consistent improvements over the linear baseline. The most significant gain of 1.6% is obtained for Seg-B†/16, for Seg-B-Mask/32 we obtain a 1.1% improvement and for Seg-L/16 a gain of 0.6%. In Table 6.4 we also examine the gain of different models with respect to the object size. We observe gains both on small and large objects, showing the benefit of using dynamical filters. In most cases the gain is more significant for large objects, i.e., 1.4% for Seg-B/32, 2.1% for Seg-B†/16 and 1.7% for Seg-L/16. The class embeddings learned by the mask transformer are semantically meaningful, i.e., similar classes are nearby, see Figure 6.8 for more details.

Transformer versus FCN. Table 6.4 and Table 6.6 compare our approach to FCN models and DeepLabv3+ [Chen, 2018d] with ResNeSt backbone [Zhang, 2020], one of the best fully-convolutional approaches. Our transformer approach provides a significant improvement over this state-of-the-art convolutional approach, highlighting the ability of transformers to capture global scene understanding. Segmenter consistently outperforms DeepLab on large instances with an improvement of more than 4% for Seg-L/16 and 6% for Seg-L-Mask/16. However, DeepLab performs similarly to Seg-B/16 on small and medium instances while having a similar number of parameters. Seg-B/8 and Seg-L/16 perform best on small and medium instances though at higher computational cost.

Performance. In Figure 6.4, we compare our models to several *state-of-the-art* methods in terms of images per seconds and mIoU and show a clear advantage of Segmenter over FCN based models (green curve). We also show that our approach compares favorably to recent transformer based approach, our largest model Seg-L-Mask/16 is on-par with Swin-L and outperforms SETR-MLA. We observe that Seg/16 models perform best in terms of accuracy versus compute time with Seg-B-Mask/16 offering a good trade-off. Seg-B-Mask/16 outperforms FCN based approaches with similar inference speed, matches SETR-MLA while being twice

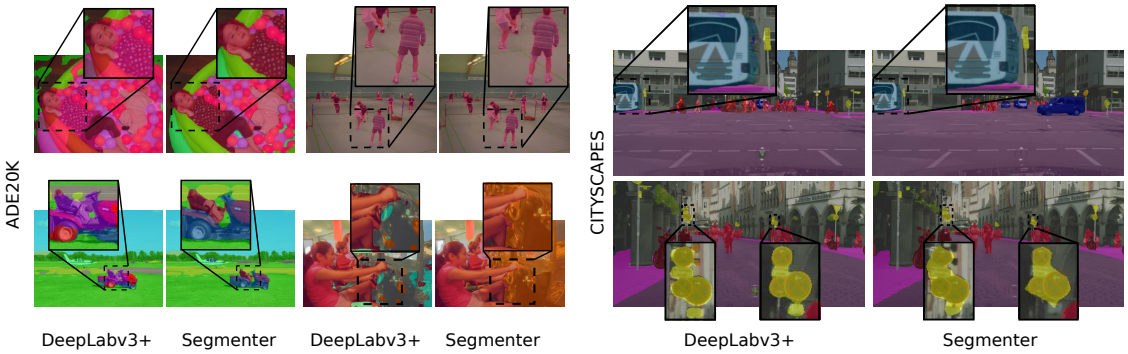


Figure 6.5: Qualitative comparison of Seg-L-Mask/16 performance with DeepLabV3+ ResNeSt-101. See Section 6.4.3 for additional qualitative results.

faster and requiring less parameters and outperforms Swin-B both in terms of inference speed and performance. Seg/32 models learn coarser segmentation maps as discussed in the previous section and enable fast inference with 400 images per second for Seg-B-Mask/32, four times faster than ResNet-50 while providing similar performances. To compute the images per second, we use a V100 GPU, fix the image resolution to 512 and for each model we maximize the batch size allowed by memory for a fair comparison.

Dataset size. Vision Transformers highlighted the importance of large datasets to attain good performance for the task of image classification. At the scale of a semantic segmentation dataset, we analyze Seg-S/16 performance on ADE20k dataset in Table 6.5 when trained with a dataset of increasing size. We observe an important drop in performance when the training set size is below 8k images. This shows that even during fine-tuning transformers performs best with a sufficient amount of data.

6.3.6 Comparison with state of the art

In this section, we compare the performance of Segmenter with respect to the state-of-the-art methods on ADE20K, Pascal Context and Cityscapes datasets.

ADE20K. Seg-B†/16 pre-trained on ImageNet-1k matches the *state-of-the-art* FCN method DeepLabv3+ ResNeSt-200 [Zhang, 2020] as shown in Table 6.6. Adding our mask transformer, Seg-B†-Mask/16 improves by 2% and achieves a 50.08% mIoU, outperforming all FCN methods. Our best model, Seg-L-Mask/16 attains a state-of-the-art performance of 53.63%, outperforming by a margin of 5.27% mIoU DeepLabv3+ ResNeSt-200 and the transformer-based methods SETR [Zheng, 2020] and Swin-L UperNet [Liu, 2021b].

Method	Backbone	Im/sec	mIoU	+MS
OCR [Yuan, 2020]	HRNetV2-W48	83	-	45.66
ACNet [Fu, 2019a]	ResNet-101	-	-	45.90
DNL [Yin, 2020]	ResNet-101	-	-	45.97
DRANet [Fu, 2020]	ResNet-101	-	-	46.18
CPNet [Yu, 2020]	ResNet-101	-	-	46.27
DeepLabv3+ [Chen, 2018d]	ResNet-101	76	45.47	46.35
DeepLabv3+ [Chen, 2018d]	ResNeSt-101	15	46.47	47.27
DeepLabv3+ [Chen, 2018d]	ResNeSt-200	-	-	48.36
SETR-L MLA [Zheng, 2020]	ViT-L/16	34	48.64	50.28
Swin-L UperNet [Liu, 2021b]	Swin-L/16	34	52.10	**53.50**
Seg-B†/16	DeiT-B/16	77	47.08	48.05
Seg-B†-Mask/16	DeiT-B/16	76	48.70	50.08
Seg-L/16	ViT-L/16	33	50.71	52.25
Seg-L-Mask/16	ViT-L/16	31	51.82	**53.63**

Table 6.6: State-of-the-art comparison on ADE20K validation set.

Pascal Context Table 6.7 reports the performance on Pascal Context. Seg-B†models are competitive with FCN methods and the larger Seg-L/16 model already provides *state-of-the-art* performance, outperforming SETR-L. Performances can be further enhanced with our mask transformer, Seg-L-Mask/16, improving over the linear decoder by 2.5% and achieving a performance of 59.04% mIoU. In particular, we report an improvement of 2.8% over OCR HRNetV2-W48 and 3.2% over SETR-L MLA.

Cityscapes. Table 6.8 reports the performance of Segmenter on Cityscapes. We use a variant of mask transformer for Seg-L-Mask/16 with only one layer in the decoder as two layers did not fit into memory due to the large input resolution of 768×768. Both Seg-B and Seg-L methods are competitive with other *state-of-the-art methods* with Seg-L-Mask/16 achieving a mIoU of 81.3%.

Qualitative results. Figure 6.5 shows a qualitative comparison of Segmenter and DeepLabv3+ with ResNeSt backbone, for which models were provided by the MMSegmentation [Contributors, 2020] library. We can observe that Deeplabv3+ tends to generate sharper object boundaries while Segmenter provides more consistent labels on large instances and handles partial occlusions better.

Method	Backbone	mIoU (MS)
DeepLabv3+ [Chen, 2018d]	ResNet-101	48.5
DANet [Fu, 2019b]	ResNet-101	52.6
ANN [Zhu, 2019]	ResNet101	52.8
CPNet [Yu, 2020]	ResNet-101	53.9
CFNet [Zhang, 2019]	ResNet-101	54.0
ACNet [Fu, 2019a]	ResNet-101	54.1
APCNet [He, 2019]	ResNet101	54.7
DNL [Yin, 2020]	HRNetV2-W48	55.3
DRANet [Fu, 2020]	ResNet-101	55.4
OCR [Yuan, 2020]	HRNetV2-W48	56.2
SETR-L MLA [Zheng, 2020]	ViT-L/16	55.8
Seg-B†/16	DeiT-B/16	53.9
Seg-B†-Mask/16	DeiT-B/16	55.0
Seg-L/16	ViT-L/16	56.5
Seg-L-Mask/16	ViT-L/16	**59.0**

Table 6.7: State-of-the-art comparison on Pascal Context validation set.

Method	Backbone	mIoU (MS)
PSANet [Zhao, 2018]	ResNet-101	79.1
DeepLabv3+ [Chen, 2018d]	Xception-71	79.6
ANN [Zhu, 2019]	ResNet-101	79.9
MDEQ [Bai, 2020]	MDEQ	80.3
DeepLabv3+ [Chen, 2018d]	ResNeSt-101	80.4
DNL [Yin, 2020]	ResNet-101	80.5
CCNet [Huang, 2019]	ResNet-101	81.3
Panoptic-Deeplab [Cheng, 2020]	Xception-71	81.5
DeepLabv3+ [Chen, 2018d]	ResNeSt-200	**82.7**
SETR-L PUP [Zheng, 2020]	ViT-L/16	**82.2**
Seg-B†/16	DeiT-B/16	80.5
Seg-B†-Mask/16	DeiT-B/16	80.6
Seg-L/16	ViT-L/16	80.7
Seg-L-Mask/16	ViT-L/16	81.3

Table 6.8: State-of-the-art comparison on Cityscapes validation set.

6.4 Appendix

This appendix presents additional results. We study the impact of ImageNet pre-training on the performance and demonstrate its importance in Section 6.4.1. To gain more insight about our approach Segmenter, we analyze its attention maps and the learned class embeddings in Section 6.4.2. Finally, we give an additional qualitative comparison of Segmenter to DeepLabv3+ on ADE20K, Cityscapes and Pascal Context in Section 6.4.3.

6.4.1 ImageNet pre-training

To study the impact of ImageNet pre-training on Segmenter, we compare our model pre-trained on ImageNet with equivalent models trained from scratch. To train from scratch, the weights of the model are initialized randomly with a truncated normal distribution. We use a base learning rate of 10^{-3} and two training procedures. First, we follow the fine-tuning procedure and use SGD optimizer with "poly" scheduler. Second, we follow a more standard procedure when training a transformer from scratch where we use AdamW with a cosine scheduler and a linear warmup for $16K$ iterations corresponding to 10% of the total number of iterations. Table 6.9 reports results for Seg-S/16. We observe that when pre-trained on ImageNet-21k using SGD, Seg-S/16 reaches 45.37% yielding a 32.9% improvement over the best randomly initialized model.

Method	Pre-training	Optimizer	mIoU (SS)
Seg-S/16	None	AdamW	4.42
Seg-S/16	None	SGD	12.51
Seg-S/16	ImageNet-21k	AdamW	34.77
Seg-S/16	ImageNet-21k	SGD	**45.37**

Table 6.9: Impact of pretraining on the performance on ADE20K validation set.

6.4.2 Attention maps and class embeddings

To better understand how our approach Segmenter processes images, we display attention maps of Seg-B/8 for 3 images in Figure 6.6. We resize attention maps to the original image size. For each image, we analyze attention maps of a patch on a small instance, for example lamp, cow or car. We also analyze attention maps of a patch on a large instance, for example bed, grass and road. We observe that the attention map field-of-view adapts to the input image and the instance size, gathering global information on large instances and focusing on local information

on smaller instances. This adaptability is typically not possible with CNN which have a constant field-of-view, independently of the data. We also note there is progressive gathering of information from bottom to top layers, as for example on the cow instance, where the model first identifies the cow the patch belongs to, then identifies other cow instances. We observe that attention maps of lower layers depends strongly on the selected patch while they tend to be more similar for higher layers.

Additionally, to illustrate the larger receptive field size of Segmenter compared to CNNs, we reported the size of the attended area in Figure 6.7, where each dot shows the mean attention distance for one of the 12 attention heads at each layer. Already for the first layer, some heads attend to distant patches which clearly lie outside the receptive field of ResNet/ResNeSt initial layers.

To gain some understanding of the class embeddings learned with the mask transformer, we project embeddings into 2D with a singular value decomposition. Figure 6.8 shows that these projections group instances such as means of transportation (bottom left), objects in a house (top) and outdoor categories (middle right). It displays an implicit clustering of semantically related categories.

6.4.3 Qualitative results

We present additional qualitative results including comparison with DeepLabv3+ ResNeSt-101 and failure cases in Figures 6.9, 6.10 and 6.11. We can see in Figure 6.9 that Segmenter produces more coherent segmentation maps than DeepLabv3+. This is the case for the wedding dress in (a) or the airplane signalmen's helmet in (b). In Figure 6.10, we show how for some examples, different segments which look very similar are confused both in DeepLabv3+ and Segmenter. For example, the armchairs and couches in (a), the cushions and pillows in (b) or the trees, flowers and plants in (c) and (d). In Figure 6.11, we can see how DeepLabv3+ handles better the boundaries between different people entities. Finally, both Segmenter and DeepLabv3+ have problems segmenting small instances such as lamp, people or flowers in Figure 6.12 (a) or the cars and signals in Figure 6.12 (b).

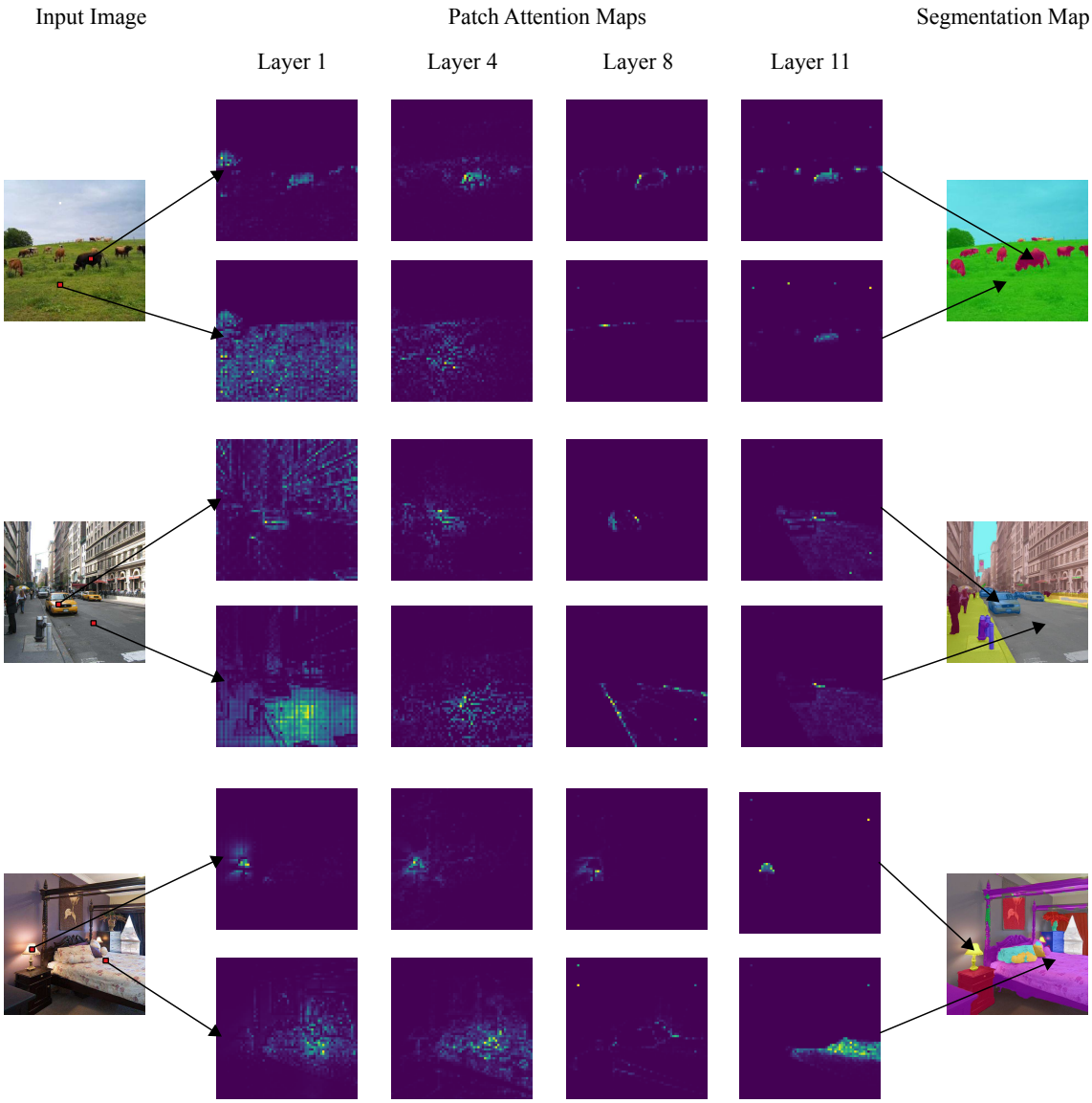


Figure 6.6: Seg-B/8 patch attention maps for the layers 1, 4, 8 and 11.

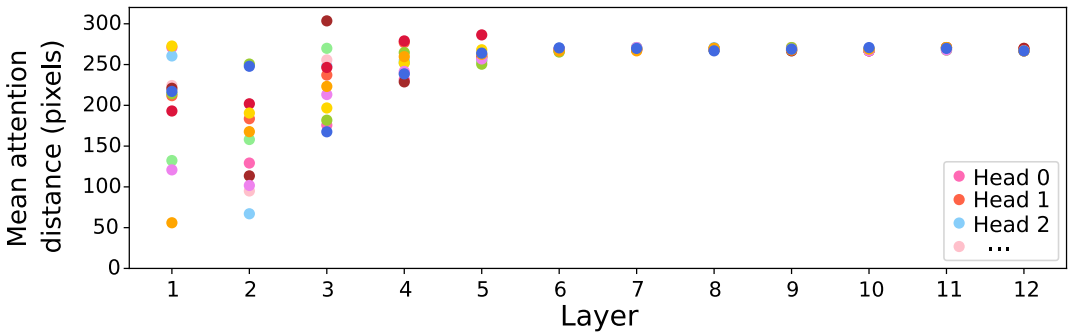


Figure 6.7: Size of attended area by head and model depth.

