Context-based Affordance Segmentation from 2D Images for Robot Actions

Timo Lüddecke[a], Tomas Kulvicius[a], Florentin Wörgötter[a]

[a] *Universität Göttingen*
Department for Computational Neuroscience at the Bernstein Center Göttingen
Inst. of Physics 3 and Leibniz Science Campus for Primate Cognition.

Abstract

Affordances play a crucial role in robotics since they allow developing truly autonomous robots, which can freely explore and interact with the environment. Most of the existing approaches for analyzing affordances in a scene consider only one or few types of affordance, e.g., grasping points, object manipulation or locomotion. In many cases only whole objects are considered. In our study we include in total 12 affordances of object-related, manipulation and locomotion affordances, considering affordances of both objects and/or their parts. We design a system that can densely predict affordances given only a single 2D RGB image. For this, we propose a method that transfers object class labels to affordances. This enables us to train convolutional neural networks, a PSPNet-based network and a U-Net-style network, to directly predict affordances from an image using a selective binary cross entropy loss function. The method is able to handle (potentially multiple) affordances of objects and their parts in a pixel-wise manner even in the case of incomplete data. We perform qualitative as well as quantitative evaluations with simulated and real data including robot experiments. In general, we find that frequent affordances are recognized with a substantial fraction of correctly assigned pixels, while this is harder for infrequent affordances and small objects. In addition, we demonstrate that our method performs better than a recent competitive approach. As the proposed method operates on 2D images, it is easier to implement than competing 3D methods and it could therefore more easily provide useful affordance estimates for robotic actions as demonstrated experimentally.

1. Introduction

To express opportunities for action of an animal in its environment, the perceptual psychologist J.J.Gibson [1] coined the term *affordances*. He defines affordances as opportunities for action between an animal and the environment.Examples for affordances from the perspective of a cat are: Shrubbery affords shelter and a mouse affords nutrition. Later, the term was adopted by the robotics community and extended from animals to robots. Essentially in robotics this term very often takes the meaning of: "Which actions could a robot perform in a given situation (with some given objects)?". This perspective on affordances is adopted in our work, too. We assume a human-like embodiement of the robot, leading to a set of affordances similar to those that humans would encounter. Task-specific refinements of the affordances can be carried out depending on actual embodiement and application. But even for other systems the capability to understand affordances can be useful, in particular if interaction with humans is required such as in a smart home. In this work, we address the problem of segmenting affordances for (but not limited to) robotic applications.

Segmentation means that the output of the system we propose in this paper consists of areas (segments) that represent different (sometimes overlapping) affordances. This is done in a pixel-wise manner by assigning presence-probabilities for all different affordances to every pixel in the image. Affordance segmentation is more challenging compared to object class segmentation due to three aspects. 1) Affordances are not disjoint, i.e., the presence of an affordance does not exclude the presence of other affordances. This requires us to make use of multi-class segmentation. 2) Affordances can refer to very small structures, which are only parts of objects. 3) There are no large-scale datasets available yet. This means that the problem cannot simply be addressed by training a semantic segmentation model on new data. These challenges in conjunction with the practical applicability makes affordance segmentation an interesting and challenging research topic.

There are multiple approaches to identify affordances, each having their own pros and cons. For example, affordance assignment can be done by considering object geometry [2]. If performed purely in this way, this leaves out all semantic object knowledge which we–humans–have access to when assigning affordances. Therefore, in this work we pursue a semantic, context dependent approach and show that this is very powerful even when only considering single 2D-RGB images. This method leverages knowledge about the relation between affordances and object parts and employs a convolutional neural network (CNN) to assign action affordances probabilistically to the pixels of a new image. The advantage of this is that it combines a knowledge-based approach with 2D images making it useful

Email address: timo.lueddecke@phys.uni-goettingen.de (Timo Lüddecke)

for a wide range of robotic applications. Naturally, limitations of all single image-based approaches apply to this work as well, e.g., in the case of 3D to 2D projection inconsistencies. However, such problems can often be solved by incorporating other methods, e.g., fusing the segmentation with a structured light sensor output and doing some post-processing.

In summary, the contributions of this work are as follows:

- A method to generate a large set of action affordances from semantic segmentations using a selective binary cross entropy loss function.
- A method to fuse simulated data with real data to improve generalization and their empiric comparison.
- An extensive evaluation of UNet- [3] and PSPNet-like [4] convolutional neural networks for affordance segmentation taking both runtime speed and prediction quality into consideration.
- Validation of the method in a robotic scenario that consists of images that had *not* been part of the training set.

For reproducibility of our proposed approach and as a starting point for using our method in practice we published the source code and pre-trained models online. [1]

2. Related Work

In the past affordances have been addressed in multiple studies, both in computer vision and in robotics. In the following, we will review existing work from psychology before discussing different approaches for affordance segmentation and relate those to our approach.

2.1. Affordances in Psychology

The term affordance originates from the field of perceptual psychology. It was originally coined by J.J. Gibson as part of his direct perception theory. He defined affordances as potential actions between an animal and its environment, given the capabilities and state of the animal and the structure of the environment [1]. The set of all affordances of an animal is called its ecological niche. Based on the seminal work of Gibson [1], later research sought to refine the question of what constitutes an affordance also addressing philosophical implications. Early theories considered affordances to be properties of the environment which are used by an animal [5]. Chemero [6] argues against this and proposes to consider affordances not as properties at all but as relations between particular aspects of animals and situations. Recently, Rietveld and Kiverstein [7] suggested applying the notion of affordances in a much broader context and on a higher level: They argue that any skill, not only motoric abilities, that a form of life possesses establishes a set of affordances. They use form of life instead of animal to emphasize cultural differences within a species of animals. A review on the historical development of the term affordance and a discussion of competing models was carried out by [8] and more recently by [9].

In addition to these theoretical and philosophical aspects, several studies have assessed the perception of affordances in humans and animals experimentally. Warren [10] suggests differentiating the transition between affordances on the basis of critical and optimal points and finds experimental evidence that the perception is affected by the capabilities of the agents. A mathematical formulation of affordances is obtained by expressing critical points as functions of ratios of body parts (leg length) and environmental quantities (riser height). Cole et al. [11] studied differences in perception of the three types of affordances leap, arm-swing and crawl in humans. They found participants to systematically underestimate their abilities in launching actions (leap and arm-swing) but improved their judgement after actually performing the actions.

2.1.1. Relation of our work to affordances in psychology

Our approach can be seen as a partial implementation of the direct perception paradigm by Gibson. Light represented by an RGB image is directly transformed into actions without explicit intermediate representations (the activations in the layers form implicit representations, though) in a pure feed-forward fashion. Instead of binary categories, affordances are represented probabilistically as suggested in [12]. One of the affordances analyzed by our system is *illumination*, which does not imply an immediate motor action. This can be seen as an example of the richer set of affordances enabled by animal skills as suggested by [7]. Wagman et al. [13] studied a multi-affordance environment and found evidence that humans sense means-end relations between affordances. Our work shares the trait of predicting multiple affordances simultaneously but does not encompass hierarchies.

2.2. Affordances in Computer Vision

A common approach for affordance segmentation is to predict affordances of whole objects [14, 15]. Akin to these is the work of Ye et al. [16] who detect bounding boxes of affordances using a two-stage approach consisting of region proposal and CNN-feature-based affordance recognition. Sawatzky et al. [17] learn to segment affordances on weakly supervised data. More recently, Sawatzky et al. [18] address a few-example-setting using label transfer on images of objects. Given a query image, a similar example is retrieved from a database. Then the segmentation is transferred to the query and refined by a CNN. It seems unlikely that this approach generalizes beyond object level as whole scenes vary much stronger and finding a similar example scene would require an extremely large database.

Affordance datasets have been proposed before. The UMD RGB-D part affordance dataset [19] focuses on objects only, was captured in a controlled lab environment and is intended for approaches that rely less on context but on depth information. The IIT-AFF dataset [20] might seem more applicable for us. However, it involves only 10 object classes which is too limited for studying whole scene affordances. Furthermore, it is sparse

[1] https://gitlab.gwdg.de/cns-group-public/aff-seg

Table 1: Comparison of related algorithms with # denoting the number of used affordances.

Approach	#	input	output
Grabner et al. [2]	1	RGB-D	per voxel
Gupta et al. [21]	4	RGB	per pixel
Savva et al. [25]	7	Video	per voxel
Rhinehart and Kitani [26]	6	Video	per grid-cell
Roy and Todorovic [27]	5	RGB	per pixel
Our approach	**12**	**RGB**	**per pixel**

as affordances are only assigned to objects but not other parts of the environment (e.g. the floor). Due to these shortcomings we decided to construct our own dataset that is more suitable for our analysis.

Affordances can also be predicted in form of (human) poses. This scheme is adopted by Gupta et al. [21] making use of scene geometry and in Grabner et al. [2] specifically for chairs and by Fouhey et al. [22] using video. Similarly, Kjellström et al. [23] learn affordances by observing interactions with objects in videos. More recently, Wang et al. [24] analyze popular sitcoms to learn pose-based affordance prediction.

The idea of "action maps" is closely related to affordance segmentation. However, the former tends to be more specific, e.g., by referring to very concrete objects and the set of considered actions is fairly small. Examples of these approaches are Savva et al. [25] who generate seven different "action maps" by tracking people in RGB-D video footage and Rhinehart and Kitani [26] who learn 6 action maps through analyzing egocentric video recordings.