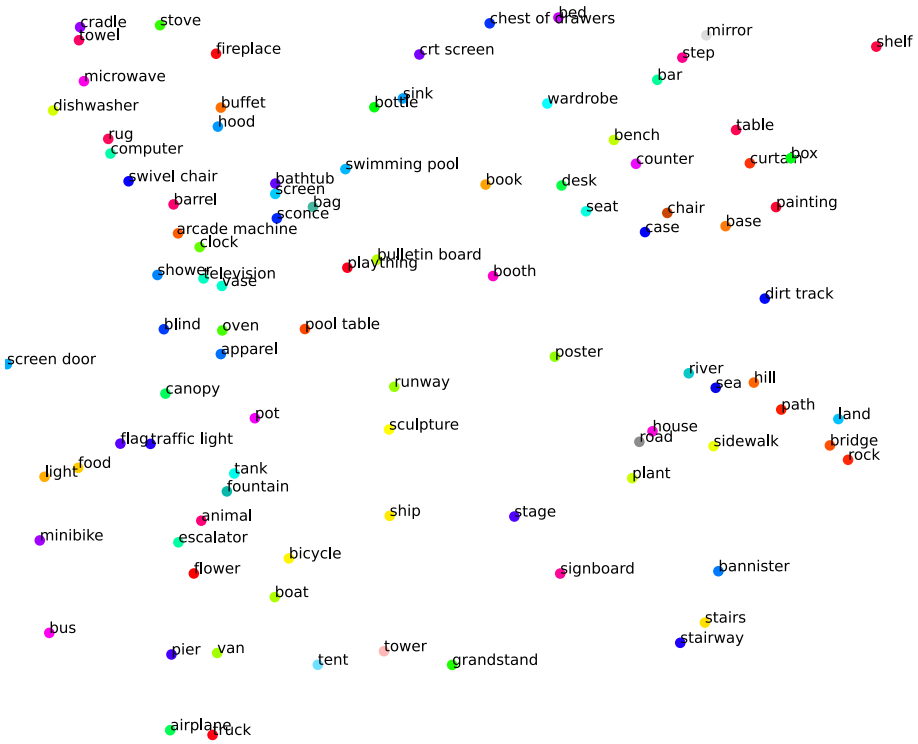


Figure 6.8: Singular value decompostion of the class embeddings learned with the mask transformer on ADE20K.

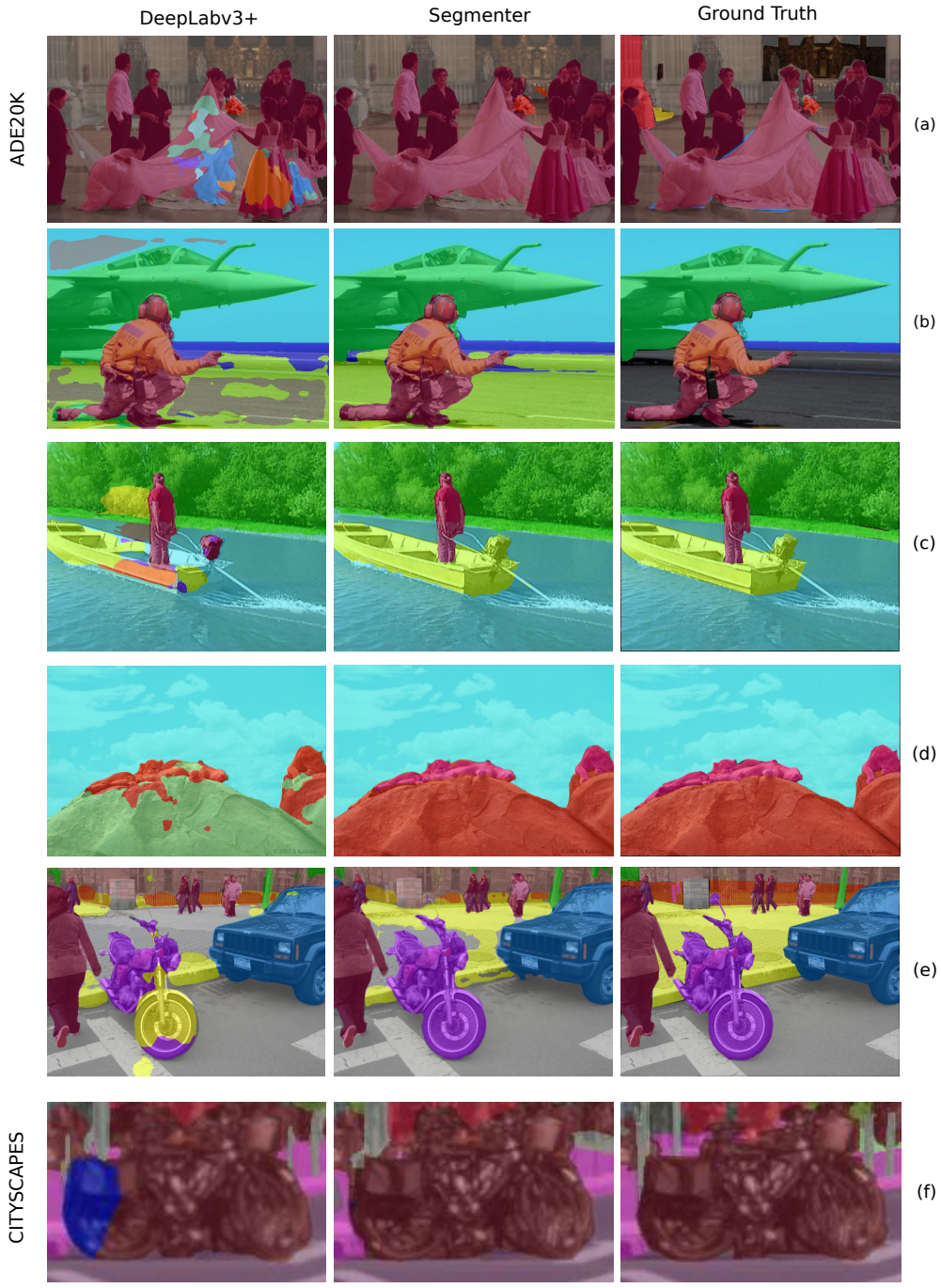


Figure 6.9: Segmentation maps where Seg-L-Mask/16 produces more coherent segmentation maps than DeepLabv3+ ResNeSt-101.

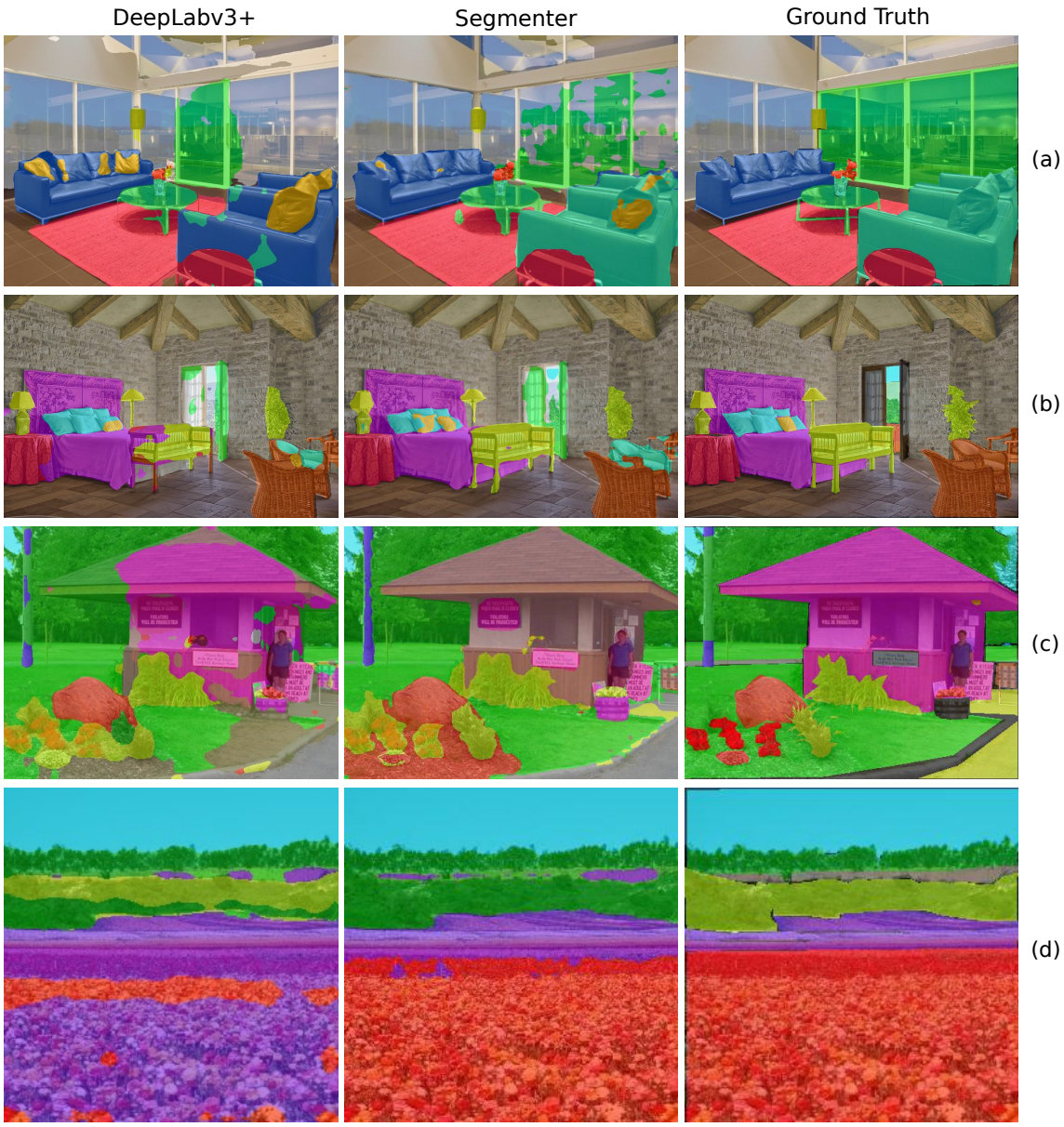


Figure 6.10: Examples for Seg-L-Mask/16 and DeepLabv3+ ResNeSt-101 on ADE20K, where elements which look very similar are confused.

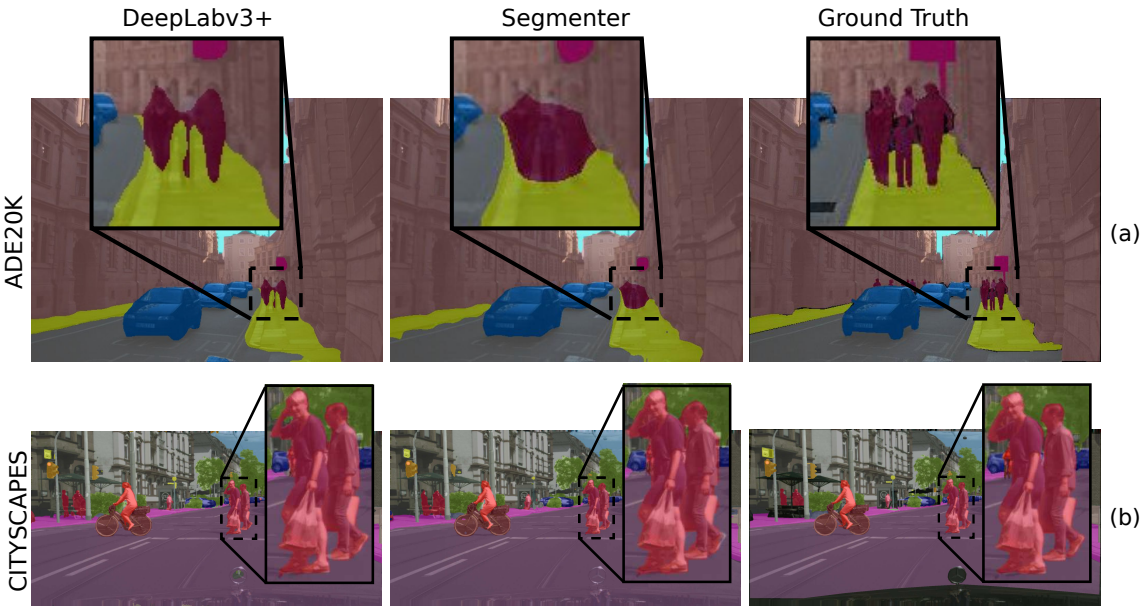


Figure 6.11: Comparison of Seg-L-Mask/16 with DeepLabV3+ ResNeSt-101 for images with near-by persons. We can observe that DeepLabV3+ localizes boundaries better.

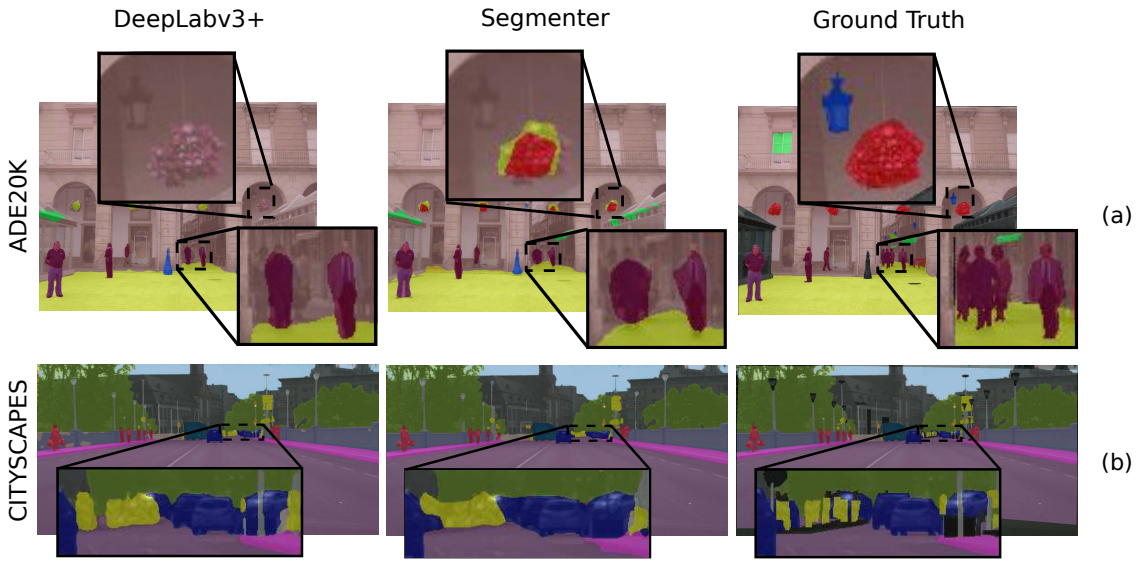


Figure 6.12: Failure cases of DeepLabV3+ ResNeSt-101 and Seg-L-Mask/16, for small instances such as (a) lamp, people, flowers and (b) cars, signals.

6.5 Conclusion

This chapter introduces a pure transformer approach for semantic segmentation. The encoding part builds up on the recent Vision Transformer (ViT), but differs in that we rely on the encoding of all images patches. We observe that the transformer captures the global context very well. Applying a simple point-wise linear decoder to the patch encodings already achieves excellent results. Decoding with a mask transformer further improves the performance. We believe that our end-to-end encoder-decoder transformer is a first step towards a unified approach for semantic segmentation, instance segmentation and panoptic segmentation.

Chapter 7

Weakly-supervised segmentation of referring expressions

The previous chapter introduces Segmenter, an approach leading to state-of-the-art performance on standard semantic segmentation benchmarks. However, relying on pixel-level annotations is extremely costly, to this end we propose a weakly-supervised method handling an open-set of labels that we detail in this chapter.

Visual grounding localizes regions (boxes or segments) in the image corresponding to given referring expressions. In this chapter we address image segmentation from referring expressions, a problem that has so far only been addressed in a fully-supervised setting. A fully-supervised setup, however, requires pixel-wise supervision and is hard to scale given the expense of manual annotation. We therefore introduce a new task of weakly-supervised image segmentation from referring expressions and propose Text grounded semantic SEGgmentation (TSEG) that learns segmentation masks directly from image-level referring expressions without pixel-level annotations. Our transformer-based method computes patch-text similarities and guides the classification objective during training with a new multi-label patch assignment mechanism. The resulting visual grounding model segments image regions corresponding to given natural language expressions. Our approach TSEG demonstrates promising results for weakly-supervised referring expression segmentation on the challenging PhraseCut and RefCOCO datasets. TSEG also shows competitive performance when evaluated in a zero-shot setting for semantic segmentation on Pascal VOC.

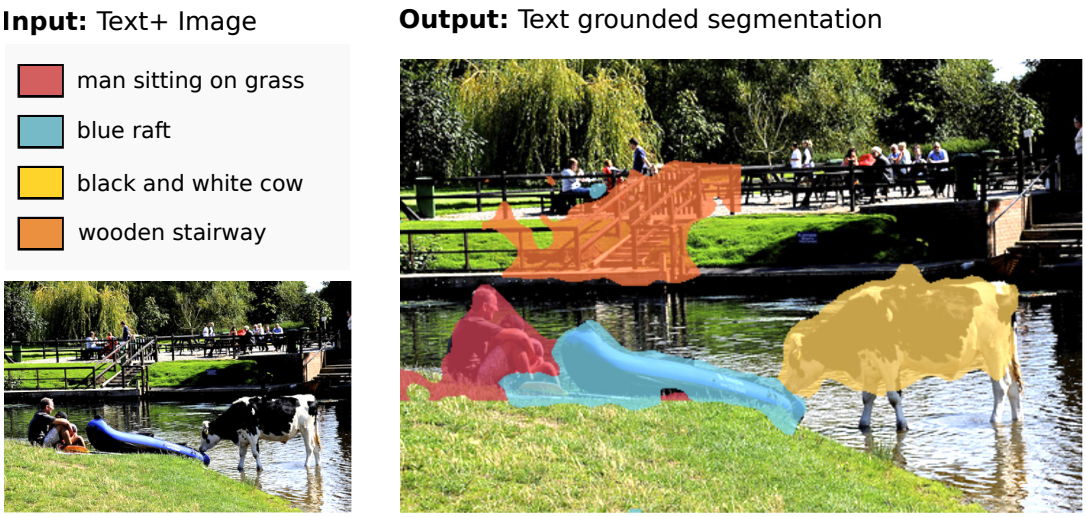


Figure 7.1: Given an image and a set of referring expressions such as *man sitting on grass* and *wooden stairway*, TSEG segments the image regions corresponding to the input expressions. Here we show results of our approach TSEG for a test image of the PhraseCut dataset. Contrary to other existing methods, TSEG only uses image-level referring expressions during training and hence does not require pixel-wise supervision.

7.1 Introduction

Image segmentation is a key component for a wide range of applications including virtual presence, virtual try on, movie post-production and autonomous driving. Powered by modern neural networks and supervised learning, image segmentation has been significantly advanced by recent work [Chen, 2018e; Cheng, 2021; Liu, 2021c; Strudel, 2021]. While most of this work addresses semantic segmentation, the more general problem of visual grounding beyond segmentation of pre-defined object classes remains open. Moreover, the majority of existing method assume full supervision and require costly pixel-wise manual labeling of training images which prevents scalability.

Manual supervision has been recognized as a bottleneck in many vision tasks including object detection [Bilen, 2014; Kantorov, 2016; Li, 2016] and segmentation [Ahn, 2018; Araslanov, 2020; Fan, 2018; Zhou, 2016], text-image and text-video matching [Miech, 2020; Radford, 2021] and human action recognition [Bojanowski, 2014; Ghadiyaram, 2019]. To this end, self-supervised methods explore regularities in images and videos and learn transferable visual representations without manual supervision [Chen, 2020b; Doersch, 2015]. Other weakly-supervised methods exploit partial and possibly noisy supervision that is either readily-available or less costly to annotate [Bilen, 2014; Miech, 2020]. In particular, weakly-supervised methods

for image segmentation avoid the costly pixel-wise annotation and limit supervision to image-level labels [Ahn, 2018; Araslanov, 2020; Fan, 2018; Zhou, 2016]. Such methods, however, remain restricted to predefined sets of classes.

A referring expression is a short text describing a visual entity such as *man sitting on grass* or *wooden stairway*, see Fig. 7.1. The task of referring expression segmentation [Hu, 2016; Yu, 2018] generalizes image segmentation from pre-defined object classes to free-form text. Given an input image and text queries (referring expressions), one should generate image segments for each referring expression. This enables segmentation using compositional referring expressions such as *man sitting on grass* and *wooden stairway*. Despite the promise of scalability, existing approaches to referring expression segmentation require pixel-wise annotation and, hence, remain limited by the size of existing datasets.

Our work aims to advance image segmentation beyond limitations imposed by the pre-defined sets of object classes and the costly pixel-wise manual annotations. Towards this goal, we propose and address the new task of *weakly-supervised referring expression segmentation*. As this task comprises difficulties of the weakly-supervised segmentation and referring expression segmentation, it introduces new challenges. In particular, existing weakly-supervised methods for image segmentation typically rely on the completeness of image-level labels, i.e., the absence of a car in the annotation implies its absence in the image. This completeness assumption does not hold for referring expression segmentation. Furthermore, the vocabulary is open and compositional.

To address the above challenges and to learn segmentation from text-based image-level supervision, we introduce a new global weighted pooling mechanism denoted as Multi-label Patch Assignment (MPA). Our method for Text grounded semantic SEGgmentation (TSEG) incorporates MPA and extends the recent transformer-based Segmenter architecture [Strudel, 2021] to referring expression segmentation. We validate our method and demonstrate its encouraging results for the task of weakly-supervised referring expression segmentation on the challenging PhraseCut [Wu, 2020] and RefCOCO [Yu, 2016b] datasets. We also evaluate TSEG in a zero-shot setting for semantic segmentation and obtain competitive performance on the Pascal VOC dataset [Everingham, 2010].

In summary, our work makes the following three contributions. (i) We introduce the new task of weakly-supervised referring expression segmentation and propose an evaluation based on the PhraseCut and RefCOCO datasets. (ii) We propose TSEG, a new method addressing weakly-supervised referring expression segmentation with

a multi-label patch assignment score. (iii) We demonstrate advantages of TSEG through a number of ablations and experimental comparisons on the challenging PhraseCut and RefCOCO datasets. Furthermore, we demonstrate competitive results for zero shot semantic segmentation on PASCAL VOC.

7.2 Related Work

Weakly-supervised semantic segmentation. Given an image as input, the goal of semantic segmentation is to identify and localize classes present in the image, e.g. annotate each pixel of the input image with a class label. Weakly-supervised Semantic Segmentation (WSS) has been introduced by [Zhou, 2016] and trains models using only image labels as supervision. Zhou *et al.* [Zhou, 2016] use Class Activation Maps (CAMs) of a Fully Convolutional Network (FCN) combined with Global Average Pooling (GAP) to obtain segmentation maps with a pooling mechanism. As CAMs tend to focus on most discriminative object parts [Wei, 2017], recent methods deploy more elaborate multi-stage approaches using pixel affinity [Ahn, 2019; Ahn, 2018], saliency estimation [Fan, 2019; Fan, 2018; Huang, 2018; Lee, 2019a; Wang, 2017; Yu, 2019] or seed and expand strategies [Huang, 2018; Kolesnikov, 2016; Wei, 2017].

While these methods provide improved segmentation, they require multiple standalone and often expensive networks such as saliency detectors [Fan, 2018; Huang, 2018; Yu, 2019] or segmentation networks based on pixel-level affinity [Ahn, 2019; Ahn, 2018]. Single-stage methods have been developed based on multiple instance learning (MIL) [Pinheiro, 2015] or expectation-maximization (EM) [Papandreou, 2015] approaches where masks are inferred from intermediate predictions. Single-stage methods have been overlooked given their inferior accuracy until the work of Araslanov *et al.* [Araslanov, 2020] that proposed an efficient single-stage method addressing the limitations of CAMs. Araslanov *et al.* [Araslanov, 2020] introduces a global weighted pooling (GWP) mechanism which we extend in this work with a new multi-label patch assignment mechanism (MPA). In contrast to prior work on weakly-supervised semantic segmentation, TSEG is a single-stage method that scales to the challenging task of referring expression segmentation.

Referring expression segmentation. Given an image and a referring expression, the goal of referring expression segmentation is to annotate the input image with a binary mask localizing the referring expression. A fully-supervised method [Hu, 2016] proposed to first combine features of a CNN with a LSTM and then decode

them with a FCN. To improve segmentation masks, [Yu, 2018] uses a two-stage method based on Mask-RCNN [He, 2017] features combined with a LSTM. To overcome the limitation of FCN to model global context and learn richer cross-modal features, *state-of-the-art* approaches [Ding, 2021; Hu, 2020; Ye, 2019] use a decoding scheme based on cross-modal attention. Despite their effectiveness, these methods are fully-supervised which limits their scalability. Several weakly-supervised approaches tackle detection tasks such as referring expression comprehension [Chen, 2018a; Gupta, 2020; Liu, 2019; Liu, 2021a; Xiao, 2017] by enforcing visual consistency [Chen, 2018a], learning language reconstruction [Liu, 2019] or with a contrastive-learning objective [Gupta, 2020]. These methods rely on an off-the-shelf object detector, Faster-RCNN [Ren, 2017], to generate region proposals and are thus limited by the object detector accuracy. None of these weakly-supervised methods address the problem of referring expression segmentation which is the focus of our work. TSEG is a novel approach that tackles weakly-supervised referring expression segmentation based on the computation of patch-text similarities with a new multi-label patch assignment mechanism (MPA).

Transformers for vision and language. Transformers [Vaswani, 2017] are now state of the art in many natural language processing (NLP) [Devlin, 2019] and computer vision [Arnab, 2021b; Cheng, 2021; Dosovitskiy, 2021a; Liu, 2021c; Strudel, 2021] tasks. Such methods capture long-range dependencies among tokens (patches or words) with an attention mechanism and achieve impressive results in the context of vision-language pretraining at scale with methods such as CLIP [Radford, 2021], VisualBERT [Li, 2019], DALL-E [Ramesh, 2021] or ALIGN [Jia, 2021]. Specific to referring expressions, MDETR [Kamath, 2021] recently proposed a method for visual grounding based on a cross-modal transformer decoder trained on a fully-supervised visual grounding task. Several methods perform zero-shot semantic segmentation with pretrained fully supervision models [Ghiasi, 2021; Xu, 2021; Zabari, 2021; Zhou, 2021]. Most similar to our work, GroupViT [Xu, 2022] relies on a large dataset of 30M image-text pairs to learn segmentation masks from text supervision, but the objective function and model architecture are different.

Our TSEG approach aims to learn patch-text associations while using only image-level annotations with referring expression. TSEG builds on CLIP [Radford, 2021] and uses separate encoders for different modalities with a cross-modal late-interaction mechanism. Its segmentation module builds on Segmenter [Strudel, 2021] which shows that interpolating patch features output by a Vision Transformer (ViT) [Dosovitskiy, 2021a] is a simple and effective way to perform semantic segmentation. Here, we extend this work to perform cross-modal segmentation. TSEG leverages a novel

patch-text interaction mechanism to compute both image-text matching scores and pixel-level text-grounded segmentation maps in a single forward pass.

7.3 Method

TSEG takes as input an image and a number of referring expressions and outputs a confidence score (Fig. 7.2, top-right) along with a segmentation mask (Fig. 7.2, bottom-right) for each referring expression. During training no segmentation masks are available and image-level labels are used to train referring expression segmentation (Fig. 7.2, top-right). TSEG is based on image patch-text matching (Fig. 7.2 left). An image encoder maps the input image to a sequence of patch tokens and a text encoder maps each input referring expression to a single text token. The tokens are then projected to a common embedding space and patch-text cosine similarities are computed as described in Section 7.3.1. To obtain an image-level score for each referring expression, the patch-text similarity matrix is summarized along the patch dimension. To do so, we introduce a novel multi-label patch assignment (MPA) mechanism described in Section 7.3.2. The model is then trained end-to-end to predict the corresponding image-text pairs as described in Section 7.3.3. At inference, the patch-text matrix is simply interpolated for patches to obtain pixel-level masks as described in Section 7.3.3. The choice of an appropriate global pooling mechanism is important to learn accurate segmentation maps as illustrated in Figure 7.3. We evaluate its impact in Section 7.4 and show that the novel multi-label patch assignment mechanism outperforms existing ones by a significant margin.

7.3.1 Patch-text similarity matrix

In this section we describe how to compute the similarity matrix between patches of an image and several referring expressions. We consider an image represented by N patches $p_1, ..., p_N$ and a set of L referring expressions $t_1, ..., t_L$. Patches are encoded by tokens $(\mathbf{x}_1, ..., \mathbf{x}_N)$, each referring expression consists of several words and is encoded by one token $(\mathbf{y}_1, ..., \mathbf{y}_L)$. The resulting similarity matrix is $\mathbf{S} = (\mathbf{x}_i \cdot \mathbf{y}_j)_{i,j} \in \mathbb{R}^{N \times L}$. See Figure 6.2 left.

Image encoder. An image $I \in \mathbb{R}^{H \times W \times C}$ is split into a sequence of patches of size (P, P). Each image patch is then linearly projected and a position embedding is added to produce a sequence of patch tokens $(p_1, ..., p_N) \in \mathbb{R}^{N \times D_I}$ where $N = HW/P^2$ is the number of patches, D_I is the number of features. A transformer encoder maps the input sequence to a sequence of contextualized patch tokens

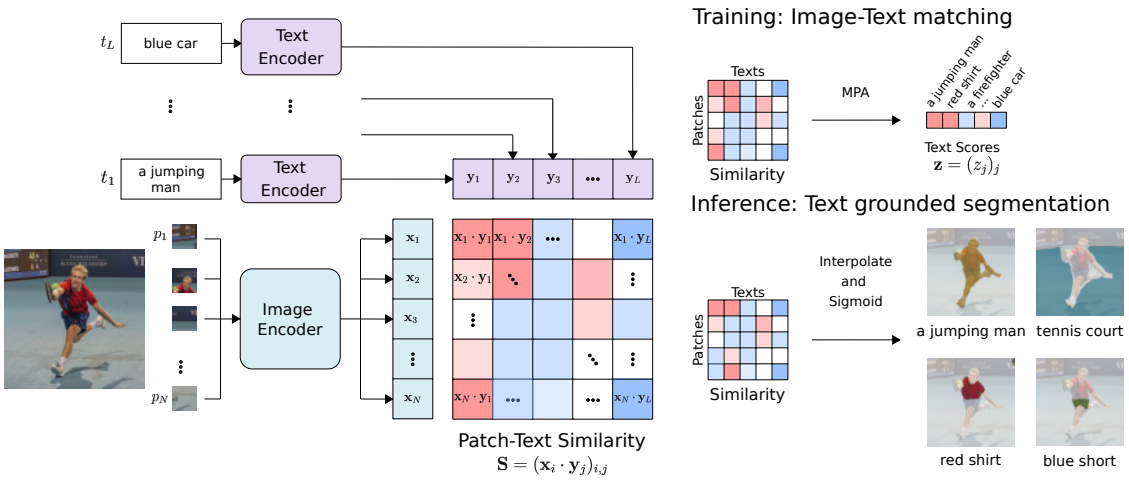


Figure 7.2: Overview of our approach TSEG. (Left) Image patches and referring expressions are mapped with transformers to patch and text embeddings and then compared by computing patch-text cosine similarity scores. (Right - Training) Our global pooling mechanism with multi-label patch assignment (MPA) reduces patch-text similarity scores to image-level labels to train the model for referring expression classification. (Right - Inference) Sequences of patch scores (columns) are rearranged into 2D masks and bilinearly interpolated to obtain pixel-level referring expression masks.

$(\mathbf{x}_1, ..., \mathbf{x}_N) \in \mathbb{R}^{N \times D_I}$. See more details in Section 7.4.2.

Text encoder. For each referring expression t_j, which can consist of multiple words, we extract one token $\mathbf{y}_j$. To do so the text t_j is tokenized into words using lower-case byte pair encoding (BPE) [Sennrich, 2016] and [BOS], [EOS] tokens are added to the beginning and the end of the sequence. A sequence of position embedding is added and a transformer encoder maps the input sequence to a sequence of contextualized word tokens from which the [BOS] token is extracted to serve as a global text representation $\mathbf{y}_j \in \mathbb{R}^{D_T}$.

Patch-text similarity scores. The visual and textual tokens are linearly projected to a multi-modal common embedding space and L^2-normalized. From the patch tokens $(\mathbf{x}_1, ..., \mathbf{x}_N)$ and the global text tokens $(\mathbf{y}_1, ..., \mathbf{y}_L)$, we compute patch-text cosine similarities as the scalar product and obtain the similarity matrix

$$\mathbf{S} = (s_{i,j})_{i,j} = (\mathbf{x}_i \cdot \mathbf{y}_j)_{i,j}, \tag{7.1}$$

with $\mathbf{S} \in \mathbb{R}^{N \times L}$. The similarities are in the range $[-1, 1]$ and scaled with a learnable temperature parameter $\tau > 0$ controlling their range.

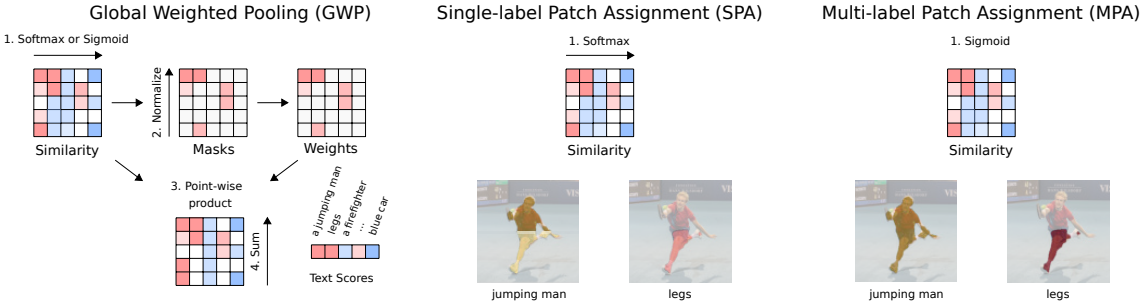


Figure 7.3: (Left) A patch assignment mechanism computes masks from patch-text similarities, the masks are used as weights in the global weighed pooling. (Center) SPA: assignment with a softmax on text channels, softly enforcing a single label per patch. (Right) MPA: assignment with a sigmoid, generalizing to multiple labels per patch.

7.3.2 Global Pooling Mechanisms

To leverage image-level text supervision, we need to map the matrix $\mathbf{S} \in \mathbb{R}^{N \times L}$ of patch-text similarities to an image-level score for each referring expression, i.e., $\mathbf{z} \in \mathbb{R}^{L}$. The score vector $\mathbf{z}$ allows us to compute a classification loss using ground truth referring expressions. Note that we cannot compute per-pixel losses given the lack of pixel-wise supervision in weakly-supervised settings.

Global average and max pooling (GAP-GMP). A straightforward way of pooling is global average pooling (GAP), where we average the similarities for a given referring expression over all patches of an image:

$$z_j^{GAP} = \frac{1}{N} \sum_{i=1}^{N} s_{i,j}. \tag{7.2}$$

This score is expected to be high if the referring expression is contained in the image. However, the score is dependent on the object size and results in low scores for small objects. An alternative to GAP is global max pooling (GMP):

$$z_j^{GMP} = \max_i (s_{i,j}). \tag{7.3}$$

The max operation in GMP decreases the influence of the object size, however it tends to focus on most discriminative regions of a class [Wei, 2017].

Global weighted pooling (GWP). To address the shortcomings of GAP and GMP, we follow [Araslanov, 2020] and make use of weighted pooling. Global weighted pooling replaces the constant patch weights $1/N$ in the sum of GAP by weights $\mathbf{W} = (w_{i,j})_{i,j} \in \mathbb{R}^{N \times L}$. The final score of a referring expression is then the

weighted average of similarities:

$$z_j^{GWP} = \sum_{i=1}^{N} w_{i,j} s_{i,j}, \tag{7.4}$$

as illustrated in Figure 7.3 left. In practice, $\mathbf{W}$ is defined in terms of spatially normalized mask scores $\mathbf{M} = (m_{i,j})_{i,j} \in \mathbb{R}^{N \times L}$, based on $w_{i,j} = m_{i,j}/(\sum_i m_{i,j} + \varepsilon)$ where $\varepsilon > 0$ allows for $\sum_i w_{i,j} = 0$ when mask scores are below a threshold. GAP is a particular case of GWP where $m_{i,j} = 1$ for all i, j and $\varepsilon = 0$. We next describe two methods to compute masks $\mathbf{M}$ from the similarity matrix $\mathbf{S}$.

Masks by single-label patch assignment (SPA) [Araslanov, 2020]. We aim at assigning patches to the relevant referring expression. To do so, we apply a softmax operation over all referring expressions $(\mathbf{y}_1, ..., \mathbf{y}_L)$ for each patch $\mathbf{x}_i$:

$$m_{i,j}^{SPA} = \frac{e^{s_{i,j}}}{e^{s_{bg}} + \sum_{j=1}^{L} e^{s_{i,j}}}. \tag{7.5}$$

We add a background column $(s_{i,0})_i$ and assign it a constant equal to $s_{bg} = 0$ for all patches $\mathbf{x}_i$. This allows to assign patches with low scores $s_{i,j} < 0$ to the background. The masks are then soft assignments with $\sum_{j=0}^{L} m_{i,j} = 1$ for any patch i. This patch assignment can be viewed as multi-class classification which is typical for semantic segmentation where one pixel is matched to a *single label* as proposed by [Araslanov, 2020].

This single-label patch assignment (SPA) is illustrated in Figure 7.3 center. The softmax operation over referring expressions softly enforces the correspondence of a patch to one expression. However, this definition is problematic for referring expression segmentation where the masks of several expressions can overlap. We illustrate this in Figure 7.3 center where pixels corresponding to *jumping man* and *legs* have lower mask weights on the overlapping region. Such lower mask weights result in decreased image-level scores for both of the expressions.