The method proposed by Roy and Todorovic [27] is similar to ours as it also generates pixel-wise maps given an RGB image. Their model learns intermediate representations for depth, surface normals and object classes, which are then employed to carry out the affordance map prediction. The learning of these representations is actively enforced during training, i.e., the method requires additional data during training, while our method only needs RGB images and affordance map ground truth. Another difference to our work is the set of considered affordances.

Concurrently to this work, a similar method is proposed by Do et al. [28]. It deviates from our method as it conducts affordance and object detection jointly and is trained on the sparse IIT-Aff dataset, while we are interested in entire scenes "in the wild".

A common trait of many approaches is that they are constrained to a specific domain or the set of considered affordances is small (see Table 1). Our dataset consists of 22,000 images, which is significantly more than IIT-AFF (8,835 images) and UMD RGB-D (more than 10,000 annotated images but only 105 object instances). As stated in the introduction, our approach is context-based and we do not explicitly consider object geometry (for such approaches see [2] or [27]).

2.3. Affordances in Robotics

Affordances play a major role in cognitive and developmental robotics since they are crucial for a robot allowing it to explore and interact with the environment fully autonomously. Affordance research in robotics has received a lot of attention during the last decade and led to many contributions. For detailed recent surveys see Min et al. [29] and Zech et al. [30]. In the following, we will only briefly review main concepts and approaches in affordance-related research in robotics and relate them to our approach. Also, we will be only concerned with object/tool related affordances and will not talk about social affordances (e.g., [31]).

There are mainly three *categories* of object-related affordances [30, 29]: 1) grasping affordances, 2) manipulation affordances, and 3) traversability and locomotion affordances. Grasping affordances are mainly concerned with how to find a graspable point of the object for a particular object manipulation [32, 33, 34, 35]. Manipulation affordances relate to manipulations of single objects, e.g., push, pull, turn, lift, etc. [36, 37, 38, 39, 40] or to the interaction with multiple objects and/or tools, e.g., stack, sort, tool use, etc. [41, 42, 43, 44]. Traversability and locomotion affordances relate to motion affordances for mobile robots, e.g., cross, climb, select foot placement, etc. [45, 46, 47, 48]. While most of the studies focus on one type of affordance or consider only few affordances (mostly in the range of two to four [30]), in our study we deal with a total of 12 affordances within three *categories* of affordances: manipulation-related affordances (break, grasp, pull, tip-push, place-on), traversability- and locomotion-related affordances (sit, roll, walk, obstruct, support) and two object-related affordances, which do not belong to the three main types of affordances stated above (illuminate and observe). Also, most of the studies assign affordances to a whole object [29] and only few studies consider parts of objects [41, 42, 43], whereas in our study we investigate affordances of both objects and their parts.

Affordances can be acquired or learned in several ways [30]. The most common strategy to learn affordances is exploration, which is inspired by the cognitive development of children [49, 50, 51, 52]. Other strategies include supervised learning approaches such as programming by demonstration [14, 53, 54] or by providing ground truth data to an agent [55, 56, 57]. Some other approaches do not use learning at all and use hard-coded affordances [58, 59, 60]. In our study we use a supervised learning approach, which is much easier to implement and is less time-consuming as compared to exploration.

Different learning strategies have been used to implement affordance learning ranging from unsupervised learning methods such as self organizing maps [61] and K-Means clustering [62, 63], and reinforcement learning techniques [41, 64, 65, 66] to supervised learning techniques such as support vector machines and multi layer perceptrons [67, 55, 68, 69] and recent approaches using CNNs [27, 57, 70, 20]. In our study we also employ CNNs, however, as already discussed above, we only rely on single 2D images. In [27, 70, 20] RGB-D images are used while in [57], in addition to RGB images, motion data were employed being specific to autonomous-driving applications.

2.4. Simulated Training Data

Previous works have studied the effect of training on simulated data. For the related tasks of stereo matching and optical flow, Mayer et al. [71] discuss several data generation schemes. However, these tasks are quite different from affordance segmentation as they require alignment of image regions instead of semantic understanding. Saleh et al. [72] address semantic segmentation, but they only consider a scenario where all training data is synthesized while we explicitly focus on how to merge simulated and real data.

3. Methods

In this section we describe the sub-modules of our method. The general pipeline of our approach is outlined in Figure 1. First, we explain how we transfer object (part) labels to affordance labels. Then the model for generating simulated training data is defined. In section 3.3 we provide an overview of the CNN architectures and the loss function we employ in this work. Finally it is explained how the system can be used in a robotic framework.

3.1. Part Labels for Affordance Definition

We use real as well as simulated data for training (and testing) our system. In the following, we will describe how to assign affordances to real scenes, which is the more complicated case. Generation of simulated scenes is described afterwards, where the same principles for affordance assignment are employed.

Our method requires access to object and part segmentations, with as fine-grained labels as possible. From these annotations we derive affordances using a transfer table while obeying these principles:

1. Affordances should be *meaningful* (in some sense) for robots or humans.
2. We require that affordance names are specific. For example, *open* is a very unspecific multi-action. *Tip-push* implies a very well defined motion, of approaching a surface (e.g. a button) with a finger (mostly the index finger). Therefore we consider *tip-push* to be specific enough.
3. Actions can have a hierarchy, but lead to the same final outcome: E.g. a house can be entered, a door, which is a part of the house, can be opened and the door's handle, as a part of the door can be pulled. All of this will be done to enter the house, where the pulling of the door handle is here the action at the lowest semantic hierarchical level. Only this level will be considered to label affordances in this study.

Considering these guiding principles as well as the underlying dataset (ADE20K), we define a set of 12 affordances: obstruct, break, sit, grasp, pull, tip-push, illumination, observe, support, place-on, roll and walk. They are presented along with short descriptions in Table 2.

Table 2: Description of the set of affordances used.

Affordance	Description
obstruct	vertical surface that prevents locomotion. *e.g. wall*
break	detachable objects that can easily be damaged or destroyed *e.g. vase*
sit	surface a human can sit on while having the feet on the ground *e.g. seat cushion*
grasp	detachable objects that can be encompassed with one hand or only few fingers and be moved with one arm.*e.g. vase)*
pull	surfaces that can be pulled through a hook or pinch movement of the fingers (all directions). *e.g. knob, handle*
tip-push	surfaces that trigger some action when being pushed. *e.g. button-panel*
illumination	surfaces that emit visible light.*e.g. bulb*
observe	surfaces that present information or art, i.e. that can be read or watched. *e.g. display*
support	stable surfaces that provide support for standing (for the agent) except ground. *e.g. wall*
place-on	raised surfaces where objects can be placed on (this excludes the ground). *e.g. tabletop*
roll	surfaces that can be used with wheels. *e.g. road*
walk	surfaces a human can walk on. *e.g. grass*

3.1.1. Object Parts

An affordance most often refers to only a part of an object. For example, it is the surface of the table that affords placing an object there, but not the table legs. Thus, we define affordances part-wise. ADE20K [73] is currently the only sufficiently large dataset that resolves objects into their parts, hence it is used in this work. Although MsCOCO [74] has many more images, this dataset is not suitable for our approach as it only provides 80 object classes.

3.1.2. Transfer Table

We manually define a mapping from object and object-part labels to 12-dimensional affordance vectors. Each dimension in this vector corresponds to one affordance. Each vector element can have a value between 0 and 1 or can be undefined. The latter is useful if the presence of an affordance cannot be reliably inferred from the object or part name. Clearly, multiple affordances can be present simultaneously, so, in contrast to semantic segmentation, the vector does not have to sum to one. This mapping from objects and parts to affordance we call the transfer table. It consists of round 250 rows that have been manually defined by us. This table serves as the basis for turning segmentation ground truth data from ADE20K into affordance ground truth data (which will be later fed to the CNN). Adding a new affordance would require updating the transfer table with an additional column. For this, compatibility (yes or no) between the new affordance and all approximately 250 object needs to be defined.

The transformation from an object-part segmentation $\mathbf{O}$ into an affordance segmentation $\mathbf{A}$ is carried out pixel-wise. The original label of a pixel is searched in the transfer table T and replaced with the associated affordance vector from the table, if there is a matching entry in the table, i.e. $\mathbf{A_{ij}} = T(\mathbf{O_{ij}})$. Otherwise we acknowledge that no affordance can be assigned. To

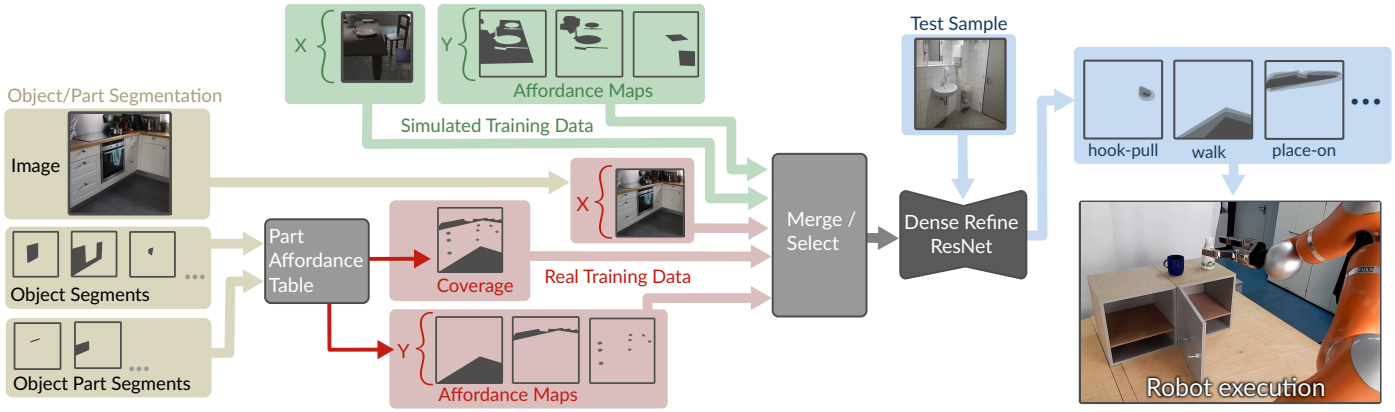


Figure 1: Our approach: We train a neural network to predict a set of affordance maps (Y) from a single RGB image (X) using a loss function that allows for incomplete data by incorporating a coverage map. Real (red) and simulated training data is mixed (green), with the real data being generated from object part segmentations using a manually specified part-affordance table. The output of the algorithm can serve as an input to many robotic tasks.

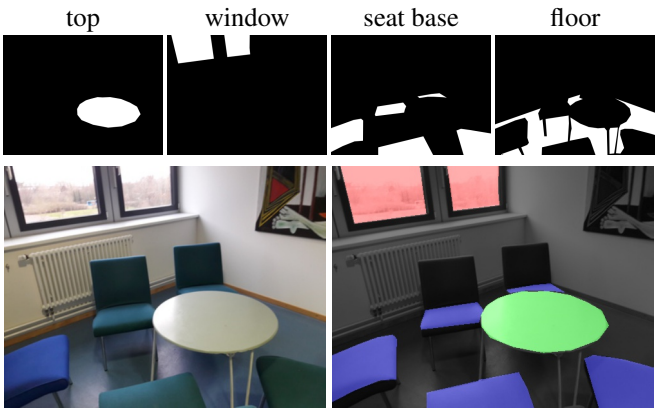


Figure 2: Example usage of the transfer table: Given a set of object-part segmentations (e.g. from ADE20K), the transfer table is queried to generate 12 affordance maps of which three are shown (sit (blue), place (green) and illumination (red)). In general, affordances can overlap, although this is not the case here.

make the latter accessible later on we store a binary mask tensor $\mathbf{M}$, that encodes the validity of the assigned affordances:

$$\mathbf{M}_{ij} = \begin{cases} 1 \text{ if } O_{ij} \in T \wedge T(O_{ij}) \neq \text{undef.} \\ 0 \text{ otherwise} \end{cases}$$

This mask is subsequently leveraged in the cost function to select valid pixels (see section 3.4). A sketch of how the transfer table works is shown in Figure 2.

Table 3 below shows an excerpt from the transfer table. Each cell can have three values: Affordance present (1), absent (0) or "unknown" ( ). A door and a swivel chair are always obstruct-able. However, only some parts of the swivel chair are sit-able and grasp-able so the corresponding fields are left blank, indicating uncertainty. The same holds for door with break, illumination and support, because it might be a glass door.

object	obstruct	pinch-pull	break	sit	grasp	illumination	support	place-on	...
swivel chair	1	0	0			0	0	0	...
door	1	0		0	0			0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Table 3: Excerpt from the transfer table

By applying the transfer table to the original segmentation labels of ADE20K we obtain our affordance maps. The distribution of the different classes is depicted in Figure 3.

3.1.3. Data Augmentation

Scene quality in ADE20K substantially varies. This leads to the situation that only a rather small number of good-quality training samples can directly be generated from ADE20K. Therefore, we augment the dataset by cropping out image patches from an original image where we then vary color and contrast within such a patch. For large images, this can lead to multiple non-overlapping crops, which can be considered individual samples. The augmentation is carried out online, i.e. therefore in each training epoch completely new samples (new crop, new color, new contrast) are fed into the network.

3.2. Simulation Model

Transferring labels from real-image object parts has some disadvantages: Maps are incomplete and some affordances occur rarely. We overcome this problem by generating a new dataset of simulated images. It relies on a probabilistic scene model of a living room and a kitchen with several constituents of the scene being randomized. Hence, we can generate strongly varying images of the scene. More precisely, the randomized variables in our model are object material, -position, -shape, scene illumination, and perspective.

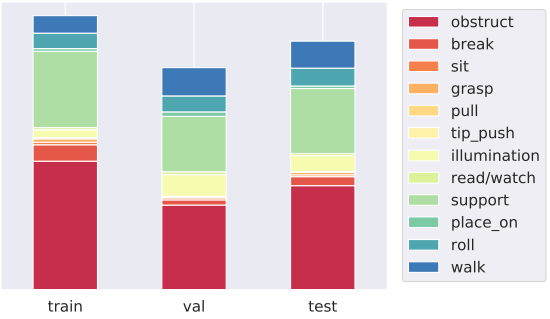


Figure 3: Occurrence distribution of the 12 affordances on the different splits. The classes are highly imbalanced, which poses a challenge for training the network. Some classes are so rare they cannot be seen in the diagram.

Object material. Objects can have different materials. A table surface, for instance, can be composed of plastic, wood or glass. Glass can be transparent or opaque. During scene generation, every object in the scene gets a randomly assigned material, with possibilities being constrained based on the object name.

Object positions. Several objects are randomly positioned in the scene and relative to other objects. Examples are a plate on a table, which is dependent on the table's height and a fork and knife, which are positioned relative to the plate.

Object shape. For some objects we define key model shapes and interpolate between these key shapes when a scene is generated. E.g. we interpolate between a chair with rounded edges and a chair with sharp edges.

Scene illumination. The world during day-time looks entirely different than at night. We account for this by varying light from the outside as well the intensity of indoor and outdoor illumination.

Perspective. Having obtained a variable scene model, we still need to simulate the process of photography by projecting the 3D scene onto a 2D plane from many possible viewpoints. For this, it is desirable to use viewpoints that sample mostly interesting aspects of the scene (e.g. multiple objects and sufficient distance), while avoiding irrelevant projections (e.g. view of the ground only) or invalid perspectives (e.g. taking an image from behind a wall). We address this challenge by sampling the camera's position randomly along a fixed heuristically assumed trajectory and introducing slight variances with respect to the position.

For each object or object part we manually define corresponding affordances and render the corresponding affordance maps in a second pass by changing the objects' materials.

This procedure allows us to generate an arbitrary number of training samples each providing consistent, fully covered affordance maps. This way, we can extend the training set by many additional images.

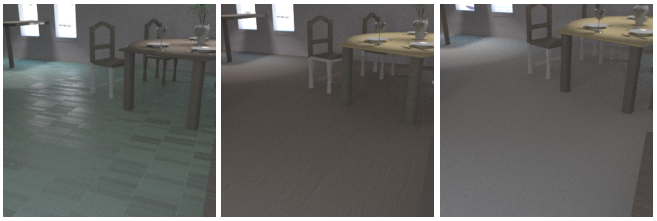


Figure 4: Three simulated samples where the perspective is fixed while other variables (e.g. floor) were randomly sampled.

The simulation model is implemented in the open source 3D modeling and simulation software blender[2] using its scripting API and the unbiased, physics-based renderer cycles. Figure 4 gives an impression of the variability of the simulated samples. The dataset obtained using this method involving 2280 scenes is subsequently denoted by Sim^T. While we can draw an infinite number of samples from the simulation, the critical task is to introduce variability. Hence, the number of artificially generated scenes represents a trade-off between performance gain and effort we put into designing the simulation environment.

3.3. CNN

Affordances are context dependent. An example makes this clear. We could ask whether a surface is walk-able or suitable to place things? If we now compare the ground with a table surface, we find that, locally, both are flat and uniform. Only context may resolve the difference between them. Walk-able surfaces, for example, may be accompanied by cars and trees, a table surface, on which we would put things — on the other hand — is often flanked by e.g. chairs. This leads to the requirement that the receptive field of a pixel should, ideally, cover the whole image because even distant pixels might be decisive for a local affordance.

This could well be in conflict with the second essential requirement, which demands that image details must not get lost during the forward pass of the network. Hence, object- and part-boundaries should be preserved. For example, many affordances concern rather smaller image aspects (e.g. a knob for pulling) and these aspects should not be lost by the network's operation.

With these requirements in mind, we propose and compare multiple neural network architectures. As it might be advisable to choose the architecture depending on the application we conduct an experiment to guide this decision in section 5.4. Our choice of models can be divided into two branches: PSPNet-based [4] and U-Net-based [3].

Both are deep convolutional neural networks that predict densely, i.e. per pixel. The former relies on the work of Zhao et al. [4], which proposed the pyramid pooling module (PPM). Evolving from the fully-convolutional network (FCN) [75], this model extracts features using a conventional encoder, applies

[2]https://blender.org

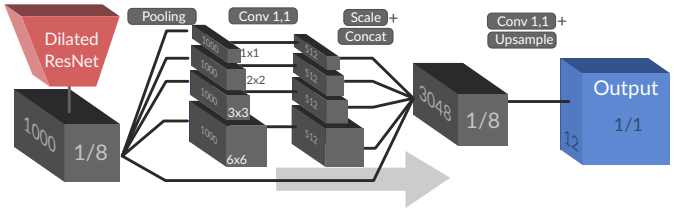


Figure 5: Architecture of the PPM module in the PSPNet model.

the PPM on the obtained feature maps and upscales to the output tensor. In our case, feature maps are extracted using a 102-layer dilated ResNet encoder[3], hence the model will be called *P-102*. Our implementation of PSPNet focuses on the architecture and avoids training tricks like an auxiliary loss (see Figure 5). The PPM ensures that contextual cues can be processed, which we believe to be an important trait. Due to the small spatial resolution output of the PPM, predictions tend to be blurry, if no further processing (such as an conditional random field [76]) is applied. Also, the PSPNet-based model is fairly complex, requires a lot of memory for training, resulting at smaller batch sizes, and is slow at inference.