Masks by multi-label patch assignment (MPA). We propose multi-label patch assignment (MPA) that overcomes the above limitations of SPA. For each patch $\mathbf{x}_i$, we rely on binary classification between a referring expression $\mathbf{y}_j$ and the background based on:

$$m_{i,j}^{MPA} = \frac{e^{s_{i,j}}}{e^{s_{bg}} + e^{s_{i,j}}}. \tag{7.6}$$

In this case, each patch can be assigned to multiple referring expressions, see Figure

7.3 right. The masks are not mutually exclusive and each referring expression can be assigned a score $m_{i,j}^{MPA} \in [0,1]$ without softmax imposed constraints. Patch assignment is viewed as a multi-label classification problem, this property is highly beneficial when performing weakly-supervised referring expression segmentation, as shown in Section 7.4.

Image-text scores. We compute GWP scores $\mathbf{z}^{GWP}$ with (7.4) using the masks $\mathbf{M}$ defined according to one of the assignment mechanism defined in (7.5), (7.6). Then, we compute mask size scores $\mathbf{z}^{size}$ as

$$z_j^{size} = (1 - \overline{m}_j)^p \log(\lambda + \overline{m_j}), \tag{7.7}$$

with $\overline{m_j} = \frac{1}{N}\sum_{i=1}^{N} m_{i,j}$. This $\mathbf{z}^{size}$ is a size-penalty term introduced by [Araslanov, 2020] to enforce mask completeness, e.g. $z_j^{size} < 0$ for small masks. The magnitude of this penalty is controlled by λ. Due to the normalization, $\mathbf{W}$ used in GWP is invariant to the masks size $\mathbf{M}$ and $\mathbf{z}^{size}$ enforces masks to be complete. The final score defining the presence of a referring expression t_j in the image is defined as the sum:

$$z_j = z_j^{GWP} + z_j^{size}. \tag{7.8}$$

7.3.3 Training and inference

In the following we describe our weakly supervised and fully supervised training procedure. Furthermore, we present the approach used for inference.

Weakly-supervised learning. Weakly-supervised segmentation is usually addressed on datasets with a fixed number of classes. To handle the more general case where visual entities in the image are defined by referring expressions we use referring expressions of samples in a mini-batch as positive and negative examples. Given a mini-batch containing (image, referring expression) pairs, the model has to predict the subset of referring expressions present in each image. For each image, we extract image-text scores $\mathbf{z} \in \mathbb{R}^L$ from the similarity matrix $\mathbf{S}$ using one of the pooling mechanism described in the previous section. Finally, we optimize over the scores to match ground truth pairings $\bar{\mathbf{z}}$ with the multi-label soft-margin loss function [Ahn, 2018; Araslanov, 2020; Wei, 2018] as a classification loss,

$$\mathcal{L}_{cls}(\mathbf{z}, \bar{\mathbf{z}}) = \sum_{j=1}^{L} -\bar{z}_j \log\left(\sigma(z_j)\right) - (1 - \bar{z}_j) \log\left(\sigma(-z_j)\right),$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. The loss encourages $z_j > 0$

for positive image-text pairs and $z_j < 0$ for negative pairs.

Fully-supervised learning. In the fully-supervised case, segmentation is learned from a dataset of images annotated with referring expressions and their corresponding segmentation masks. Only positive referring expressions $(\mathbf{y}_1, ..., \mathbf{y}_L)$ are passed to the text encoder and the similarity matrix $\mathbf{S}$ is bilinearly interpolated to obtain pixel-level similarities of shape $\mathbb{R}^{H \times W \times L}$. Then, we minimize the Dice loss between the sigmoid of the pixel-level similarities $\mathbf{M} = \sigma(\mathbf{S})$ and the ground truth masks $\overline{\mathbf{M}}$:

$$\mathcal{L}_{dice}(\mathbf{M}, \overline{\mathbf{M}}) = 1 - 2\frac{|\mathbf{M} \cap \overline{\mathbf{M}}|}{|\mathbf{M}| + |\overline{\mathbf{M}}|}, \tag{7.9}$$

where $|\mathbf{M}| = \sum_{i,j} m_{i,j}$ and $\mathbf{M} \cap \overline{\mathbf{M}} = (m_{i,j}\overline{m}_{i,j})_{i,j}$.

Inference. To produce segmentation masks, we reshape the patch-text masks $\mathbf{M} \in \mathbb{R}^{N \times L}$ into a 2D map and bilinearly interpolate it to the original image size to obtain pixel-level masks of shape $\mathbb{R}^{H \times W \times L}$. For SPA, pixel annotations are obtained by adding a background mask to $\mathbf{M}$ and applying an argmax over the referring expressions. For MPA, we threshold the values of $\mathbf{M}$ using the background score. For GAP and GMP, we follow the standard approach from [Ahn, 2018] to compute the masks $\mathbf{M}$. Directly interpolating patch-level similarity scores to generate segmentation maps has been proven effective by Segmenter [Strudel, 2021] in the context of semantic segmentation. Our decoding scheme is an extension of Segmenter linear decoding where the set of fixed class embeddings is replaced by text embeddings.

7.4 Experiments

In this section we first outline datasets and implementation details in Sections 7.4.1 and 7.4.2. We then validate our implementation of two *state-of the-art* methods for weakly-supervised semantic segmentation in Section 7.4.3. Next, we ablate different parameters of the proposed TSEG method for the task of referring expression segmentation in Section 7.4.4. Finally, we compare TSEG to methods introduced in Section 7.4.3 on referring expression datasets in Section 7.4.5.

7.4.1 Datasets and metrics

Pascal VOC 2012. Pascal [Everingham, 2010] is an established benchmark for weakly-supervised semantic segmentation. Following standard practice [Ahn, 2019; Ahn, 2018; Araslanov, 2020; Kolesnikov, 2016], we augment the original training

data with additional images from [Hariharan, 2011]. The dataset contains 10.5K images for training and 1.5K images for validation.

PhraseCut. PhraseCut [Wu, 2020] is the largest referring expression segmentation dataset with 77K images annotated with 345K referring expressions from Visual Genome [Krishna, 2017]. The expressions comprise a wide vocabulary of objects, attributes and relations. The dataset is split into 72K images, 310K expressions for training and 3K images, 14K expressions for validation.

RefCOCO. RefCOCO and RefCOCO+ [Yu, 2016b] are the two most commonly used datasets for referring expression segmentation and comprehension. RefCOCO has 20K images and 142K referring expressions for 50K objects while RefCOCO+ contains 20k images and 142K expressions for 50K objects. RefCOCO+ is a harder dataset where words related to the absolute location of the objects are forbidden. RefCOCOg is a dataset of 27K images with 105K expressions referring to 55K objects. Compared to RefCOCO(+), RefCOCOg has longer sentences and richer vocabulary.

Metrics. We follow previous work and report mean Intersection over Union (mIoU) for all Pascal classes. For referring expression segmentation we use standard metrics where mIoU is the IoU averaged over all image-region pairs resulting in a balanced evaluation for small and large objects [Yu, 2016b; Wu, 2020].

7.4.2 Implementation details

Initialization. Our TSEG model contains an image encoder initialized with an ImageNet pretrained Vision Transformer [Dosovitskiy, 2021a; Steiner, 2021] and a text encoder initialized with a pretrained BERT model [Devlin, 2019]. We use ViT-S/16 [Steiner, 2021] and BERT-Small [Turc, 2019] which are both expressive models achieving strong performance on vision and language tasks, while remaining fast and compact. Our model has a total number of 42M parameters. Following [Dosovitskiy, 2021a; Strudel, 2021], we bilinearly interpolate ViT position embeddings when using an image resolution that differs from its pretraining.

Optimization. For weakly-supervised learning, we use SGD optimizer [Robbins, 1951] with a base learning rate γ_0, and set weight decay to 10^{-4}. Following DeepLab [Chen, 2018e], we adopt the poly learning rate decay $\gamma = \gamma_0(1-\frac{n_{iter}}{n_{total}})^{0.9}$. We use a stochastic drop path rate [Huang, 2016] of 0.1 following standard practices to train transformers [Devlin, 2019; Dosovitskiy, 2021a; Steiner, 2021]. For Pascal, PhraseCut and RefCOCO, we set the base learning rate $\gamma_0 = 10^{-3}$. We found this learning scheme

Method	Image encoder	Class encoding		mIoU
		Vector	Language model	
GAP [Ahn, 2018]	WideResNet38	✓	✗	48.0
GAP [Ahn, 2018]†	WideResNet38	✗	✓	46.8
GAP [Ahn, 2018]†	ViT-S/16	✗	✓	50.2
GMP [Zhou, 2016]†	WideResNet38	✗	✓	44.3
GMP [Zhou, 2016]†	ViT-S/16	✗	✓	48.1
SPA [Araslanov, 2020]	WideResNet38	✓	✗	62.7
SPA [Araslanov, 2020]†	WideResNet38	✗	✓	62.4
SPA [Araslanov, 2020]†	ViT-S/16	✗	✓	**66.4**

Table 7.1: State-of-the-art single-stage methods for weakly-supervised semantic segmentation on the Pascal VOC validation set. † denotes our implementation. Multi-scale processing and CRF are used for inference.

to be stable resulting in good results for all three datasets. Regarding training iterations and the batch size, we use 16K iterations and batches of size 16 for Pascal, 80K iterations and batches of size 32 for RefCOCO, and 120K iterations with batches of size 32 for PhraseCut. When training on referring expressions, we randomly sample three positive expressions per image on average. The resolution of images at train time is set to 384×384 and following standard practices we use random rescaling, horizontal flipping and random cropping.

For the fully-supervised setup we use AdamW [Kingma, 2015b; Loshchilov, 2019] optimizer and set the base learning rate γ_0 to 5×10^{-5}. We set the batch size to 16 for all datasets and use the same number of iterations as for weakly-supervised setups. The resolution of images at train time is 512×512.

7.4.3 State-of-the-art methods for weakly-supervised semantic segmentation

As we are the first to propose an approach for weakly-supervised learning for referring expression segmentation, we implemented state-of-the-art methods for weakly-supervised semantic segmentation to use as baselines. We use three single-stage methods presented in Section 7.3.2, namely GMP [Zhou, 2016], the seminal work GAP [Ahn, 2018], and the more recent state-of-the-art approach SPA [Araslanov, 2020]. SPA performs close to the best two-stage weakly-supervised methods, DRS [Kim, 2021] and EPS [Lee, 2021b], two more complex methods relying on off-the-shelf saliency detectors, which is not the focus of our work.

Table 7.1 reports the performance on the Pascal VOC 2012 dataset. With a language

$\lambda \downarrow$ $p \rightarrow$	0	1	3	5
0.0	26.8	27.4	27.9	27.7
0.01	26.8	26.8	27.6	**28.3**
0.1	26.3	26.8	27.2	28.0

(a) Size penalty term.

Dimension	mIoU
384	28.3
512	28.6
1024	**28.8**

(b) Multi-modal embedding dimension.

Similarity	mIoU
identity	**28.8**
tf-idf	28.4

(c) Ground truth similarity score.

Dataset Size	mIoU
10%	16.2
50%	25.3
100%	28.8

(d) Dataset size.

Dataset	mIoU
ImageNet	28.8
COCO	**31.7**

(e) Pretraining dataset.

Table 7.2: Ablations of TSEG with ViT-S/16 as the image encoder and Bert-Small as the language model on PhraseCut validation set.

model as class encoding as shown in Figure 6.2, we obtain similar performances as GAP [Ahn, 2018] and SPA [Araslanov, 2020] using the same WideResNet38 backbone. By using the more recent ViT-S/16 backbone with SPA, we obtain 66.4% mIoU, a 4% gain over WideResNet38. We also report results with GMP [Zhou, 2016] for which we did not find methods reporting results on Pascal VOC 2012. The GMP results are below the GAP results and again the ViT-S/16 backbone gives improved results. In the following sections we use ViT-S/16 as the image encoder, BERT-Small as the text encoder and GAP, GMP and SPA as a point of comparison to our proposed TSEG method. The models can directly be used to perform referring expression segmentation by replacing the class label given as input to the language model by referring expressions.

7.4.4 TSEG ablations

We now perform weakly-supervised referring expression segmentation. At train time the model has to maximize the score of the image and text embeddings of correct pairings while minimizing the score of incorrect pairings. At test time, following the standard visual grounding setting, the model is given as input the set of referring expressions present in the image and outputs a mask for each referring expression. TSEG uses the proposed MPA to compute scores from patch-text similarities.

Table 7.2 reports ablations of our TSEG model on the PhraseCut validation set. First, we ablate over the size penalty parameters λ and p from Eq. 7.7 in Table 7.2a. Smaller λ values induce a larger penalty for masks with a small size and larger p values increase the focal penalty term, see [Lin, 2020] for more details. We

Method	PhraseCut	RefCOCO	RefCOCO+	RefCOCOg
GMP [Zhou, 2016]†	5.77	6.54	5.12	6.54
GAP [Ahn, 2018]†	9.35	6.65	7.21	6.07
SPA [Araslanov, 2020]†	21.12	10.32	9.16	8.35
TSEG	28.77	25.44	22.01	22.05
TSEG (CRF)	**30.12**	**25.95**	**22.62**	**23.41**

Table 7.3: Comparison of different weakly-supervised methods for referring expression segmentation on Phrasecut and RefCOCO validation set. † denotes our implementation, validated in Table 7.1.

find TSEG is quite robust to the objective hyperparameters λ and p. The best values are $\lambda = 0.01$ and $p = 5$; we fix λ and p to these values in the remaining of the chapter. Table 7.2b reports performance for different cross-modal embedding dimension, increasing the embedding size improves results overall. In Table 7.2c, we consider different definitions for the ground truth. In the *identity* setup, two referring expressions of a batch are considered the same if they exactly match. In the *tf-idf* setup, the similarity between two referring expressions if computed according to a tf-idf score. If a *tabby cat* is present in an image, and there is a *brown cat* in a second image, the ground truth score for *brown cat* in the first image will be positive because both referring expression share the word *cat*. Using tf-idf performed slightly worse than the identity score, and we thus use *identity* to define the ground truth. Table 7.2d reports the validation score for an increasing training dataset size. We observe that TSEG improves with the dataset size, a desirable property of weakly-supervised segmentation approach where annotations are much cheaper to collect than in the fully-supervised case. Finally, Table 7.2e reports results when pretraining the visual backbone on only ImageNet for classification or by additionally pretraining the visual and language model on RefCOCO for visual grounding. For pretraining on COCO we use box ground truth annotations as follows. The model is given as input an image and referring expressions to detect, for each referring expressions the model predicts patches that are within the object bounding box. We observe that leveraging detection related information as pretraining improves the result by 3%. In the following we report results with ImageNet pretraining only, following standard practice from the weakly-supervised semantic segmentation literature.

7.4.5 Weakly supervised referring expression segmentation

We now compare TSEG on referring expression datasets to weakly supervised *state-of-the-art* methods presented in Section 7.4.3, we report results in Table 7.3 and

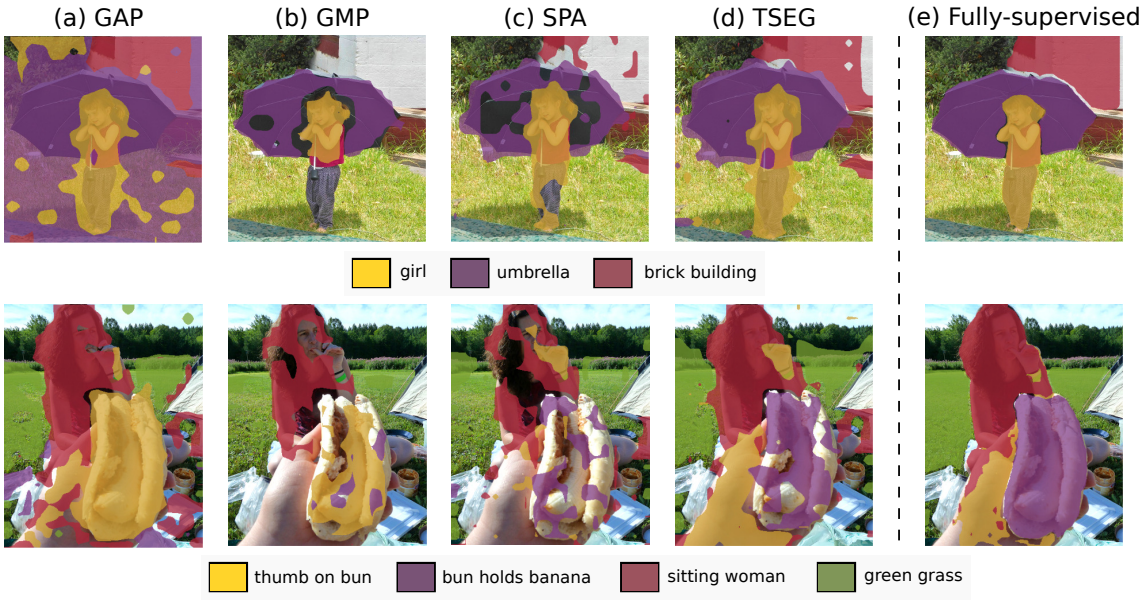


Figure 7.4: Comparison of different pooling mechanisms for weakly supervised segmentation from referring expressions on example images from the PhraseCut dataset: (a) Global average pooling (GAP), (b) Global max pooling (GMP), (c) Single-label patch assignment (SPA), (d) TSEG with multi-label patch assignment (TSEG). (e) Fully supervised results.

show qualitative results in Figure 7.4.

PhraseCut: GMP and GAP achieve an mIoU of 5.7 and 9.3 respectively, showing that it is possible to learn meaningful masks using referring expressions as labels. However, GAP averages patch-text similarity scores and depends on the instance mask size which tends to generate over-saturated activation maps (Fig. 7.4a). GMP exhibits complementary properties focusing on the most discriminative object parts (Fig. 7.4b). SPA outperforms GAP and GMP with a mIoU of 21.1, consistent with results on Pascal VOC 2012, see Table 7.1. MPA further improves SPA by 7%, with 28.77% mIoU on Phrasecut, showing its crucial importance for referring expression segmentation. This improvement can partly be explained by the fact that our objective allows multiple masks to overlap by design, a highly desirable property that is not satisfied by GMP, GAP and SPA. From Figure 7.4d we observe that MPA generates more complete masks with both higher recall, e.g. the *thumb on bun* instance is detected, and we obtain higher precision, e.g. masks achieve better completeness as for the *sitting woman* instance. Using CRF [Chen, 2018c] further improves the performance to 30.12 mIoU. Qualitative results are presented in Figure 7.5.