Therefore we designed an alternative network following the U-Net [3] paradigm, which has the advantage of being faster to train and to run, compared to [4]. This model is depicted in Figure 6. The encoder is based on ResNet [77] and the architecture adopts the idea of refinement modules [78]. It had been shown by these authors that ResNet50 together with refinement modules successfully generates sharp object proposals, because refinement modules offer an elegant way for merging local with scene-level information. Thus, here we use a modified version of the architecture from [78].

This model integrates abstract information from deep layers with the spatially more accurate representations still present in less deep layers. Here both input layers will deliver maps of the same image size where they are then first stacked on top of each other (concatenated along depth) and subsequently convolved with the learned filters to obtain feature maps. While the base model is fixed to be a ResNet, we experiment with several configurations: The encoder size is varied from 18 to 152 layers, which heavily influences the execution (and training) speed of the model. In the decoder, we only vary the number feature maps in the last two refinement modules: In addition to the normal setting with 32 feature maps each we introduce a "small" setting involving 16 feature maps each (see DS in Table 4). We initialize the ResNet encoder with features obtained from ImageNet [79] pre-training, but do not freeze any weights.

3.4. Cost function

We propose a novel cost function we call selective binary cross entropy. This cost function deals with two aspects: 1) affordances are often not unique and for a given pixel multiple affordances may exist simultaneously. Hence, we imply a binary (present vs not present) probability distribution for each

[3]We use this implementation https://github.com/fyu/drn

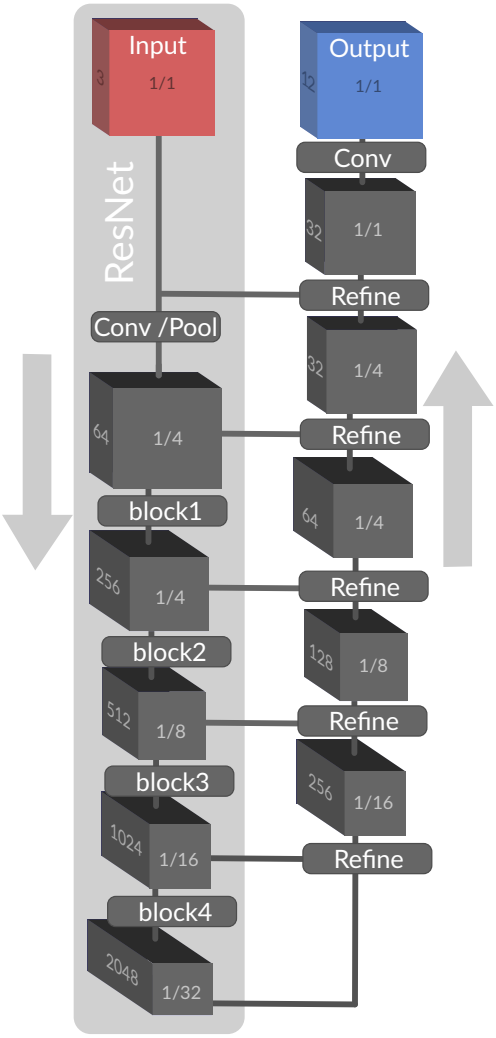


Figure 6: Architecture of the U-Net-based model. The ResNet encoder on the left involves four blocks. The boxes indicate the corresponding intermediate tensor sizes (number of channels and fraction of the image size).

pixel and each affordance. 2) For some parts of the image no affordances may be defined. For those, we cannot tell whether an affordance is present or not, because the corresponding object or part is not found in the transfer table. However, since we also generate the corresponding validity mask, we know the location of the invalid regions. The idea is to incorporate also this information into the cost function.

This means during optimization we search for a model that agrees with the ground truth, but only where the latter is defined. For some regions no ground truth is defined. Here the model is free to predict whatever it considers to fit best and is not falsely punished due to over-generalizing annotations.

This concerns objects and object parts where no decision about the presence of an affordance can be made based on the object or part name alone. For example consider a bench. Does it afford placing-on? This depends on whether the bench's sitting surface is even. Does a coffee table afford support? Larger tables might but smaller ones do not. In these cases, we prefer

to ignore those uncertain fractions of the data in order to avoid false annotations. This is implemented by masking in the loss function.

Both aspects from above lead to the fact that commonly-used cost functions for semantic segmentation cannot be employed here. Subsequently, we will formally derive the here used *selective binary cross entropy cost function*.

We annotate the ground truth matrix of an image for affordance $a \in \mathcal{A}$ and pixel $i \in \mathcal{I}$ with $\mathbf{Y}_{ai}$ and the associated model prediction is given by $\hat{\mathbf{Y}}_{ai}$. Then the binary cross entropy BCE is defined by: $\mathrm{BCE}(p, q) = -p \log(q) - (1-p) \log(1-q)$. This is summed up to render a scalar loss (cost), which captures the average binary entropy over all affordances and the image.

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = (|\mathcal{A}||\mathcal{I}|)^{-1} \sum_{a \in \mathcal{A}} \sum_{i \in \mathcal{I}} \mathrm{BCE}(\mathbf{Y}_{ai}, \hat{\mathbf{Y}}_{ai})$$

So far this definition is compatible with non-exclusive classes, but it does not yet account for incomplete data. To achieve this, we mask the cross entropy matrix, excluding all regions where no (or indecisive) affordances are present, before averaging. Masking is a very efficient and simple way for removing the incompleteness ambiguities and we get the following loss:

$$\mathcal{L}^{\mathrm{m}}(\mathbf{Y}, \hat{\mathbf{Y}}) = (|\mathcal{A}| \sum_{i \in \mathcal{I}} \mathbf{M}_{ai})^{-1} \sum_{a \in \mathcal{A}} \sum_{i \in \mathcal{I}} \mathbf{M}_{ai} \mathrm{BCE}(\mathbf{Y}_{ai}, \hat{\mathbf{Y}}_{ai})$$

with $\mathbf{M}_{ai} \in \{0, 1\}$ indicating if pixel i is valid, i.e. if a corresponding entry is found in the transfer table.

Contrary to [80], the mask M (of each image) is defined for every pixel *and* affordance. This allows us to specify uncertainties in the mapping from object/part names to affordances in a more fine-grained way during the learning phase. In test mode, the mask is no longer required.

In Figure 7 we highlight how the mask loss leads to a different gradient. It can be seen that the gradient is zero where the mask is zero. Hence, relevant weights are changed with higher magnitude while irrelevant weights are ignored.

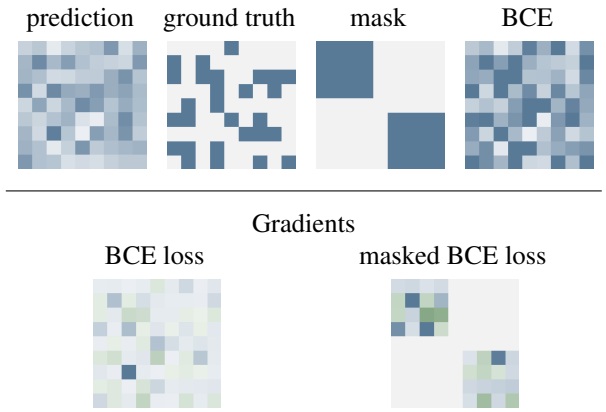


Figure 7: Illustration of how masking changes the gradient. The top row depicts network prediction, ground truth, a toy mask and the binary cross entropy computed from prediction and ground truth. In the bottom row the gradients resulting from conventional BCE and masked BCE loss are shown (green indicates negative values). Gradients are used to change the preceding layer's weights. We can see that masking leads to stronger changes in relevant regions while irrelevant weights are ignored (ie. not changed).

4. Experimental Setup

4.1. Evaluation Datasets

The training and validation samples of ADE^T are generated from the ADE20K *training* dataset [4]. The ADE20K validation dataset is used for testing. It contains 2000 scenes and will be called ADE^E. From this we manually pick 50 high quality (sharp, multi-object) images and transfer the annotations to affordances using the table. Then we let a person (expert) manually correct this according to the definitions provided by Table 2 by editing each affordance map individually with an image editor. Due to this manual correction, systematic errors of the part-to-affordance conversion procedure are punished during evaluation and we obtain a more realistic estimate of the error. Hence, this metric assesses a stronger form of generalization. This dataset will be called "Expert50" subsequently. While a number of 50 samples might seem small, note that the networks outputs probabilities for every pixel and affordance resulting in a few million predictions even in the expert dataset. Hence we consider this dataset reliable enough to be used for evaluation. Scores are determined by comparing predictions of our network on the test set with corresponding ground truth data according to the metrics that will be discussed next.

[4] Here you can use the interactive dataset browser to get an impression of how the samples look like: http://groups.csail.mit.edu/vision/datasets/ADE20K/dataset_browser/

4.2. Metrics

To quantify the performance, we use the intersection over union (IoU) metric (sometimes referred to as Jaccard Index), which is defined as follows:

$$\text{mean IoU}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \frac{\sum_{i \in \mathcal{I}} 1[\mathbf{Y}_{ai} = 1 \wedge \hat{\mathbf{Y}}_{ai} = 1]}{\sum_{i \in \mathcal{I}} 1[\mathbf{Y}_{ai} = 1 \vee \hat{\mathbf{Y}}_{ai} = 1]},$$

following the notation introduced in section 3.4 with $\mathbf{Y}$ denoting ground truth and $\hat{\mathbf{Y}}$ a model's prediction. $1[\cdot]$ is the indicator function defined as:

$$1[c] = \begin{cases} 1 & \text{if } c \\ 0 & \text{else} \end{cases}$$

Maximal IoU would be 1.0. It is important to note that IoU is measuring the overlap with the ground truth image segment area and punishes both, lack of overlap in the labeling as well as false positive outside-of-segment labeling. To compute this, the probabilistic predictions of the network must be binarized. In our experiments we use two fixed threshold values: 0.1 and 0.5 for this.

Additionally, we report the mean average precision. This metric has the advantage of not relying on a single threshold

level but averages over multiple levels. It is based on the precision P defined as follows

$$\mathrm{P}(\mathbf{Y}_a, \hat{\mathbf{Y}}_a) = \frac{\sum_{i \in \mathcal{I}} 1[\mathbf{Y}_{ai} = 1 \wedge \hat{\mathbf{Y}}_{ai} = 1]}{\sum_{i \in \mathcal{I}} 1[\hat{\mathbf{Y}}_{ai} = 1]},$$

which is evaluated at several recall levels to obtain an average precision. This is equivalent to the area below the precision-recall curve. The mean average precision is obtained by computing an average score over all (affordance) classes.

4.3. Implementation Details

The models are trained on a single Geforce 1080 Ti GPU or Titan V with a pytorch backend [81]. Weights are updated using RMSprop [82] with mini batch sizes of 16 (or 8 for large models) and a learning rate of 0.0001. The training is stopped after 25 epochs.

4.4. Configuration

We assess various components of our model with respect to their impact on the performance.

Encoder Since a large fraction of the computational budget is dedicated to extracting features using the encoder, it seems justified to investigate different choices in more detail. For our U-Net based model, we constrain the analysis to three common ResNets [77] having 18, 50 and 152 layers, since they provide a trade-off between accuracy and speed, which is suitable for our task. While a larger encoder tends to exhibit a better performance it might also be more sensitive to overfitting due to its larger number of parameters. Depending on the number of layers these models will be referred to as R-18, R-50 and R-152. The PSPNet is only evaluated in one configuration with a 105-layer encoder involving dilated convolutions, it is called P-105.

Decoder size Analogously to the encoder, also the decoder can be configured in different ways. While we keep the number of 5 skip connections constant, we vary the number of feature maps each decoder uses.

Masked Loss Above we described how to take care of undefined values in our loss function. In our experiments we empirically evaluate whether these changes actually lead to an increase in performance.

Pre-training: Knowledge acquired for recognizing images can be leveraged in affordance segmentation. Does the model benefit from pre-trained features or is the architecture (ResNet) good enough as a prior?

5. Results

5.1. Ablation and Additions

First we conduct an ablation study to identify the best performing configurations and validate the impact of various modifications (see Table 4). The encoder is fixed to be R-50, i.e. a ResNet with 50 layers.

In Table 4 we present our analysis on various modifications regarding sampling, loss and architecture on the base model

Table 4: Ablation study. This experiment has been conducted exclusively on R-50 and ADE^T as training dataset. DS: decoder size, IMP: importance sampling, SEL: scene selection, PRE: ImageNet pre-training, M: masked loss, CM: class mean, CW: class weighting. For IoU, 0.1 and 0.5 are binarization thresholds. The expert columns refer to manually corrected samples as explained in Section 4.1.

							ADE^E			Expert50		
							IoU			IoU		
DS	IMP	SEL	PRE	M	CM	CW	@0.1	@0.5	mAP	@0.1	@0.5	mAP
			✓	✓			41.5	**45.6**	**61.8**	**36.5**	**37.2**	**53.4**
	✓	✓	✓	✓			35.6	38.8	51.5	35.2	35.1	51.1
	✓		✓	✓			41.8	44.9	54.0	34.5	34.2	52.9
			✓				42.4	42.3	58.7	33.4	32.9	48.9
				✓			34.7	36.3	51.4	33.0	28.7	50.9
			✓	✓	✓		35.3	39.1	54.5	31.1	30.0	48.9
			✓	✓		✓	37.7	39.6	53.8	31.0	29.4	48.0
s			✓				**43.0**	41.1	58.4	33.4	31.0	48.4
s			✓	✓			41.1	44.7	50.5	35.3	34.7	51.3

R-50. Abbreviations in the text refer to the respective table columns. The columns of this table indicate different configurations of the respective model as discussed in Section 4.4 and the associated scores on: ADE^E and the expert dataset.

First, we empirically verify the utility of masked loss. With mask, scores tend to be higher by around 2 to 4 percentage points. Further additions address the problem of infrequent classes, i.e. those affordances that are rare and cover small areas. One natural way to tackle the unequal distribution of classes in the dataset is giving less frequent classes more weight. We experiment with two mechanism to achieve this. Class-specific weights (CW) in the loss and Class-Mean loss (CM). In the former case we multiply with manually specified, class-specific factors in the loss tensor before summing it up to a scalar. The factors roughly express how rare a class is and the idea is to compensate for rare classes. In the latter case, instead of computing the loss individually for every pixel, we compute a mean over class-specific losses, resulting in each class being weighted equally. Since both metrics also average over class-specific scores we would expect them to increase, as rare classes should perform better. However, this is not the case. We even observe a substantial drop in performance if CM or CW is enabled. An alternative to manipulating the loss function is to change the data that is fed to the network during training. Instead of showing a uniformly sampled cropped image exactly once per epoch, we changed the sampling to include on rare classes in the crop with higher probability (IMP) and even excluded images that do not contain specific classes (SEL). Also, these modification did not lead to a better performance but are kept in the paper for completeness. We conclude that data quantity outweighs data quality at least for this task.

During experimentation we found the choice of the optimizer and its learning rate to be crucial for the performance of rare classes. High learning rates drive the optimizer into a minimum where rare classes are never predicted. Hence, in the presence of highly imbalanced classes it seems advisable to use dynamic learning rates as conducted by RMSprop and other more sophisticated gradient descent algorithms.

Table 5: Comparison of combined training strategies. The '+' denotes joint training while → indicates multiple subsequent training procedures. The expert columns refer to manually corrected samples as explained in Section 4.1

			ADE^E			Expert50		
			IoU			IoU		
model	train data	M	@0.1	@0.5	mAP	@0.1	@0.5	mAP
R-50	ADE^T	✓	41.5	45.6	61.8	36.5	37.2	53.4
R-50	$ADE^T + Sim^T$	✓	43.9	46.5	58.9	37.4	37.4	54.3
R-50	Sim^T	✓	16.1	14.6	20.9	18.1	17.7	24.0
R-50	$Sim^T \rightarrow ADE^T$	✓	42.8	46.2	60.2	37.5	38.8	53.9
R-50	COCO $\rightarrow ADE^T$		**45.6**	41.7	59.4	34.7	32.1	48.9
R-50	COCO $\rightarrow ADE^T$	✓	43.4	47.1	54.3	37.1	38.0	53.6
P-105	Sim^T	✓	13.6	11.8	16.6	18.6	18.0	24.9
P-105	ADE^T	✓	44.3	**48.5**	**62.3**	37.9	38.2	52.7
P-105	$ADE^T + Sim^T$	✓	44.7	47.1	59.8	**38.2**	**39.4**	54.1
P-105	$Sim^T \rightarrow ADE^T$	✓	42.7	46.1	59.2	37.7	37.9	**54.5**

5.2. Simulated Data

With around 20,000 training samples the ADE dataset is fairly small compared to other datasets which are common in deep learning such as COCO [74] or OpenImages [83]. To compensate for the small number of training samples we generated simulated scenes. We propose and assess two methods of integrating simulated data with real world data:

- *Joint Training* In addition to models trained on individual datasets, we also train models on both datasets conjointly: The datasets are first concatenated and then randomly mixed. This way, each mini-batch for training can encompass samples from both datasets and each update of the network weights through the gradient will reflect this. We will refer to this training method by +, i.e. $A + B$ means joint training mixing samples from A and B.

- *Simulation Pre-Training* The network is first trained exclusively on simulated data. Subsequently, in a second training stage, the network is fine-tuned to the target dataset. To reflect the sequential nature of this training we denote it with an arrow (→).

As an alternative to simulated data, we pre-train networks on the COCO dataset [74], too. Our results are presented in Table 5.

The R-50 model seems to take advantage of additional data and improves most scores (with mAP on ADE^T being the exception). The increase is subtle, though. On P-105, performance on some scores even decreases when simulated data is used. Regarding the comparison between joint and pre-training we note that both methods perform on-par. Possibly, this could be explained by the limited variability in the simulated scenes we employ here. In general, simulating data seems a promising option if data is scarce but a more sophisticated simulation model, which is beyond the scope of this paper, would possibly be required.