To obtain an upper-bound, we also train TSEG with full supervision and obtain

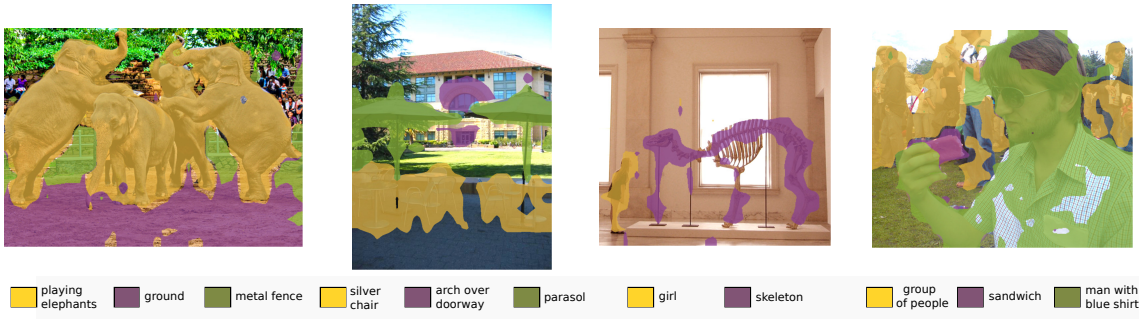


Figure 7.5: TSEG segmentation results on the PhraseCut test set. Our method segments a rich set of open-vocabulary concepts without using pixel-level supervision at the training.

a 49.6 mIoU. This is close to the best fully supervised method MDETR [Kamath, 2021], which obtains 53.1 mIoU while pretraining on a much larger dataset annotated for visual grounding and higher training resolution. While there is still a gap compared to full supervision, we believe our proposed results to be promising and the first step towards large-scale weakly supervised referring expression segmentation. Additional qualitative results and comparison to the fully-supervised model are presented in the appendix.

RefCOCO: We also evaluate our method on the three RefCOCO datasets and report results on the *val* split in Table 7.3. Again, MPA outperforms GMP, GAP and SPA by a large margin. Training TSEG with full supervision we obtain 66.00 mIoU on RefCOCO, 55.35 on RefCOCO+ and 54.71 on RefCOCOg. This is slightly better than the best fully supervised method VLT [Ding, 2021], which obtains 65.65, 55.50 and 52.99 mIoU respectively. There is a larger gain from using full supervision than on PhraseCut. This could be explained by more fine-grained referring expressions such as *broccoli stalk that is pointing up and is touching a sliced carrot* or *a darker brown teddy bear in a row of lighter teddy bears* that are harder to localize without pixel-level supervision.

7.4.6 Zero-shot transfer on Pascal VOC

We evaluate the ability of TSEG to detect and localize visual concepts from text supervision by performing zero-shot experiments on Pascal VOC 2012 dataset, see Fig. 7.6. We take our TSEG model trained on the PhraseCut dataset, i.e., with the text supervision based on the referring expressions from PhraseCut. We, then, pass the names of Pascal classes as input to the text encoder and obtain segmentation masks and confidence scores for all 20 object classes in each image. We filter classes by thresholding with the model confidence scores then use argmax between the

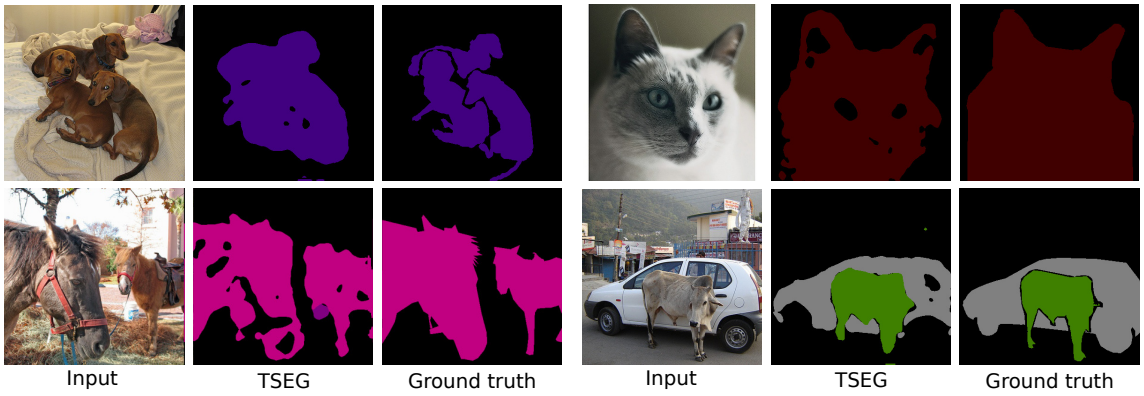


Figure 7.6: Zero-shot transfer of our approach TSEG trained from text supervision on PhraseCut and evaluated on Pascal VOC 2012. The method has not been explicitly trained for PASCAL classes and has never obtained pixel-level supervision.

remaining masks to determine the class of each pixel. We set the threshold to 0.5.

In the zero-shot setting, our TSEG model achieves an mIoU of 48.5 while the SPA baseline achieves an mIoU of 43.5. Interestingly, TSEG performs well on all classes except the *person* class. As can be observed from Figure 7.7, the model does not detect the *person* label, but can be improved with label engineering by using more specific labels for the text encoder, such as *woman* and *rider*. This bias partly comes from the annotations of PhraseCut training set, and we believe that the need for label engineering may be reduced by training TSEG on a larger dataset with richer text annotations. On the person class, by passing *person* as input to the text encoder we obtain an IoU of 0.6 while by merging masks for the words *man, woman, men, women, child, boy, girl, baby* we improve the IoU to 30.4. By performing label engineering, TSEG reaches 50.3 mIoU. In comparison, GroupViT [Xu, 2022] reports an mIoU of 51.2, but it has been trained on a much larger dataset of 30M image-text pairs and was designed for zero-shot segmentation. TSEG performs comparably to GroupViT, while trained on 350k image-text pairs. This demonstrates the ability of our approach to learn general visual concepts accurately.

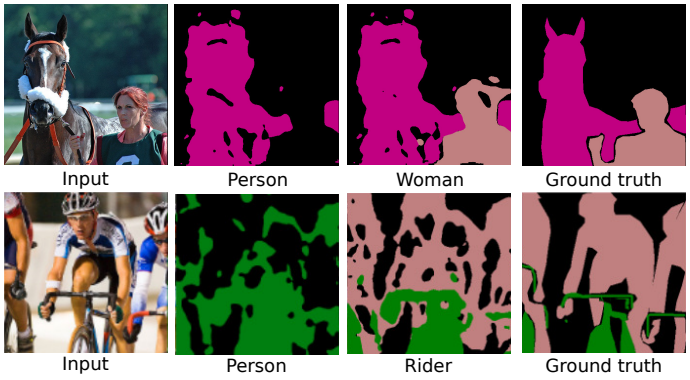


Figure 7.7: Failure cases on the person class for zero-shot results on Pascal VOC 2012. While the horse (violet) or bicycle (green) are well localized, the class person (pink) is not detected with the *person* label (column 2). The model detects it by using more specific labels such as *rider* or *woman* (column 3, pink). Column 4 shows the ground truth.

7.5 Acknowledgements

This work was partially supported by the HPC resources from GENCI-IDRIS (Grant 2021-AD011011163R1), the Louis Vuitton ENS Chair on Artificial Intelligence, and the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

7.6 Qualitative results

We present additional qualitative results in Figures 7.8 and 7.9. In particular, we compare TSEG trained with weak supervision to the same model trained with full supervision in Figure 7.8. TSEG captures cloth related concepts, animals and parts of the bodies reasonably well, however it can fail at capturing colors, distinguish between a book and a laptop, or between a blue jean and different type of trousers. In Figure 7.9, we observe that TSEG captures a rich variety of visual concepts, even rarely occurring ones quite accurately.

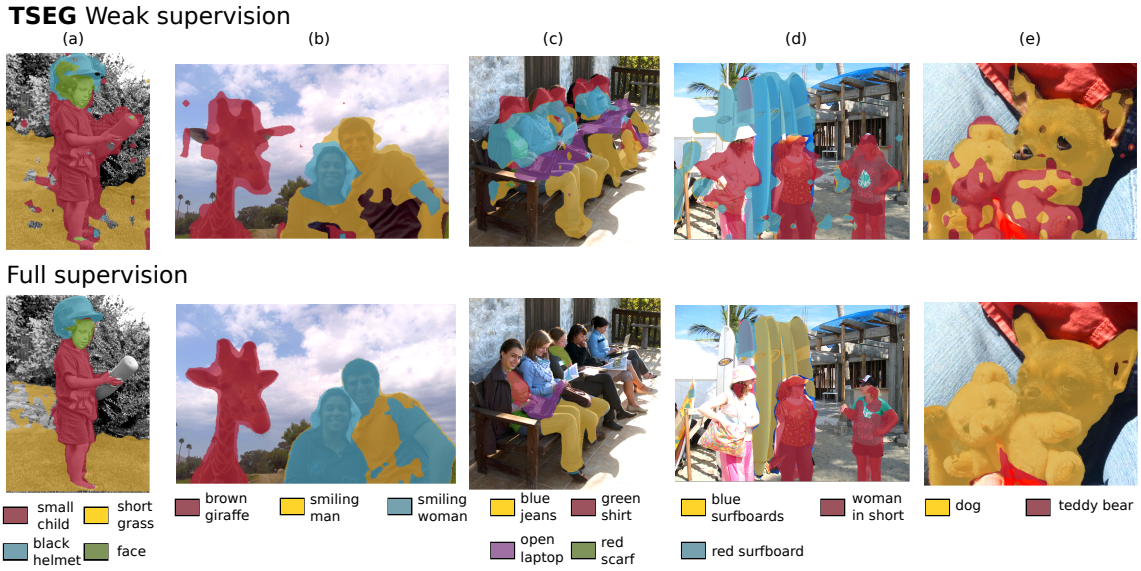


Figure 7.8: Comparison of TSEG to fully-supervised results on PhraseCut validation set. (a) Both methods perform well. (b) Both approaches do not distinguish well man and woman. (c-d) TSEG captures coarse semantic meaning such as legs (c) or surfboards (d) but misses the difference between a book and a laptop (c) or color attributes (d). (e) TSEG distinguishes the teddy bear and dog better than the fully-supervised model.

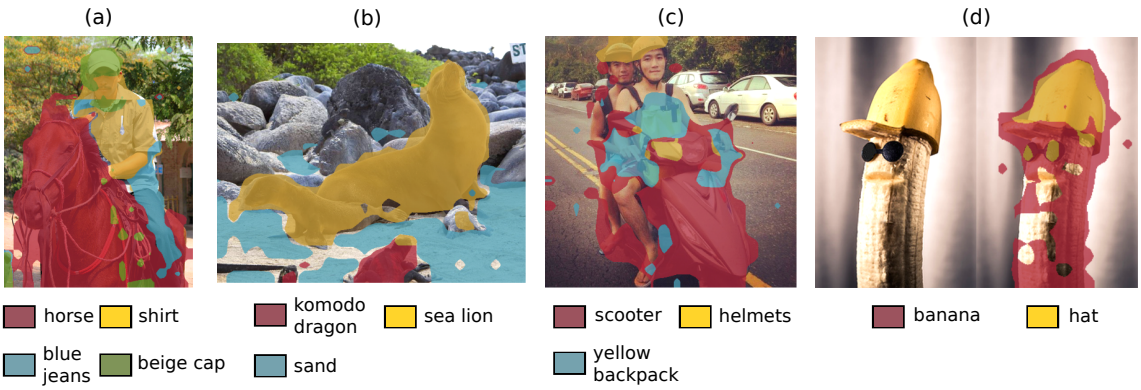


Figure 7.9: Additional qualitative results of our approach TSEG. Our approach captures rarely occurring visual concepts such as a komodo dragon or a banana-made hat.

7.7 Conclusion

This chapter introduces TSEG for weakly-supervised referring expression segmentation. We propose a multi-label patch assignment (MPA) mechanism that improves previous methods by a margin on this task. We believe our work makes an important step towards scalable image segmentation from natural language. Future work will address how to reduce the performance gap between weakly supervised and fully supervised methods and segment regions directly from image captions.

Chapter 8

Conclusion

In this last chapter, we summarize the thesis contributions before concluding with opening remarks about potential future work.

8.1 Summary of contributions

Acquiring robotic skills from demonstrations. In Chapter 3, we present an approach to acquire robotic skills from demonstrations by learning a vocabulary of simple skills that can be combined to perform moderately complex tasks such as a simple breakfast. We leverage simulation to easily design expert scripts and generate a large set of synthetic demonstrations to train policies. Then, we use our own sim-to-real method detailed in Chapter 4 to deploy policies on a real robot.

Optimizing sim-to-real data augmentations. In Chapter 4, we propose a method to search and automatically discover data augmentations for depth sensors. We discovered data augmentations by optimizing object localization as a proxy task and show that augmentations can transfer to policy learning.

Vision-based motion planning. In Chapter 5, we propose a novel approach to perform motion planning from point cloud representations of the environment. This approach alleviates the need for a predefined reconstruction of the environment typically needed in classical methods. It also allows handling dynamically changing environment and accelerates the search of solutions.

Segmentation as a sequence-to-sequence mapping. In Chapter 6, we develop Segmenter, a novel method to perform semantic segmentation. We build on Vision Transformer and propose a pure transformer model to perform segmentation by

leveraging global image context at every layer of the model and show state-of-the art results on classical benchmarks.

Weakly-supervised segmentation from referring expressions. Finally, in Chapter 7, we propose TSEG, a method that learns to localize objects without pixel-level supervision. We extend previous work that has been focusing on learning from a fixed set of labels to visual entities described by natural language. The proposed approach allows to segment a large family of visual entity and is a step towards scaling segmentation methods.

8.2 Perspectives

In this section, we present an overview of perspectives following the research projects from this thesis.

Composing skills with natural language. In Chapter 3, we proposed to learn a vocabulary of simple skills where each skill is specified by an integer value and learned with behavioral cloning. Our experience combined with recent progress in robotics suggests that simple behavioral cloning is a robust approach that scales well and can be pushed further. A more general approach to define skills is to use natural language description, similar to the description of a class to segment proposed in Chapter 7. We can learn a large set of skills specified by natural language that can be then combined to perform more complex tasks by using a sequence of instructions. An interesting problem to explore is compositionality, if we acquire skills such as *push a cube* and *pick a cylinder* with a single policy, can this policy perform tasks such as *push a cylinder* or *pick a cube*? This perspective is motivated by the success of large language models [Brown, 2020] that exhibits emerging properties on many language tasks it was not trained on such as translation or summarization when trained in a purely generative setting on a large corpus of unstructured data. Scaling simple imitation learning approaches on a large corpus of tasks could lead to the acquisition of general and reusable manipulation policies performing a wide range of tasks.

Learning multi-modal distributions. Behavioral cloning are typically trained to fit expert trajectories by minimizing a mean-squared error between predicted actions and ground truth actions that are assumed to be continuous. Policy learning is formulated as a regression problem and while this simplicity is appealing, a major drawback is that it can only fit unimodal distributions. This is problematic for tasks that are inherently multi-modal, where decision-making is involved. Given a task such as stacking two cubes, current approaches use a color cue or a text cue, such as

stack the red cube on top of green cube, to circumvent the need of choosing among multiple possible solutions to the problem. However, we aspire to acquire general stacking skills with a policy that learns to stack two cubes by first choosing which cube to stack. To do so we need to learn a distribution rather than regressing a single action. Behavioral cloning can be framed as a trajectory generation problem instead of an action fitting problem. Continuous space generative models for policy learning is a promising direction that have been recently explored with implicit models [Florence, 2021] or diffusion models [Janner, 2022]. Generative models for trajectories conditioned with natural language descriptions of tasks is an interesting direction to acquire general manipulation policies.

Contact-rich tasks. Many manipulation tasks considered in policy learning, including the ones in this thesis, are performed with a parallel gripper. The tasks either consist in a sequence of pick-and-place actions or non-prehensile interactions on tasks such as pushing an object. Pushing an object is already harder than pick-and-place as it requires closed-loop feedback to adapt the control to unpredictable displacements of the pushed object. Harder manipulation problems arise once manipulation is performed with a robotic hand composed of three or more fingers. For tasks such as manipulating a cube, it is extremely challenging, if impossible, to manually design controllers needed to generate demonstrations. An interesting direction is to leverage recent progress in trajectory optimization with optimal control based on differentiable dynamics [Lidec, 2022; Montaut, 2022]. For object manipulation, trajectory optimization applied to a robotic hand can generate a corpus of demonstrations that can be then synthesized into a single policy with imitation learning. Complex manipulation tasks such as flipping a cube upside down with robotic fingers are inherently multi-modal as they involve sequences of low-level decision-making such as fingertips points of contacts. Coupling trajectory optimization with generative modelling to learn general and flexible controllers is a thrilling research direction.

Sim-to-real with high-quality renderer and accurate physics simulation. Recent work on facial expressions [Wood, 2021] achieved impressive results on real data by using synthetic data generated high-quality rendering, showing the effectiveness of better rendering to reduce the sim-to-real gap. We observed that switching from PyBullet [Courmans, 2016] to MuJoCo [Todorov, 2012a], providing high-quality rendering and suited for domain randomization, led to improved sim-to-real performance. A promising direction to improve policies performance on a real robot is to generate images with a high quality renderer using ray-tracing such as Blender [Community, 2018], and rely on physical simulators with an accurate modelling of

contact physics inspired by progresses from simulators such as Isaac [Makoviychuk, 2021] or robotic libraries such as Pinocchio [Carpentier, 2019].

Neural motion planning. In Chapter 5, we proposed an approach to perform visually-guided motion planning with reinforcement learning. A natural extension of this work is to consider poly-articulate robots such as a UR5 or a humanoid robot as Talos. Our method learns a robot dependent policy that implicitly learns the robot geometry by colliding with obstacles while we would like to develop more general policies taking the robot geometry, composed of its bodies and links, as input to the model. An interesting problem we encountered in this project is learning to solve the slot environment, where an S-shaped robot has to pass through a thin slot. If the slot is too thin then a policy trained with RL cannot solve the problem, indeed when initialized policies generate random walks that are very unlikely to solve the problem once. To circumvent this issue, we trained on a collection of problems by randomly sampling the size of the slot leading to simpler problems to solve with a large slot. We observed that the policy was able to solve thin slots problem near the end of training, showing the importance of solving simpler problem first to generate interesting candidates for harder ones. Interestingly, most successful approaches in RL such as AlphaZero [Silver, 2017] rely on self-play to reach unprecedented performance. Self-play has a key role as the ELO, e.g. level, of the opponent player, starts at low values and increase progressively during training as both players use the policy that is training to play. Training a motion planning policy directly on hard problems is similar to trying to train an RL policy from scratch against great players such as Kasparov, the learning problem becomes much harder. Formulating motion planning as a two-player game, one player moving the robot and the other designing obstacles could be an interesting research direction.