Table 6: Performance on the Oregon dataset. `NYU` indicates the original roughly 50-50 train-test split, while `OWN` uses more samples for training.

		IoU		
model	train dataset	@0.1	@0.5	mAP
R-50	$\text{Oregon}_{\text{NYU}}$	26.0	16.8	34.5
R-50	$\text{Oregon}_{\text{OWN}}$	29.3	20.5	39.4
P-105	$\text{Oregon}_{\text{NYU}}$	30.1	22.1	42.6
P-105	$\text{Oregon}_{\text{OWN}}$	34.2	26.9	50.3
R-50	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{NYU}}$	53.0	56.4	72.4
R-50	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{OWN}}$	58.1	**65.5**	**83.5**
P-105	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{NYU}}$	**59.0**	63.0	82.4
P-105	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{OWN}}$	58.8	63.0	81.9
Roy and Todorovic [27]		49.6		n/a
Roy and Todorovic [27] using GT cues		53.2		n/a

We also find that pre-training on the COCO dataset yields a similar performance boost as the simulated dataset. However, considering the enormous annotation efforts of COCO we still think that simulation is a better option for pre-training. Comparing both scores involving COCO also confirms the usefulness of the loss masking, which was discussed earlier.

5.3. Comparison to State-of-the-Art Model

Although some previous work is akin to the idea of affordance segmentation we only find the work of Roy and Todorovic [27] to be suitable for a direct comparison. They evaluate their system on images of the NYUv2 [84] dataset, which provides depth maps in addition to RGB images as they rely on 3d scans for training their network. They collected pixel-wise annotations for a set of five affordances for all images of the NYUv2 dataset. This set does not directly correspond to our affordances thus we cannot directly run a network, which was trained with our method on their data. Instead, we make use of transfer learning: We take a trained model and replace the last layer responsible for classifying with a new one involving only five classes. The weights of all other layers are maintained. During training all weights are changed, i.e. the transfer learning only affects initializations.

The experiment involves different models and two different train/validation/test splits. One is the original NYUv2 split used in [27], which divides the 1,449 training samples in almost equally sized halves. The disadvantage of this split it that the training set is very small. Therefore we incorporate a more training-heavy split into the analysis, involving 1,100 samples for training and validation leaving the remaining samples for test.

The results reported in Table 6 reveal a strong improvement over state-of-the-art by models that were pre-trained using our method. Baselines, that were trained on the NYUv2 affordance dataset only but with encoders being pre-trained on ImageNet, performed much worse and don't even come close to state-of-the-art. Pre-training, in this case by join-training, on real and simulated data yields a large performance improvement. All of

Figure 8: Probabilistic segmentations generated using our method on the NYUv2 dataset [84]. Columns (left to right): original image, predictions of P105, and ground truth.

our models that were trained using this method outperformed the network of [27]. By switching to a more training-heavy split, we are able to obtain IoU scores up to 64.5, which is more than 12 percentage points above state-of-the-art results which used ground truth cues.

Also the qualitative results shown in Figure 8 are remarkable. In the bottom row, the P-102 network even discovers a place-able surface that was not annotated in the ground truth. Apart from that, the depicted predictions of walk-able, grasp-able and place-able are close to ground truth.

5.4. Model Comparison

Following the conclusions we can draw from the previous experiments, we evaluate a set of specifically tuned models for different purposes and discuss the specific trade-offs. This is necessary as affordance segmentation can be used in a variety of diverse environments, such as a mobile robot or a fixed installation. Each of these environments has its own requirements with respect to inference speed and CPU/memory demands. As the encoder carries out a large share of the computations we compare different encoder sizes, all based on the ResNet architecture. For comparison, we also report PSPNet scores. Following our nomenclature from above, the number after R denotes the number of layers. Results are shown in Table 7. We find that a larger encoder does not improve performance, while a smaller one only has a slight impact on performance.

5.5. Affordance-wise Evaluation

Table 8 reports individual scores for selected configurations. We can observe a strong variation in performance across the affordances. For example, obstruct, walk and support are learned well while grasp, place-on and observe turn out to be more challenging. In particular the small structures of pull and tip-push seem to be hard to predict as their scores are close to zero. This is probably due to these structures not only being small

Table 7: Comparison of models. Best performance is achieved by the most complex models, but even very simple models (R18) can perform well.

		ADE^E			Expert		
		IoU			IoU		
model	train data	@0.1	@0.5	mAP	@0.1	@0.5	mAP
R-152	Sim^T	16.0	13.9	20.8	18.6	17.8	25.5
R-152	$Sim^T \rightarrow ADE^T$	39.3	43.2	56.9	36.9	34.9	54.2
R-152	$ADE^T + Sim^T$	43.1	44.6	56.8	35.7	35.0	53.1
R-50	Sim^T	16.1	14.6	20.9	18.1	17.7	24.0
R-50	$Sim^T \rightarrow ADE^T$	42.8	46.2	60.2	37.5	38.8	53.9
R-50	$ADE^T + Sim^T$	43.9	46.5	58.9	37.4	37.4	**54.3**
R-18	Sim^T	16.9	14.9	23.0	20.0	17.6	27.5
R-18	$Sim^T \rightarrow ADE^T$	41.7	44.1	56.8	35.6	35.6	51.2
R-18	$ADE^T + Sim^T$	43.3	44.8	57.6	36.9	35.6	53.2
P-105	$Sim^T \rightarrow ADE^T$	42.7	46.1	59.2	37.7	37.9	54.5
P-105	$ADE^T + Sim^T$	44.7	47.1	59.8	38.2	**39.4**	54.1
P-105	$ADE^T + Sim^T$	44.1	46.8	58.7	**38.5**	38.2	53.9
P-105	$ADE^T + Sim^T$	**45.3**	**47.4**	**63.1**	38.4	36.8	53.8

but also rare. This means they need to be learned from less samples than other classes, which is more challenging. Additionally, their geometry might be harder to learn, in particular as features are more difficult to be recognized due to their small size. The more complex encoders of PSPNet and R-152 exhibit the best performance on these small classes, which might be due to their higher capacity allowing them to preserve details of smaller structures while encoding. At the same time, larger networks are more prone to overfitting, because of their larger number of parameters, which limits their overall performance.

Note, the scores of zero do not mean that rare affordances are never predicted as we can see in the qualitative evaluation. Also, due to the cross entropy loss encouraging cautiousness, the predictions are fairly weak such that false negatives are common if the same threshold value is used for all affordances. A possible way to overcome this in practice would be to use a very low threshold (<0.1) to generate candidates and then apply an application-specific post-processing to remove false detections. Alternatively, false detections could be filtered by rule-based approaches using heuristics.

As we can see in the last row, modifying the loss to be a mean over class-specific losses instead of a pixel-wise loss does not improve in rare classes. It seems that, for now, the most straightforward way to get better performance would be to use more training samples.

5.6. Speed Trade-off

In practice, networks cannot be arbitrarily large and the availability of memory is limited. This holds in particular if a robotic platform is used, which runs on batteries. To obtain an intuition on the respective trade-offs of the models we report inference time and memory footprint of selected models in Figure 9. Here, we constrain the batch size to one as we ex-

Table 8: Performance for individual affordances. The IoU threshold is 0.1. * indicates that class mean loss is used.

		obstruct		break		sit		grasp		pull		tip-push		illum.		observe		support		place-on		roll		walk		mean	
model	dataset	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP
P-102	$Sim^T \rightarrow ADE^T$	87.5	94.1	41.6	71.8	40.4	53.7	19.3	29.5	3.3	1.6	3.1	11.2	7.4	24.9	23.2	40.1	54.4	89.7	29.4	44.6	72.3	96.3	70.8	96.0	37.7	54.5
R-152	$Sim^T \rightarrow ADE^T$	86.4	94.1	38.6	69.2	30.6	41.3	19.0	34.2	1.2	0.8	8.6	5.4	5.5	30.4	28.0	47.5	53.5	89.1	24.6	45.9	75.0	96.6	72.1	96.2	36.9	54.2
R-50	$Sim^T \rightarrow ADE^T$	87.6	93.5	38.1	73.2	41.4	60.2	17.7	25.0	1.2	1.0	5.3	3.0	12.4	23.7	27.4	40.7	53.3	90.2	27.0	42.7	69.4	96.9	69.2	96.7	37.5	53.9
R-50	Sim^T	79.4	87.7	14.5	16.9	8.8	11.9	8.0	7.7	0.0	0.4	0.0	0.2	16.0	23.3	3.5	8.4	27.8	50.0	14.4	21.1	22.5	30.5	22.7	29.8	18.1	24.0
R-18	$Sim^T \rightarrow ADE^T$	86.6	93.7	36.2	67.1	35.3	46.8	19.3	30.2	2.3	1.0	2.7	1.3	11.1	19.8	24.2	42.4	51.9	88.5	21.0	32.1	69.1	96.1	67.5	95.8	35.6	51.2
R-50*	ADE^T	85.7	93.9	27.1	60.6	27.4	42.0	16.8	23.0	0.0	0.6	0.0	0.5	11.6	30.6	21.6	36.3	49.7	85.3	18.4	27.8	59.8	93.0	54.6	92.9	31.1	48.9

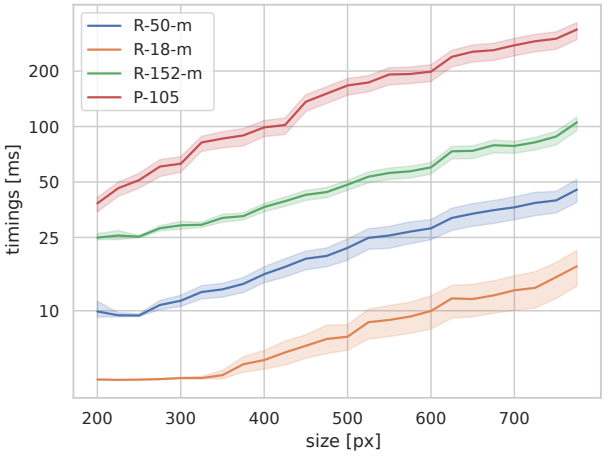


Figure 9: Speed vs. accuracy trade-off between the models. Note, we use the log scale of timings for better readability, the execution speed grows exponential with image size.

pect that responses of the system are required immediately in realtime systems.