Extending Segmenter to general segmentation tasks. While we specifically tackled semantic segmentation in Chapter 6, several extensions to more general segmentation tasks can be considered as instance segmentation or panoptic segmentation. Our approach can readily be adapted to panoptic segmentation following DETR [Carion, 2020] approach that relies on a Hungarian matching algorithm to find matches between candidate masks and ground truth masks. A limiting factor is then the need for dense annotations, independently of the segmentation task.

Localizing objects from image captions. In Chapter 7, we presented TSEG, a method that learns to localize objects from image-level supervision, using the presence or absence of a visual entity as supervision. The visual entity is specified by a natural language description, a referring expression, which allows performing

segmentation on an open vocabulary. However, the need for referring expressions as supervision is still a limitation of the approach. To scale to large datasets, we need to extract and localize visual concepts in an image directly from an image caption, obtaining similar results as MDETR [Kamath, 2021] but without localization supervision. This problem is particularly challenging because both image and text segmentation are required, as in the sentence *The handyman is eating a hot dog*, the expression *hot dog* represents a single visual entity that cannot be split into two image entities *hot* and *dog*.

Open-vocabulary object localization. TSEG, presented in Chapter 7, proposes to perform open-vocabulary object localization in a weakly-supervised fashion. TSEG allows localizing objects specified by a feature vector produced from a text description. There is a large variety of conditioning vector that can be defined to specify the object to detect, in particular OWL-ViT [Minderer, 2022] explores text and image conditioning object detection. This line of work is important as it aims to provide general purpose vision models able to localize novel objects in the wild. This is especially interesting for robotics where we would like to develop general vision representations useful to perform manipulation. Conditioning variables, specifying an object or a task with a goal image or with a text description, are a central concept to develop general vision models for robotics that are flexible and easily reusable to perform new tasks in unstructured environments.

Bibliography

[Abbeel, 2010] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. "Autonomous Helicopter Aerobatics through Apprenticeship Learning". *Int. J. Robotics Res.* 29.13 (2010), pp. 1608–1639 (cit. on p. 14).

[Abbeel, 2004] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". *ICML*. Vol. 69. ACM International Conference Proceeding Series. ACM, 2004 (cit. on p. 14).

[Ahn, 2019] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. "Weakly Supervised Learning of Instance Segmentation With Inter-Pixel Relations". *CVPR*. 2019 (cit. on pp. 20, 94, 101).

[Ahn, 2018] Jiwoon Ahn and Suha Kwak. "Learning Pixel-Level Semantic Affinity With Image-Level Supervision for Weakly Supervised Semantic Segmentation". *CVPR*. 2018 (cit. on pp. 20, 92–94, 100, 101, 103–105).

[Akgün, 2012] Baris Akgün et al. "Keyframe-based Learning from Demonstration - Method and Evaluation". *Int. J. Soc. Robotics* 4.4 (2012), pp. 343–355 (cit. on p. 14).

[Amirul Islam, 2017] Md Amirul Islam et al. "Gated Feedback Refinement Network for Dense Image Labeling". *CVPR*. 2017 (cit. on pp. 18, 69).

[Andrychowicz, 2017a] Marcin Andrychowicz et al. "Hindsight Experience Replay". *NIPS*. 2017 (cit. on p. 33).

[Andrychowicz, 2017b] Marcin Andrychowicz et al. "Hindsight experience replay". *Advances in Neural Information Processing Systems*. 2017. arXiv: 1707.01495 (cit. on pp. 57, 59).

[Araslanov, 2020] Nikita Araslanov and Stefan Roth. "Single-Stage Semantic Segmentation From Image Labels". *CVPR*. 2020 (cit. on pp. 92–94, 98–101, 103–105).

[Argall, 2009] Brenna D. Argall et al. "A survey of robot learning from demonstration". *Robotics Auton. Syst.* 57.5 (2009), pp. 469–483 (cit. on p. 14).

[Arnab, 2021a] Anurag Arnab et al. "ViViT: A Video Vision Transformer". *ICCV*. 2021 (cit. on p. 70).

[Arnab, 2021b] Anurag Arnab et al. "ViViT: A Video Vision Transformer". *ICCV* (2021) (cit. on p. 95).

[Ba, 2016] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". *arXiv preprint* (2016) (cit. on p. 76).

[Bacon, 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup. "The Option-Critic Architecture." *AAAI*. 2017 (cit. on p. 25).

[Badrinarayanan, 2017] V. Badrinarayanan, A. Kendall, and R. Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". *TPAMI* (2017) (cit. on pp. 18, 69).

[Bai, 2020] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. "Multiscale Deep Equilibrium Models". *NeurIPS*. 2020 (cit. on p. 82).

[Bertasius, 2021] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. "Is Space-Time Attention All You Need for Video Understanding?" *arXiv preprint* (2021) (cit. on p. 70).

[Bilen, 2014] Hakan Bilen, Marco Pedersoli, and Tinne Tuytelaars. "Weakly supervised object detection with posterior regularization". *BMVC*. 2014 (cit. on p. 92).

[Billard, 2004] Aude Billard et al. "Discovering optimal imitation strategies". *Robotics Auton. Syst.* 47.2-3 (2004), pp. 69–77 (cit. on p. 14).

[Bojanowski, 2014] Piotr Bojanowski et al. “Weakly supervised action labeling in videos under ordering constraints”. *ECCV*. 2014 (cit. on p. 92).

[Bojarski, 2016] Mariusz Bojarski et al. “End to End Learning for Self-Driving Cars”. *CoRR* abs/1604.07316 (2016) (cit. on p. 14).

[Bousmalis, 2018] Konstantinos Bousmalis et al. “Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping”. *ICRA*. IEEE, 2018, pp. 4243–4250 (cit. on pp. 15, 17, 41).

[Branicky, 2008] Michael S. Branicky, Ross A. Knepper, and James J. Kuffner. “Path and trajectory diversity: Theory and algorithms”. *Proceedings - IEEE International Conference on Robotics and Automation*. 2008 (cit. on p. 53).

[Brown, 2020] Tom B. Brown et al. “Language Models are Few-Shot Learners”. *NeurIPS*. 2020 (cit. on p. 113).

[Calinon, 2009] Sylvain Calinon. *Robot Programming by Demonstration - a Probabilistic Approach*. EPFL Press, 2009 (cit. on p. 14).

[Calinon, 2010] Sylvain Calinon et al. “Learning and Reproduction of Gestures by Imitation”. *IEEE Robotics Autom. Mag.* 17.2 (2010), pp. 44–54 (cit. on p. 14).

[Çalli, 2015] Berk Çalli et al. “The YCB object and Model set: Towards common benchmarks for manipulation research”. *ICAR*. IEEE, 2015, pp. 510–517 (cit. on pp. 58, 63).

[Carion, 2020] Nicolas Carion et al. “End-to-End Object Detection with Transformers”. *ECCV*. 2020 (cit. on pp. 70, 71, 73, 115).

[Carpentier, 2015–2019] Justin Carpentier, Florian Valenza, Nicolas Mansard, et al. *Pinocchio: fast forward and inverse dynamics for poly-articulated systems*. https://stack-of-tasks.github.io/pinocchio. 2015–2019 (cit. on p. 60).

[Carpentier, 2019] Justin Carpentier et al. “The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives”. *IEEE International Symposium on System Integrations (SII)*. 2019 (cit. on pp. 60, 115).

[Chang, 2015a] Angel X. Chang et al. “ShapeNet: An Information-Rich 3D Model Repository”. *arXiv* (2015). arXiv: 1512.03012 (cit. on p. 30).

[Chang, 2015b] Angel X. Chang et al. “ShapeNet: An Information-Rich 3D Model Repository”. *CoRR* abs/1512.03012 (2015) (cit. on p. 46).

[Chen, 2018a] Kan Chen, Jiyang Gao, and Ram Nevatia. “Knowledge Aided Consistency for Weakly Supervised Phrase Grounding”. *CVPR*. 2018 (cit. on p. 95).

[Chen, 2017] Liang-Chieh Chen et al. “Rethinking Atrous Convolution for Semantic Image Segmentation”. *arXiv preprint* (2017) (cit. on pp. 18, 68, 69).

[Chen, 2018b] Liang-Chieh Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. *TPAMI* (2018) (cit. on pp. 18, 68, 69).

[Chen, 2018c] Liang-Chieh Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. *IEEE Trans. Pattern Anal. Mach. Intell.* 40.4 (2018), pp. 834–848 (cit. on p. 106).

[Chen, 2018d] Liang-Chieh Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. *ECCV*. 2018 (cit. on pp. 18, 67–69, 75, 79, 81, 82).

[Chen, 2018e] Liang-Chieh Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. *ECCV*. 2018 (cit. on pp. 92, 102).

[Chen, 2020a] Mark Chen et al. “Generative Pretraining from Pixels”. *PLMR* (2020) (cit. on p. 68).

[Chen, 2020b] Ting Chen et al. “A Simple Framework for Contrastive Learning of Visual Representations”. *ICML*. 2020 (cit. on p. 92).

[Cheng, 2021] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. “Per-Pixel Classification is Not All You Need for Semantic Segmentation”. *NIPS*. 2021 (cit. on pp. 92, 95).

[Cheng, 2020] Bowen Cheng et al. “Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation”. *CVPR*. 2020 (cit. on p. 82).

[Cheng, 2018] Ching-An Cheng et al. “Fast Policy Learning through Imitation and Reinforcement”. *UAI* (2018) (cit. on pp. 25, 26).

[Choi, 2012] Changhyun Choi et al. “Voting-based pose estimation for robotic assembly using a 3D sensor”. *ICRA*. IEEE, 2012, pp. 1724–1731 (cit. on p. 13).

[Clevert, 2016] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016 (cit. on p. 56).

[Community, 2018] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018 (cit. on p. 114).

[Contributors, 2020] MMSegmentation Contributors. *MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark*. https://github.com/open-mmlab/mmsegmentation. 2020 (cit. on pp. 75, 79, 81).

[Cordts, 2016] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". *CVPR*. 2016 (cit. on pp. 69, 73).

[Coulom, 2006] Rémi Coulom. "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search". *Computers and Games*. Vol. 4630. Lecture Notes in Computer Science. Springer, 2006, pp. 72–83 (cit. on p. 45).

[Courmans, 2016] Erwin Courmans and Yunfei Bai. *PyBullet, Python module for physics simulation, robotics and machine learning*. http://pybullet.org/. 2016 (cit. on pp. 29, 45, 114).

[Cubuk, 2019] Ekin D. Cubuk et al. "AutoAugment: Learning Augmentation Strategies From Data". *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 113–123 (cit. on pp. 17, 42).

[Cubuk, 2020] Ekin D. Cubuk et al. "Randaugment: Practical automated data augmentation with a reduced search space". *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. IEEE, 2020, pp. 3008–3017 (cit. on p. 74).

[Das, 2018] Abhishek Das et al. "Neural Modular Control for Embodied Question Answering". *CoRL*. 2018 (cit. on pp. 22, 25, 26).

[Dayan, 1993] Peter Dayan and Geoffrey Hinton. "Feudal Reinforcement Learning". *NIPS*. 1993 (cit. on p. 25).

[Deng, 2009] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". *CVPR*. 2009 (cit. on p. 70).

[Deng, 2020] Xinke Deng et al. "Self-supervised 6D Object Pose Estimation for Robot Manipulation". *ICRA*. IEEE, 2020, pp. 3665–3671 (cit. on p. 13).

[Devlin, 2019] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *NAACL-HLT*. 2019 (cit. on pp. 76, 95, 102).

[Devries, 2017] Terrance Devries and Graham W. Taylor. "Improved Regularization of Convolutional Neural Networks with Cutout". *CoRR* abs/1708.04552 (2017) (cit. on pp. 44, 46).

[Ding, 2021] Henghui Ding et al. "Vision-Language Transformer and Query Generation for Referring Segmentation". *ICCV*. 2021 (cit. on pp. 95, 107).

[Doersch, 2015] Carl Doersch, Abhinav Gupta, and Alexei A Efros. "Unsupervised visual representation learning by context prediction". *ICCV*. 2015 (cit. on p. 92).

[Doersch, 2020] Carl Doersch, Ankush Gupta, and Andrew Zisserman. "CrossTransformers: spatially-aware few-shot transfer". *NeurIPS*. 2020 (cit. on p. 70).

[Dosovitskiy, 2021a] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". *ICLR*. 2021 (cit. on pp. 8, 18, 19, 68–70, 74–76, 95, 102).

[Dosovitskiy, 2021b] Alexey Dosovitskiy et al. *Vision Transformer Models*. https://console.cloud.google.com/storage/browser/vit_models. 2021 (cit. on p. 74).

[Duan, 2017] Y. Duan et al. "One-Shot Imitation Learning". *NIPS* (2017) (cit. on pp. 24, 41).

[Dvornik, 2018] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. "Modeling Visual Context Is Key to Augmenting Object Detection Datasets". *ECCV (12)*. Vol. 11216. Lecture Notes in Computer Science. Springer, 2018, pp. 375–391 (cit. on p. 42).

[Eitel, 2015] Andreas Eitel et al. "Multimodal deep learning for robust RGB-D object recognition". *IROS*. IEEE, 2015, pp. 681–687 (cit. on p. 42).

[Everingham, 2010] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". *IJCV* 88.2 (2010), pp. 303–338 (cit. on pp. 20, 93, 101).

[Fan, 2018] Ruochen Fan et al. "Associating Inter-image Salient Instances for Weakly Supervised Semantic Segmentation". *ECCV*. 2018 (cit. on pp. 20, 92–94).

[Fan, 2019] Ruochen Fan et al. "S4Net: Single Stage Salient-Instance Segmentation". *CVPR*. 2019 (cit. on pp. 20, 94).

[Farabet, 2013] C. Farabet et al. "Learning Hierarchical Features for Scene Labeling". *TPAMI* (2013) (cit. on pp. 18, 69).
[Florence, 2021] Pete Florence et al. "Implicit Behavioral Cloning". *CoRL*. Vol. 164. Proceedings of Machine Learning Research. PMLR, 2021, pp. 158–168 (cit. on pp. 14, 16, 114).
[Florensa, 2017] Carlos Florensa, Yan Duan, and Pieter Abbeel. "Stochastic Neural Networks for Hierarchical Reinforcement Learning". *ICLR* (2017) (cit. on p. 25).
[Fox, 1997] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance". *IEEE Robotics Autom. Mag.* 4.1 (1997), pp. 23–33 (cit. on p. 51).
[Frans, 2018] Kevin Frans et al. "Meta Learning Shared Hierarchies". *ICLR* (2018). arXiv: 1710.09767 (cit. on p. 25).
[Frome, 2013] Andrea Frome et al. "DeViSE: A Deep Visual-Semantic Embedding Model". *NIPS*. 2013, pp. 2121–2129 (cit. on p. 20).
[Fu, 2020] J. Fu et al. "Scene Segmentation With Dual Relation-Aware Attention Network". *TNNLS* (2020) (cit. on pp. 67–69, 81, 82).
[Fu, 2019a] Jun Fu et al. "Adaptive Context Network for Scene Parsing". *ICCV* (2019) (cit. on pp. 81, 82).
[Fu, 2019b] Jun Fu et al. "Dual Attention Network for Scene Segmentation". *CVPR*. 2019 (cit. on pp. 67–69, 82).
[Fulkerson, 2009] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. "Class segmentation and object localization with superpixel neighborhoods". *ICCV*. IEEE Computer Society, 2009, pp. 670–677 (cit. on p. 18).
[Gao, 2018] Yang Gao et al. "Reinforcement Learning from Imperfect Demonstrations". *ICLR workshop* (2018) (cit. on p. 25).
[Ghadiyaram, 2019] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. "Large-scale weakly-supervised pre-training for video action recognition". *CVPR*. 2019 (cit. on p. 92).
[Ghiasi, 2021] Golnaz Ghiasi et al. "Open-Vocabulary Image Segmentation". *CoRR* (2021) (cit. on p. 95).
[Giusti, 2016] Alessandro Giusti et al. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots". *IEEE Robotics Autom. Lett.* 1.2 (2016), pp. 661–667 (cit. on p. 14).
[Glasius, 1995] Roy Glasius, Andrzej Komoda, and Stan C.A.M. Gielen. "Neural Network Dynamics for Path Planning and Obstacle Avoidance". *Neural Networks* (1995) (cit. on pp. 52, 54).
[Gu, 2016] Shixiang Gu et al. "Deep Reinforcement Learning for Robotic Manipulation". *ICML*. 2016 (cit. on pp. 24, 25, 41).
[Gupta, 2020] Tanmay Gupta et al. "Contrastive Learning for Weakly Supervised Phrase Grounding". *ECCV*. 2020 (cit. on p. 95).
[Haarnoja, 2018a] Tuomas Haarnoja et al. "Latent Space Policies for Hierarchical Reinforcement Learning". *ICML* (2018). eprint: 1804.02808 (cit. on p. 25).
[Haarnoja, 2018b] Tuomas Haarnoja et al. "Soft Actor-Critic Algorithms and Applications". *CoRR* abs/1812.05905 (2018). arXiv: 1812.05905 (cit. on pp. 52, 58).
[Handa, 2015] Ankur Handa et al. "SceneNet: Understanding Real World Indoor Scenes With Synthetic Data". *CoRR* abs/1511.07041 (2015) (cit. on p. 42).
[Hanin, 2018] Boris Hanin and David Rolnick. "How to Start Training: The Effect of Initialization and Architecture". *NeurIPS*. 2018 (cit. on p. 75).
[Harada, 2014] Kensuke Harada, Eiichi Yoshida, and Kazuhito Yokoi. *Motion Planning for Humanoid Robots*. Springer Publishing Company, Incorporated, 2014 (cit. on p. 52).
[Hariharan, 2011] Bharath Hariharan et al. "Semantic contours from inverse detectors". *ICCV*. 2011 (cit. on p. 102).
[Hausman, 2018] Karol Hausman et al. "Learning an Embedding Space for Transferable Robot Skills". *ICLR*. 2018 (cit. on p. 25).
[He, 2019] Junjun He et al. "Adaptive Pyramid Context Network for Semantic Segmentation". *CVPR*. 2019 (cit. on p. 82).
[He, 2016a] Kaiming He et al. "Deep Residual Learning for Image Recognition". *CVPR* (2016) (cit. on pp. 31, 47).