5.7. Qualitative Evaluation

In addition to these quantitative findings we now discuss qualitative output of the models on various different scenes. In Figure 10 we show predictions of network P-102. For these results we do not use binarization but visualize the probabilistic predictions directly. All colored pixels encode the probability for the corresponding affordance by color intensity. This renders an assessment of the degree of confidence the model attains for any given pixel's affordance. Mixed colors indicate the presence of multiple affordances. The here used images challenge the network with difficult situations like front-lighting and transparent materials. We observe that the network generalizes well to these new situations, even small structures are mostly correctly predicted, for instance the drawer knobs in the bottom row. There are some false positive cases: The tissue and soap dispenser and the trash bin in the second row are marked as grasp-able. However, in order to know that these items are actually fixed one would need to physically inspect the scene (the trash bin could well be detachable), so, given only an image, these predictions are plausible. Another observation is that the prediction intensity is sometimes weak, as in the second column for place-on. However, depending on the application, this can overcome either by scaling the intensities or By some post-processing.

Figure 11 features a qualitative comparison showing the same scenes but predictions of different models. Here we can see that the predictions of the networks look fairly similar, which could be expected since both were trained on the same data. Furthermore, the subtle differences between the models are in accordance with the quantitative findings above.

5.8. Robot Experiment

In order to demonstrate the potential of the proposed system we performed a robotic experiment using a KUKA LWR robot-arm [85] with a 3-finger Schunk SDH hand [86]. The task for the robot was to conduct simple manipulation actions based on an affordance segmentation of the scene.

Execution of robotic actions is based on our previous work and was implemented using a library of manipulation actions [87, 88], which utilizes modified dynamic movement primitive (DMP) framework [89, 90], for trajectory generation. DMPs are formalized as stable attractor dynamic system and can generalize to new start and end points while being robust against perturbations. Specifically, here we used "pick-and-place", "take down" and "push" (to perform pull) actions to manipulate an object or an object part in the scene. For simplicity, in our study we used predefined object poses. In general, this can be done using existing state-of-the-art methods for pose estimation [91, 92], however, this is out of the scope of our study.

We show how affordance maps enable the robot to select and perform actions that are at that moment available in a scene. As we are not interested in any kind of planning problem, we let the robot "decide what to do" on its own, implementing some kind of playful mode. The robot had to select and perform two actions in a sequence. Hence, affordances need to be re-analyzed after the first action. However, this does not perturb execution of the second action due to the speed of affordance computation. Figure 9 has already shown that new affordances can be provided in realtime.

As in all quantification experiments above, affordances are assigned in a pixel-wise manner, where the color-intensity in the visualization (Figure 12) indicates the existence probability of the affordance. Hence, thresholding the affordance map enables determination of potential target locations, such that un-

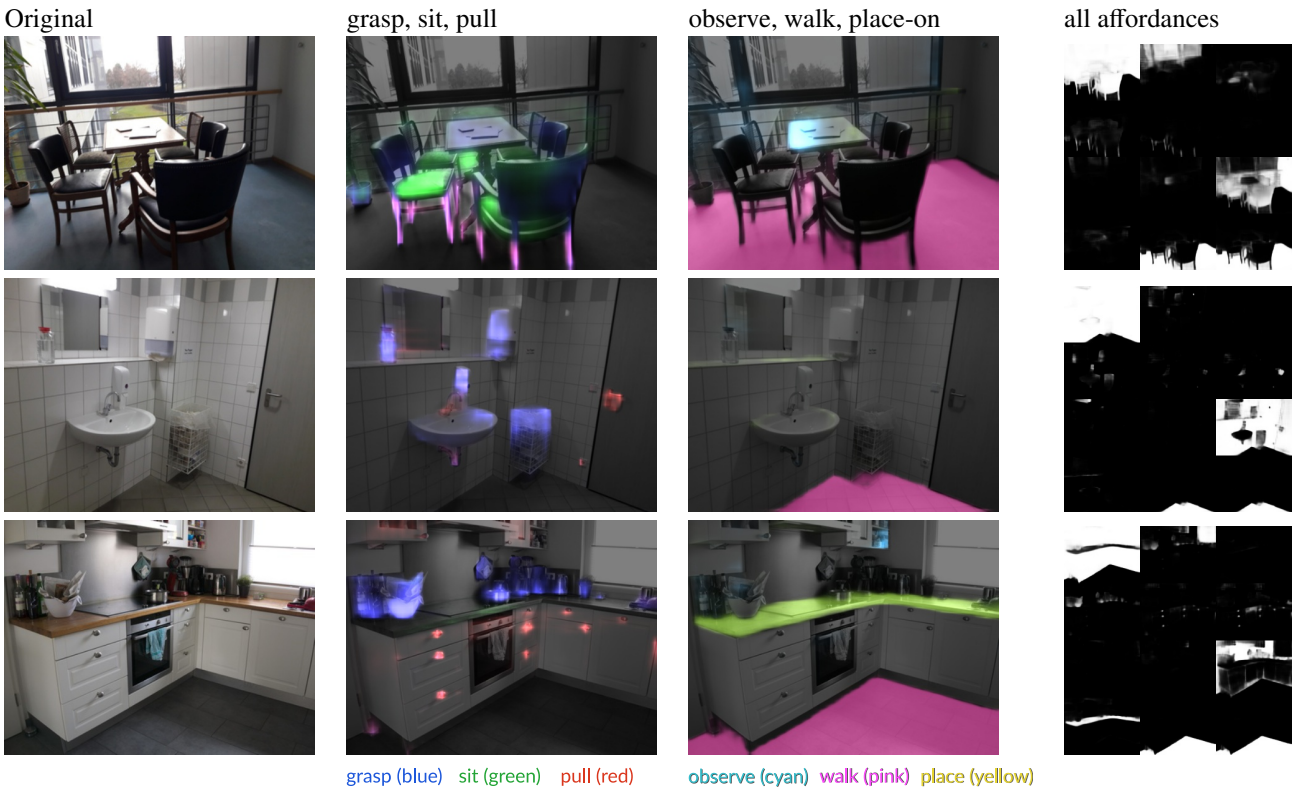


Figure 10: Probabilistic segmentations generated by the network P-102 jointly trained with simulated data. From left to right: Original Image; predictions grasp (blue), sit (green), pull (red); predictions of observe (cyan), walk (pink), place (yellow); all 12 affordance map predictions in their original form with the intensity corresponding to the presence probability of the affordance. Note, these images are not from the ADE dataset and therefore no ground truth is shown.

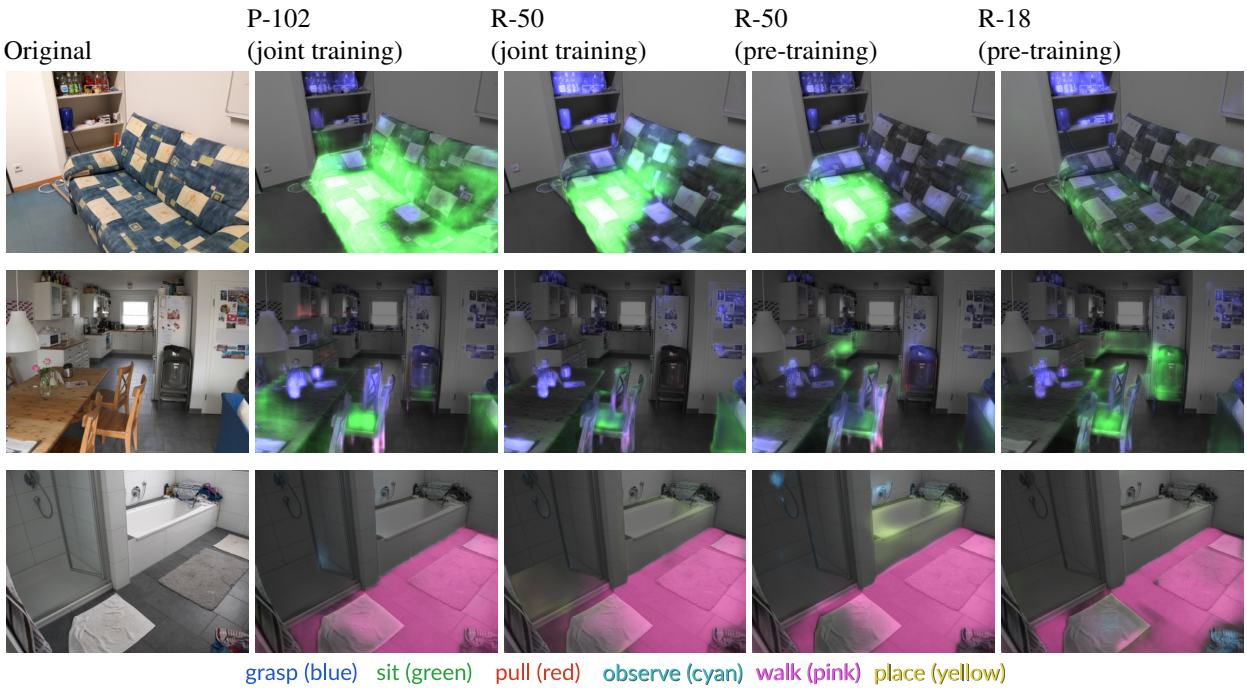
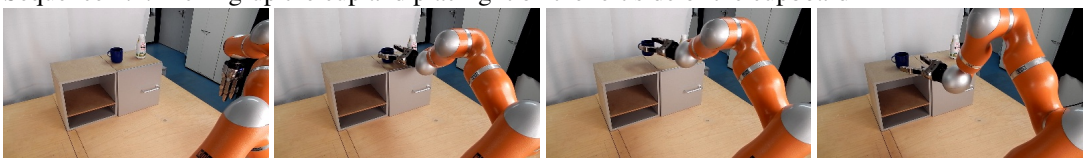
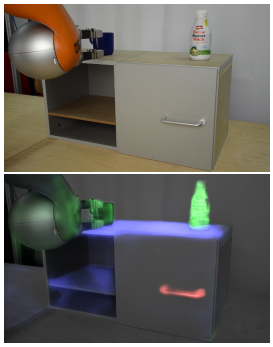