[He, 2016b] Kaiming He et al. "Deep Residual Learning for Image Recognition". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 52, 55).
[He, 2017] Kaiming He et al. "Mask R-CNN". *ICCV*. 2017 (cit. on p. 95).
[He, 2020] Yisheng He et al. "PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation". *CVPR*. Computer Vision Foundation / IEEE, 2020, pp. 11629–11638 (cit. on p. 13).
[Hesamian, 2019] Mohammad Hesam Hesamian et al. "Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges". *JDI* (2019) (cit. on pp. 5, 66).
[Hester, 2018] Todd Hester et al. "Deep Q-Learning from Demonstrations". *AAAI* (2018) (cit. on p. 25).
[Ho, 2016] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". *NIPS*. 2016 (cit. on p. 25).
[Hong, 2018] Zhang-Wei Hong et al. "Virtual-to-real: Learning to control in visual semantic segmentation". *IJCAI* (2018) (cit. on pp. 5, 66).
[Hsieh, 2019] Chia-Wei Hsieh et al. "FashionOn: Semantic-guided image-based virtual try-on with detailed human and clothing information". *ACM MM*. 2019 (cit. on p. 67).
[Hu, 2016] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. "Segmentation from Natural Language Expressions". *ECCV*. 2016 (cit. on pp. 93, 94).
[Hu, 2020] Zhiwei Hu et al. "Bi-Directional Relationship Inferring Network for Referring Image Segmentation". *CVPR*. 2020 (cit. on p. 95).
[Huang, 2016] Gao Huang et al. "Deep Networks with Stochastic Depth". *ECCV*. 2016 (cit. on pp. 74, 76, 102).
[Huang, 2018] Zilong Huang et al. "Weakly-Supervised Semantic Segmentation Network With Deep Seeded Region Growing". *CVPR*. 2018 (cit. on p. 94).
[Huang, 2019] Zilong Huang et al. "CCNet: Criss-Cross Attention for Semantic Segmentation". *ICCV*. 2019 (cit. on pp. 67, 82).
[Ichter, 2020] Brian Ichter. *Learning sampling distributions*. https://github.com/StanfordASL/LearnedSamplingDistr 2020 (cit. on pp. 58, 59).
[Ichter, 2018] Brian Ichter, James Harrison, and Marco Pavone. "Learning Sampling Distributions for Robot Motion Planning". *Proceedings - IEEE International Conference on Robotics and Automation*. 2018. arXiv: 1709.05448 (cit. on pp. 52, 54, 55, 57–60, 62).
[Ichter, 2019] Brian Ichter and Marco Pavone. "Robot Motion Planning in Learned Latent Spaces". *IEEE Robotics and Automation Letters* (2019). arXiv: 1807.10366 (cit. on p. 64).
[Ioffe, 2015a] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". *ICML*. 2015 (cit. on p. 28).
[Ioffe, 2015b] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". *ICML*. 2015 (cit. on p. 76).
[James, 2017] Stephen James, Andrew J. Davison, and Edward Johns. "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task". *CoRL*. Vol. 78. Proceedings of Machine Learning Research. PMLR, 2017, pp. 334–343 (cit. on p. 17).
[James, 2019] Stephen James et al. "Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks". *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 12627–12637 (cit. on p. 41).
[Janner, 2022] Michael Janner et al. "Planning with Diffusion for Flexible Behavior Synthesis". *ICML*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 9902–9915 (cit. on p. 114).
[Jetchev, 2010] Nikolay Jetchev and Marc Toussaint. "Trajectory prediction in cluttered voxel environments". *Proceedings - IEEE International Conference on Robotics and Automation*. 2010 (cit. on p. 53).
[Jia, 2021] Chao Jia et al. "Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision". *ICML*. 2021 (cit. on p. 95).
[Jr, 2000] James J. Kuffner Jr. and Steven M. LaValle. "RRT-Connect: An Efficient Approach to Single-Query Path Planning". *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*. IEEE, 2000, pp. 995–1001 (cit. on pp. 52, 53, 56, 62, 64).

[Jurgenson, 2020] Tom Jurgenson. *Implementation of DDPG-MP.* https://github.com/tomjur/ModelBasedDDPG. 2020 (cit. on p. 59).

[Jurgenson, 2019] Tom Jurgenson and Aviv Tamar. "Harnessing Reinforcement Learning for Neural Motion Planning". *RSS* abs/1906.00214 (2019). arXiv: 1906.00214 (cit. on pp. 52, 54, 55, 59, 62).

[Kalashnikov, 2018] Dmitry Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". *CoRR* abs/1806.10293 (2018) (cit. on p. 15).

[Kamath, 2021] Aishwarya Kamath et al. "MDETR - Modulated Detection for End-to-End Multi-Modal Understanding". *ICCV* (2021) (cit. on pp. 19, 20, 95, 107, 116).

[Kantorov, 2016] Vadim Kantorov et al. "Contextlocnet: Context-aware deep network models for weakly supervised localization". *ECCV*. 2016 (cit. on p. 92).

[Kavraki, 1996] Lydia E. Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". *IEEE Trans. Robotics Autom.* 12.4 (1996), pp. 566–580 (cit. on pp. 52, 53).

[Kim, 2021] Beomyoung Kim, Sangeun Han, and Junmo Kim. "Discriminative Region Suppression for Weakly-Supervised Semantic Segmentation". *AAAI*. 2021 (cit. on p. 103).

[Kingma, 2015a] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". *ICLR* (2015) (cit. on p. 28).

[Kingma, 2015b] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". *ICLR*. 2015 (cit. on pp. 59, 103).

[Kober, 2013] J. Kober, J. A. Bagnell, and J. Peters. "Reinforcement learning in robotics: A survey". *IJRR* 32.11 (2013) (cit. on p. 25).

[Kolesnikov, 2016] Alexander Kolesnikov and Christoph H. Lampert. "Seed, Expand and Constrain: Three Principles for Weakly-Supervised Image Segmentation". *ECCV*. 2016 (cit. on pp. 20, 94, 101).

[Kostrikov, 2018] Ilya Kostrikov. *PyTorch Implementations of Reinforcement Learning Algorithms.* https://github.com/ikostrikov/pytorch-a2c-ppo-acktr. 2018 (cit. on p. 28).

[Krishna, 2017] Ranjay Krishna et al. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations". *IJCV* 123.1 (2017), pp. 32–73 (cit. on p. 102).

[Krizhevsky, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems (NIPS)*. 2012, pp. 1097–1105 (cit. on pp. 18, 52, 55).

[Kulkarni, 2016] Tejas D Kulkarni et al. "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation". *NIPS*. 2016 (cit. on p. 25).

[Kumar, 2016] Vikash Kumar et al. "Learning Dexterous Manipulation Policies from Experience and Imitation". *arXiv* (2016) (cit. on p. 25).

[Labbé, 2020] Yann Labbé et al. "CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation". *ECCV (17)*. Vol. 12362. Lecture Notes in Computer Science. Springer, 2020, pp. 574–591 (cit. on p. 13).

[Lampe, 2013] Thomas Lampe and Martin Riedmiller. "Acquiring Visual Servoing Reaching and Grasping Skills using Neural Reinforcement Learning". *IJCNN* (2013) (cit. on p. 24).

[Laskey, 2017] Michael Laskey et al. "DART: Noise Injection for Robust Imitation Learning". *CoRL*. 2017 (cit. on p. 25).

[Latombe, 1991] Jean-Claude Latombe. *Robot Motion Planning*. USA: Kluwer Academic Publishers, 1991 (cit. on pp. 51, 53).

[Latombe,] Lydia E Kavraki Jean-Claude Latombe. "Probabilistic roadmaps for robot path planning" () (cit. on pp. 53, 58, 64, 65).

[Laumond, 2006] J.-P. Laumond. "Kineo CAM: a success story of motion planning algorithms". *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 90–93 (cit. on p. 52).

[LaValle, 2006] Steven M. LaValle. *Planning Algorithms*. USA: Cambridge University Press, 2006 (cit. on pp. 51, 53).

[Le, 2018] Hoang M. Le et al. "Hierarchical Imitation and Reinforcement Learning". *ICML*. 2018 (cit. on pp. 25, 26).

[Lee, 2021a] Alex X. Lee et al. "Beyond Pick-and-Place: Tackling Robotic Stacking of Diverse Shapes". *CoRL*. Vol. 164. Proceedings of Machine Learning Research. PMLR, 2021, pp. 1089–1131 (cit. on pp. 15, 16).

[Lee, 2019a] Jungbeom Lee et al. "Frame-to-Frame Aggregation of Active Regions in Web Videos for Weakly Supervised Semantic Segmentation". *ICCV*. 2019 (cit. on p. 94).
[Lee, 2019b] Kuan-Hui Lee et al. "SPIGAN: Privileged Adversarial Learning from Simulation". *ICLR (Poster)*. OpenReview.net, 2019 (cit. on p. 41).
[Lee, 2021b] Seungho Lee et al. "Railroad Is Not a Train: Saliency As Pseudo-Pixel Supervision for Weakly Supervised Semantic Segmentation". *CVPR*. 2021 (cit. on p. 103).
[Lee, 2019c] Youngwoon Lee et al. "Composing Complex Skills by Learning Transition Policies with Proximity Reward Induction". *ICLR*. 2019 (cit. on p. 25).
[Levine, 2015] Sergey Levine et al. "End-to-End Training of Deep Visuomotor Policies". *JMLR* (2015) (cit. on p. 24).
[Levine, 2016] Sergey Levine et al. "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection". *ISER* (2016) (cit. on pp. 39, 41).
[Levine, 2018] Sergey Levine et al. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection". *Int. J. Robotics Res.* 37.4-5 (2018), pp. 421–436 (cit. on pp. 15, 16).
[Li, 2016] Dong Li et al. "Weakly supervised object localization with progressive domain adaptation". *CVPR*. 2016 (cit. on p. 92).
[Li, 2019] Liunian Harold Li et al. "VisualBERT: A Simple and Performant Baseline for Vision and Language". *arXiv preprint arXiv:1908.03557* (2019) (cit. on p. 95).
[Li, 2018] Yi Li et al. "DeepIM: Deep Iterative Matching for 6D Pose Estimation". *ECCV (6)*. Vol. 11210. Lecture Notes in Computer Science. Springer, 2018, pp. 695–711 (cit. on p. 13).
[Li, 2020] Yonggang Li et al. "DADA: Differentiable Automatic Data Augmentation". *CoRR* abs/2003.03780 (2020) (cit. on p. 17).
[Liang-Chieh, 2015] Chen Liang-Chieh et al. "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs". *ICLR*. 2015 (cit. on pp. 5, 18, 69, 75).
[Lidec, 2022] Quentin Le Lidec et al. "Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems". *CoRR* abs/2203.03986 (2022) (cit. on p. 114).
[Lien, 2010] Jyh Ming Lien and Yanyan Lu. "Planning motion in environments with similar obstacles". *Robotics: Science and Systems*. 2010 (cit. on p. 53).
[Lillicrap, 2016] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". *ICLR* (2016) (cit. on p. 33).
[Lin, 2017] Guosheng Lin et al. "RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation". *CVPR*. 2017 (cit. on pp. 18, 69).
[Lin, 2020] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". *IEEE Trans. Pattern Anal. Mach. Intell.* 42.2 (2020), pp. 318–327 (cit. on p. 104).
[Litvak, 2019] Yuval Litvak, Armin Biess, and Aharon Bar-Hillel. "Learning Pose Estimation for High-Precision Robotic Assembly Using Simulated Depth Images". *ICRA*. IEEE, 2019, pp. 3521–3527 (cit. on pp. 13, 41).
[Liu, 2019] Xuejing Liu et al. "Adaptive Reconstruction Network for Weakly Supervised Referring Expression Grounding". *ICCV*. 2019 (cit. on p. 95).
[Liu, 2021a] Yongfei Liu et al. "Relation-aware Instance Refinement for Weakly Supervised Visual Grounding". *CVPR*. 2021 (cit. on p. 95).
[Liu, 2021b] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". *arXiv preprint* (2021) (cit. on pp. 70, 75, 80, 81).
[Liu, 2021c] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". *ICCV*. 2021 (cit. on pp. 92, 95).
[Long, 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for semantic segmentation". *CVPR*. 2015 (cit. on pp. 18, 69).
[Loshchilov, 2019] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". *ICLR*. 2019 (cit. on p. 103).
[Lozano-Pérez, 2014] Tomás Lozano-Pérez and Leslie Pack Kaelbling. "A constraint-based method for solving sequential manipulation planning problems". *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 3684–3691 (cit. on p. 22).
[Makoviychuk, 2021] Viktor Makoviychuk et al. "Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning". *NeurIPS Datasets and Benchmarks*. 2021 (cit. on p. 115).

[Martin, 2007] Sean R. Martin, Steve E. Wright, and John W. Sheppard. "Offline and online evolutionary bi-directional RRT algorithms for efficient re-planning in dynamic environments". *Proceedings of the 3rd IEEE International Conference on Automation Science and Engineering, IEEE CASE 2007*. 2007 (cit. on p. 53).

[Miech, 2020] Antoine Miech et al. "End-to-end learning of visual representations from uncurated instructional videos". *CVPR*. 2020 (cit. on p. 92).

[Minaee, 2021] Shervin Minaee et al. "Image Segmentation Using Deep Learning: A Survey". *TPAMI* (2021) (cit. on p. 67).

[Minderer, 2022] Matthias Minderer et al. "Simple Open-Vocabulary Object Detection with Vision Transformers". *CoRR* abs/2205.06230 (2022) (cit. on pp. 20, 116).

[Mnih, 2015] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". *Nat.* 518.7540 (2015), pp. 529–533 (cit. on pp. 15, 16, 22, 25).

[Montaut, 2022] Louis Montaut et al. "Differentiable Collision Detection: a Randomized Smoothing Approach". *CoRR* abs/2209.09012 (2022) (cit. on p. 114).

[Mottaghi, 2014] Roozbeh Mottaghi et al. "The Role of Context for Object Detection and Semantic Segmentation in the Wild". *CVPR*. 2014 (cit. on pp. 69, 73).

[Müller, 2018] Matthias Müller et al. "Driving Policy Transfer via Modularity and Abstraction". *CoRL*. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1–15 (cit. on p. 41).

[N Yang, 2012] N. Yang, Y. Kim, and R. Park. "Depth hole filling using the depth distribution of neighboring regions of depth holes in the kinect sensor". *ICSPCC*. 2012 (cit. on p. 42).

[Nachum, 2018] Ofir Nachum et al. "Data-Efficient Hierarchical Reinforcement Learning". *NIPS* (2018). arXiv: 1805.08296 (cit. on p. 25).

[Nair, 2018] Ashvin Nair et al. "Overcoming Exploration in Reinforcement Learning with Demonstrations". *ICRA*. 2018 (cit. on p. 25).

[Ng, 2000] Andrew Y. Ng and Stuart J. Russell. "Algorithms for Inverse Reinforcement Learning". *ICML*. Morgan Kaufmann, 2000, pp. 663–670 (cit. on pp. 14, 22, 25).

[Pan, 2012] Jia Pan, Sachin Chitta, and Dinesh Manocha. "FCL: A general purpose library for collision and proximity queries". *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3859–3866 (cit. on p. 60).

[Papandreou, 2015] George Papandreou et al. "Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation". *ICCV*. 2015 (cit. on p. 94).

[Pashevich, 2019] Alexander Pashevich et al. "Learning to Augment Synthetic Images for Sim2Real Policy Transfer". *IROS* (2019) (cit. on pp. 10, 23, 29, 34).

[Paulin, 2014] Mattis Paulin et al. "Transformation Pursuit for Image Classification". *CVPR*. IEEE Computer Society, 2014, pp. 3646–3653 (cit. on p. 42).

[Peng, 2018] Xue Bin Peng et al. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". *ICRA*. IEEE, 2018, pp. 1–8 (cit. on p. 17).

[Perez, 2018] Ethan Perez et al. "FiLM: Visual Reasoning with a General Conditioning Layer". *AAAI*. 2018 (cit. on p. 27).

[Pfeiffer, 2017] Mark Pfeiffer et al. "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots". *Proceedings - IEEE International Conference on Robotics and Automation*. 2017. arXiv: 1609.07910 (cit. on pp. 52, 54).

[Pinheiro, 2014] Pedro Pinheiro and Ronan Collobert. "Recurrent Convolutional Neural Networks for Scene Labeling". *ICML*. 2014 (cit. on pp. 18, 69).

[Pinheiro, 2015] Pedro H. O. Pinheiro and Ronan Collobert. "From image-level to pixel-level labeling with Convolutional Networks". *CVPR*. 2015 (cit. on p. 94).

[Pinto, 2016a] Lerrel Pinto and Abhinav Gupta. "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours". *ICRA*. IEEE, 2016, pp. 3406–3413 (cit. on p. 15).

[Pinto, 2016b] Lerrel Pinto and Abhinav Gupta. "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours". *ICRA* (2016) (cit. on pp. 39, 41).

[Pinto, 2018] Lerrel Pinto et al. "Asymmetric Actor Critic for Image-Based Robot Learning". *RSS* (2018) (cit. on pp. 22, 23, 25, 33, 41).

[Plappert, 2018] Matthias Plappert et al. "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research". *arXiv* (2018). arXiv: 1802.09464 (cit. on pp. 23, 32).

[Pohlen, 2017] Tobias Pohlen et al. "Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes". *CVPR*. 2017 (cit. on pp. 18, 69).

[Pomerleau, 1988] Dean Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network". *NIPS*. Morgan Kaufmann, 1988, pp. 305–313 (cit. on pp. 14, 22–24, 26, 42, 56).

[Pong, 2020] Vitchyr Pong. *Reinforcement learning framework and algorithms implemented in PyTorch*. https://github.com/vitchyr/rlkit. 2020 (cit. on p. 59).

[Popov, 2017] Ivaylo Popov et al. "Data-efficient Deep Reinforcement Learning for Dexterous Manipulation". *arXiv* (2017) (cit. on pp. 24, 25).

[Qi, 2017] Charles R. Qi et al. "PointNet: Deep learning on point sets for 3D classification and segmentation". *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017. arXiv: 1612.00593 (cit. on pp. 51–55).

[Qureshi, 2020] Ahmed Qureshi. *Implementation of MPNet: Motion Planning Networks*. https://github.com/ahq1993/MPNet. 2020 (cit. on pp. 58, 59).

[Qureshi, 2019] Ahmed H. Qureshi et al. "Motion planning networks". *Proceedings - IEEE International Conference on Robotics and Automation*. 2019. arXiv: 1806.05767 (cit. on pp. 52, 54, 55, 57–59, 62, 64).

[Rad, 2017] Mahdi Rad and Vincent Lepetit. "BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth". *ICCV*. IEEE Computer Society, 2017, pp. 3848–3856 (cit. on p. 13).

[Radford, 2021] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". *ICML*. 2021 (cit. on pp. 6, 20, 92, 95).

[Ramesh, 2021] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". *ICML*. 2021 (cit. on p. 95).

[Ramesh, 2022] Aditya Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents". *CoRR* abs/2204.06125 (2022) (cit. on p. 20).

[Ratliff, 2009] Nathan D. Ratliff et al. "CHOMP: Gradient optimization techniques for efficient motion planning". *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*. IEEE, 2009, pp. 489–494 (cit. on p. 62).

[Ren, 2017] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". *PAMI* 39.6 (2017), pp. 1137–1149 (cit. on p. 95).

[Riedmiller, 2018a] Martin A. Riedmiller et al. "Learning by Playing - Solving Sparse Reward Tasks from Scratch". *MLR* (2018). arXiv: 1802.10567 (cit. on pp. 24, 41).

[Riedmiller, 2018b] Martin A. Riedmiller et al. "Learning by Playing Solving Sparse Reward Tasks from Scratch". *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4341–4350 (cit. on p. 15).

[Robbins, 1951] H. Robbins and S. Monro. "A Stochastic Approximation Method". *Annals of Mathematical Statistics* (1951) (cit. on pp. 75, 102).

[Ronneberger, 2015] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". *MICCAI*. 2015 (cit. on pp. 18, 69).

[Ross, 2014] Stephane Ross and J. Andrew Bagnell. "Reinforcement and Imitation Learning via Interactive No-Regret Learning". *arXiv*. 2014. arXiv: 1406.5979 (cit. on pp. 22, 25, 33).

[Ross, 2011] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". *AISTATS*. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 627–635 (cit. on pp. 42, 61).

[Rothganger, 2006] Fred Rothganger et al. "3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints". *Int. J. Comput. Vis.* 66.3 (2006), pp. 231–259 (cit. on p. 13).

[Ruiz, 2019] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. "Learning To Simulate". *ICLR (Poster)*. OpenReview.net, 2019 (cit. on p. 42).

[Sadeghi, 2018] Fereshteh Sadeghi et al. "Sim2Real Viewpoint Invariant Visual Servoing by Recurrent Control". *CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4691–4699 (cit. on p. 41).

[Schaal, 2003] Stefan Schaal et al. "Learning Movement Primitives". *ISRR*. Vol. 15. Springer Tracts in Advanced Robotics. Springer, 2003, pp. 561–572 (cit. on pp. 14, 56).

[Schulman, 2017] John Schulman et al. "Proximal Policy Optimization Algorithms". *arXiv* (2017). arXiv: 1707.06347 (cit. on p. 28).

[Schwartz, 1986] Jacob T. Schwartz, Micha Sharir, and John E. Hopcroft, eds. *Planning, Geometry, and Complexity of Robot Motion*. USA: Ablex Publishing Corp., 1986 (cit. on p. 52).

[Sennrich, 2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". *ACL*. 2016 (cit. on p. 97).
[Shelhamer, 2017] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". *TPAMI* (2017) (cit. on p. 67).
[Shotton, 2008] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. "Semantic texton forests for image categorization and segmentation". *CVPR*. IEEE Computer Society, 2008 (cit. on p. 18).
[Siam, 2017] M. Siam et al. "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges". *ITSC*. 2017 (cit. on pp. 5, 66).
[Silver, 2016] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". *Nat.* 529.7587 (2016), pp. 484–489 (cit. on pp. 15, 16, 22, 25, 40, 41).
[Silver, 2017] David Silver et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". *CoRR* abs/1712.01815 (2017) (cit. on p. 115).
[Simonyan, 2014] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *arXiv* (2014). arXiv: 1409.1556 (cit. on p. 31).
[Srivastava, 2014] Nitish Srivastava et al. "Dropout: a Simple Way to Prevent Neural Networks from Overfitting". *JMLR* (2014) (cit. on pp. 22, 74, 76).
[Steiner, 2021] Andreas Steiner et al. "How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers". *arXiv preprint arXiv:2106.10270* (2021) (cit. on pp. 74, 102).
[Stevsic, 2020] Stefan Stevsic, Sammy Christen, and Otmar Hilliges. "Learning to Assemble: Estimating 6D Poses for Robotic Object-Object Manipulation". *IEEE Robotics Autom. Lett.* 5.2 (2020), pp. 1159–1166 (cit. on p. 13).
[Strudel, 2020a] R. Strudel et al. "Learning Obstacle Representations for Neural Motion Planning". 2020 (cit. on p. 10).
[Strudel,] Robin Strudel and Ricardo Garcia. "Learning Obstacle Representations for Neural Motion Planning, project webpage" () (cit. on p. 53).
[Strudel, 2022a] Robin Strudel, Ivan Laptev, and Cordelia Schmid. "Weakly-supervised segmentation of referring expressions". *CoRR* abs/2205.04725 (2022) (cit. on pp. 10, 20).
[Strudel, 2020b] Robin Strudel et al. "Learning to combine primitive skills: A step towards versatile robotic manipulation §". *ICRA*. IEEE, 2020, pp. 4637–4643 (cit. on pp. 10, 14, 15).
[Strudel, 2021] Robin Strudel et al. "Segmenter: Transformer for Semantic Segmentation". *ICCV*. IEEE, 2021, pp. 7242–7252 (cit. on pp. 10, 19, 92, 93, 95, 101, 102).
[Strudel, 2022b] Robin Strudel et al. "Self-conditioned Embedding Diffusion for Text Generation". *arXiv preprint* (2022) (cit. on p. 11).
[Suárez-Ruiz, 2018] Francisco Suárez-Ruiz, Xian Zhou, and Quang-Cuong Pham. "Can robots assemble an IKEA chair?" *Science Robotics* 3.17 (2018), eaat6385 (cit. on p. 22).
[Sultana, 2020] Farhana Sultana, Abu Sufian, and Paramartha Dutta. "Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey". *Knowledge-Based Systems* (2020) (cit. on p. 67).
[Sun, 2018] Wen Sun, J Andrew Bagnell, and Byron Boots. "Truncated horizon policy search: Combining reinforcement learning & imitation learning". *ICLR*. 2018 (cit. on pp. 25, 26).
[Sutton, 1999] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". *Artificial intelligence* 112.1-2 (1999), pp. 181–211 (cit. on p. 25).
[Sutton, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018 (cit. on pp. 16, 56, 57).
[Tobin, 2017] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". *IROS*. IEEE, 2017, pp. 23–30 (cit. on pp. 17, 39, 41).
[Todorov, 2012a] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". *IROS*. IEEE, 2012, pp. 5026–5033 (cit. on pp. 16, 114).
[Todorov, 2012b] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". *IROS*. 2012 (cit. on p. 32).
[Toussaint, 2015] Marc Toussaint. "Logic-geometric programming: An optimization-based approach to combined task and motion planning". *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015 (cit. on p. 22).

[Touvron, 2020] Hugo Touvron et al. "Training Data-Efficient Image Transformers and Distillation Through Attention". *arXiv preprint* (2020) (cit. on pp. 68, 70, 74, 76).
[Tu, 2010] Zhuowen Tu and Xiang Bai. "Auto-Context and Its Application to High-Level Vision Tasks and 3D Brain Image Segmentation". *IEEE Trans. Pattern Anal. Mach. Intell.* 32.10 (2010), pp. 1744–1757 (cit. on p. 18).
[Turc, 2019] Iulia Turc et al. "Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation". *arXiv preprint arXiv:1908.08962* (2019) (cit. on p. 102).
[Vaswani, 2017] Ashish Vaswani et al. "Attention is All you Need". *NIPS*. 2017 (cit. on pp. 68, 70, 71, 95).
[Vezhnevets, 2017] Alexander Sasha Vezhnevets et al. "FeUdal Networks for Hierarchical Reinforcement Learning". *ICML*. 2017 (cit. on p. 25).
[Wang, 2019] Chen Wang et al. "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion". *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 3343–3352 (cit. on p. 13).
[Wang, 2020a] Huiyu Wang et al. "Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation". *ECCV*. 2020 (cit. on p. 70).
[Wang, 2020b] Huiyu Wang et al. "MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers". *arXiv preprint* (2020) (cit. on p. 70).
[Wang, 2020c] Huiyu Wang et al. "MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers". *arXiv preprint* (2020) (cit. on p. 73).
[Wang, 2017] Jingdong Wang et al. "Salient Object Detection: A Discriminative Regional Feature Integration Approach". *IJCV* 123.2 (2017), pp. 251–268 (cit. on p. 94).
[Wang, 2018] Xiaolong Wang et al. "Non-Local Neural Networks". *CVPR*. 2018 (cit. on p. 70).
[Wang, 2020d] Xinlong Wang et al. "SOLOv2: Dynamic and Fast Instance Segmentation". *NeurIPS*. 2020 (cit. on p. 73).
[Wei, 2017] Yunchao Wei et al. "Object Region Mining with Adversarial Erasing: A Simple Classification to Semantic Segmentation Approach". *CVPR*. 2017 (cit. on pp. 20, 94, 98).
[Wei, 2018] Yunchao Wei et al. "Revisiting Dilated Convolution: A Simple Approach for Weakly- and Semi-Supervised Semantic Segmentation". *CVPR*. 2018 (cit. on p. 100).
[Wightman, 2020] Ross Wightman. *PyTorch Image Models*. `https://github.com/rwightman/pytorch-image-models`. 2020 (cit. on p. 74).
[Winston, 1970] Patrick H. Winston. *Copy Demo*. 1970 (cit. on pp. 12, 13).
[Wood, 2021] Erroll Wood et al. "Fake it till you make it: face analysis in the wild using synthetic data alone". *ICCV*. IEEE, 2021, pp. 3661–3671 (cit. on p. 114).
[Wu, 2020] Chenyun Wu et al. "PhraseCut: Language-Based Image Segmentation in the Wild". *CVPR*. 2020 (cit. on pp. 93, 102).
[Wu, 2015] Zhirong Wu et al. "3D ShapeNets: A deep representation for volumetric shapes". *CVPR*. IEEE Computer Society, 2015, pp. 1912–1920 (cit. on p. 46).
[Xiang, 2018] Yu Xiang et al. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes". *Robotics: Science and Systems*. 2018 (cit. on p. 13).
[Xiao, 2017] Fanyi Xiao, Leonid Sigal, and Yong Jae Lee. "Weakly-Supervised Visual Grounding of Phrases with Linguistic Structures". *CVPR*. 2017 (cit. on p. 95).
[Xu, 2022] Jiarui Xu et al. "GroupViT: Semantic Segmentation Emerges from Text Supervision". *CoRR* (2022) (cit. on pp. 95, 108).
[Xu, 2021] Mengde Xu et al. "A Simple Baseline for Zero-shot Semantic Segmentation with Pre-trained Vision-language Model". *CoRR* (2021) (cit. on p. 95).
[Yang, 2000] Simon X. Yang and Max Meng. "An efficient neural network approach to dynamic robot motion planning". *Neural Networks* (2000) (cit. on pp. 52, 54).
[Ye, 2019] Linwei Ye et al. "Cross-Modal Self-Attention Network for Referring Image Segmentation". *CVPR*. 2019 (cit. on p. 95).
[Yin, 2020] Minghao Yin et al. "Disentangled Non-local Neural Networks". *ECCV*. 2020 (cit. on pp. 67, 69, 81, 82).
[Yu, 2020] Changqian Yu et al. "Context Prior for Scene Segmentation". *CVPR*. 2020 (cit. on pp. 67–69, 81, 82).
[Yu, 2016a] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". *ICLR*. 2016 (cit. on p. 69).

[Yu, 2016b] Licheng Yu et al. "Modeling Context in Referring Expressions". *ECCV*. 2016 (cit. on pp. 93, 102).
[Yu, 2018] Licheng Yu et al. "MAttNet: Modular Attention Network for Referring Expression Comprehension". *CVPR*. 2018 (cit. on pp. 93, 95).
[Yu, 2019] Zeng Yu et al. "Joint Learning of Saliency Detection and Weakly Supervised Semantic Segmentation". *ICCV*. 2019 (cit. on pp. 20, 94).
[Yuan, 2020] Yuhui Yuan, Xilin Chen, and Jingdong Wang. "Object-Contextual Representations for Semantic Segmentation". *ECCV*. 2020 (cit. on pp. 67, 81, 82).
[Yuan, 2018] Yuhui Yuan and Jingdong Wang. "OCNet: Object Context Network for Scene Parsing". *arXiv preprint* (2018) (cit. on pp. 68, 69).
[Zabari, 2021] Nir Zabari and Yedid Hoshen. "Semantic Segmentation In-the-Wild Without Seeing Any Segmentation Examples". *CoRR* (2021) (cit. on p. 95).
[Zhang, 2019] H. Zhang et al. "Co-Occurrent Features in Semantic Segmentation". *CVPR*. 2019 (cit. on pp. 67, 82).
[Zhang, 2018a] Hang Zhang et al. "Context Encoding for Semantic Segmentation". *CVPR*. 2018 (cit. on p. 14).
[Zhang, 2020] Hang Zhang et al. "ResNeSt: Split-Attention Networks". *arXiv preprint* (2020) (cit. on pp. 79, 80).
[Zhang, 2018b] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018 (cit. on p. 74).
[Zhang, 2018c] Tianhao Zhang et al. "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation". *ICRA*. IEEE, 2018, pp. 1–8 (cit. on pp. 14, 15, 39, 41).
[Zhao, 2017] Hengshuang Zhao et al. "Pyramid Scene Parsing Network". *CVPR*. 2017 (cit. on pp. 67, 69).
[Zhao, 2018] Hengshuang Zhao et al. "PSANet: Point-wise Spatial Attention Network for Scene Parsing". *ECCV*. 2018 (cit. on pp. 67–69, 82).
[Zhao, 2020] Shengyu Zhao et al. "Differentiable Augmentation for Data-Efficient GAN Training". *CoRR* abs/2006.10738 (2020) (cit. on p. 17).
[Zheng, 2020] Sixiao Zheng et al. "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers". *arXiv preprint* (2020) (cit. on pp. 70, 80–82).
[Zhou, 2016] Bolei Zhou et al. "Learning Deep Features for Discriminative Localization". *CVPR*. 2016 (cit. on pp. 19, 92–94, 103–105).
[Zhou, 2019] Bolei Zhou et al. "Semantic Understanding of Scenes Through the ADE20K Dataset". *IJCV* (2019) (cit. on pp. 68, 69, 73).
[Zhou, 2021] Chong Zhou, Chen Change Loy, and Bo Dai. "DenseCLIP: Extract Free Dense Labels from CLIP". *CoRR* (2021) (cit. on p. 95).
[Zhu, 2018a] Yuke Zhu et al. "Reinforcement and Imitation Learning for Diverse Visuomotor Skills". *Robotics: Science and Systems*. 2018 (cit. on pp. 15, 16).
[Zhu, 2018b] Yuke Zhu et al. "Reinforcement and Imitation Learning for Diverse Visuomotor Skills". *arXiv* (2018). arXiv: 1802.09564 (cit. on p. 25).
[Zhu, 2019] Zhen Zhu et al. "Asymmetric Non-local Neural Networks for Semantic Segmentation". *ICCV*. 2019 (cit. on p. 82).
[Ziebart, 2008] Brian D. Ziebart et al. "Maximum Entropy Inverse Reinforcement Learning". *AAAI*. AAAI Press, 2008, pp. 1433–1438 (cit. on p. 14).

RÉSUMÉ

Le but de cette thèse est de développer des modèles, représentations et méthodes d'apprentissage pour l'acquisition automatique de compétences robotiques guidées par la vision à partir de démonstrations, ainsi que la localisation d'objets. Nous présentons tout d'abord une méthode pour l'acquisition de compétences robotiques à partir de démonstrations. Un vocabulaire de compétences élémentaires est appris avec de l'apprentissage de politiques par imitation. Les compétences sont ensuite combinées avec une politique de planification apprise par renforcement afin de réaliser des taches plus complexes. Nous montrons sur plusieurs taches que la politique entrainée en simulation transfère sur un robot réel. Pour ce faire, nous avons développé une méthode qui optimise des séquences d'augmentations de données synthétiques afin de résoudre une tache auxiliaire de localisation d'objets sur des données réelles.

Nous proposons ensuite une méthode de planification de mouvements à partir de capteurs. Notre méthode exploite la connaissance des obstacles environnants pour accélérer la recherche de chemins sans collision. La représentation apprise généralise sur une grande variété d'obstacles et la politique de planification fonctionne sur de nouveaux environnements avec des obstacles se déplaçant de manière dynamique.

Alors que les politiques guidées par la vision apprennent des représentations visuelles à partir du contrôle, une autre approche consiste à apprendre des représentations visuelles centrées sur les objets à manipuler. Une fois que la localisation d'un objet est estimée, elle est ensuite intégrée à des contrôleurs robotiques classiques. Les représentations centrées sur les objets reposent sur des méthodes de segmentation que nous proposons d'améliorer avec les contributions suivantes.

Nous introduisons une méthode de segmentation sémantique basée sur les transformers qui exploite l'information globale contenue dans une image à toutes les couches du modèle. Nous obtenons des résultats état de l'art et montrons l'avantage de notre modèle comparé à des réseaux de convolution. Notre méthode de segmentation présente deux limitations, le modèle localise des objets qui sont prédéfinis et son entrainement nécessite des images annotées pour chaque pixel. Pour remédier à ces limitations, nous présentons une méthode qui segmente des objets définis à partir d'une description texte et ne nécessite pas de supervision au niveau des pixels. Notre méthode apprend à localiser des objets en utilisant des annotations au niveau de l'image uniquement comme la présence ou l'absence d'un objet dans l'image.

MOTS CLÉS

Robotique, Vision par ordinateur, Apprentissage par imitation, Segmentation d'images

ABSTRACT

The goal of this thesis is to develop models, representations and learning algorithms for the automatic acquisition of visually-guided robotic skills from demonstrations and for object localization.

We first introduce a method to acquire robotic skills from demonstrations by learning a vocabulary of basic skills with behavioral cloning. Skills are then combined with a planning policy learned with reinforcement learning in order to perform more complex tasks. We show successful transfer of multiple tasks from simulation to a real robot by using a method developed in this thesis optimizing a sequence of data augmentations on synthetic data to solve a proxy object localization task on real data.

We then focus on sensor-based motion planning and propose an approach leveraging the knowledge of surrounding obstacles observed with a camera to accelerate the finding of collision-free paths. The learned representation generalizes across a large variety of objects, and the planning policy can handle new environments with dynamically moving obstacles.

While visually-guided policies learn task-centric image representations from control supervision, another line of work consists in learning object-centric representations that can be plugged into classical robotics methods. Object-centric approaches rely on a segmentation backbone for which we propose the following contributions.

Towards this goal we propose a transformer-based semantic segmentation model that leverages global context of the image at every stage of the model and show state-of-the-art results when compared to convolution-based approaches on classical benchmarks. Our segmentation model presents two limitations, it localizes a pre-defined set objects and requires dense annotations to be trained, which limits its scalability to large datasets. To address these limitations, we propose a method that segments an open set of visual concepts defined by natural language and does not require pixel-level supervision. Our method learns to localize objects by using image-level labels such as the presence of an object in the image.

KEYWORDS

Robotics, Computer Vision, Imitation learning, Image segmentation