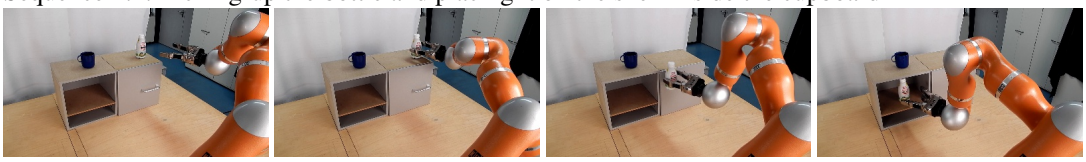
Figure 11: Probabilistic segmentations generated by different networks. Row 1 and 2: grasp, sit, pull, Row 3: observe, walk, place. Note, image sizes vary as the networks are fully convolutional and there is no need to rescale to a common resolution.

Sequence 1.1: Picking-up the cup and placing it on the left side of the cupboard

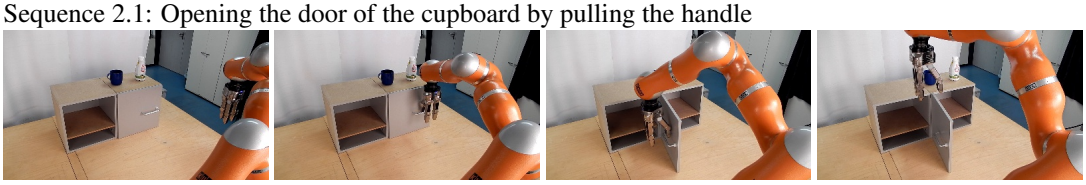




Sequence 1.2: Picking-up the bottle and placing it on the shelf inside the cupboard



Sequence 2.1: Opening the door of the cupboard by pulling the handle



Sequence 2.2: Picking-up the cup and placing it on the shelf inside the cupboard

grasp (green) place (blue) pull (red)

Figure 12: Results of execution of two action sequences based on affordance maps obtained using R-50. Affordance maps (left) and selected frames of robot action executions (right) are shown. For the complete experiment see supplementary video. Colors denote grasp (green), place (blue) and pull (red).

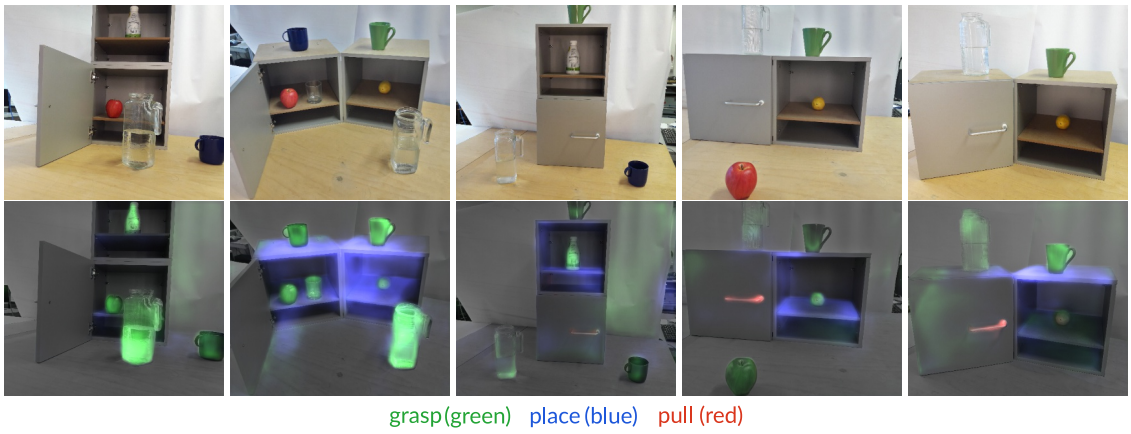


Figure 13: Qualitative Evaluation in a lab setting using R-50. Left: Frames from the experiment with corresponding affordances, right: the same objects from a different perspective. Colors denote grasp (green), place (blue) and pull (red).

likely places, like "placing on the bottom shelf" (faint blue intensity), are ruled out. No explicit object knowledge is required for this task and – as mentioned – no planner was used. The general setting is illustrated in Figure 1.

We show two such action sequences resulting from this setup given the same initial conditions: 1) pick-and-place a cup and take a bottle down; and 2) pull a handle and take a cup down. Note that pick-and-place and take down actions consist of a sequence of grasp, place and release actions. Results of the robot experiment are shown in Figure 12, where we show affordance maps obtained using R-50 and the key frames of the robot action execution (please see supplementary video for the full experiment). It is important to stress that the network has not been trained on this kind of the scenes. We can see that the affordances of objects (the bottle and the cup) and object parts (the door handle and the shelf) were correctly identified and action sequences successfully executed.

In order to show robustness and generalization abilities of our approach we have also generated affordance maps for five other scenes with different object configurations and different perspectives. The scenes and their corresponding affordance maps are presented in Figure 13. Again, note that the training set (same as for Figure 12) for these experiments consists of scenes, which are quite different from the ones shown here. It can be seen that most of the here-considered affordances are correctly assigned and not many errors are found. Challenging (e.g. transparent) objects are also correctly identified. The ground plane is usually not considered for a placing affordance as the system would recognize it rather as "walk-able" (coloring not shown). Surfaces, which are nearly horizontal, appear in the 2D image with only a few pixels and the system also considers them not for placing. Furthermore, some other placing-surfaces are only partially detected. But note, that these results are all based only on single 2D views. Methods for accumulating knowledge (e.g. based on a voting scheme across a sequence of images) can without problems be added to further improve on this. However, already these results demonstrate that our approach can generalize very well to different scenes with variable object configurations even though the network has not been specifically trained on such scenes.

6. Conclusion

In this paper we have described a method that labels a comparatively large set of 12 affordances pixel-wise given only single 2D RGB images. We have shown how to construct an affordance training dataset from object parts segmentation and apply recent semantic segmentation methods to learn affordances effectively. An extensive analysis on the impact of modifications to the loss, sampling and architecture has been carried out. The state-of-the-art method of Roy and Todorovic [27] is substantially outperformed by adopting features that were obtained using our method.

A strength of our method is that it is fast (less than 10ms) while operating on 2D images of arbitrary size. It is applicable on all kinds of scenes, even in presence of light-absorbing, transparent and reflecting materials where structured light cannot be used. Hence, it can easily be adapted to multiple scenarios. The transfer to a scene that had not been part of any training set was not a problem in the here-shown robotic test. When our method is to used in practical applications it can easily be modified: The set of affordances can be adopted according to the capabilities of a robot or detection thresholds of individual affordances can be modified. The field of autonomous robotics, for example considering service robots, relies heavily on semantic scene analysis methods. We think that the here-presented fast and simple 2D-affordance detection can be used to provide a machine with good estimates of what to do in a scene using few computational resources. Given a task ("clean up the room") and pairing this type of affordance analysis with planning algorithms would make such a system applicable in different domains that require autonomous action decisions.

Acknowledgment

This research was funded by the European Union H2020-ICT-2016-2017/H2020-ICT-2016-1 / 731761 (IMAGINE)

References

[1] J. J. Gibson, The Ecological Approach to Visual Perception, Houghton Mifflin, 1979.

[2] H. Grabner, J. Gall, L. Van Gool, What makes a chair a chair?, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2011.

[3] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015.

[4] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2017.

[5] M. T. Turvey, Affordances and prospective control: An outline of the ontology, Ecological psychology 4 (3) (1992) 173–187.

[6] A. Chemero, An outline of a theory of affordances, Ecological psychology 15 (2) (2003) 181–195.

[7] E. Rietveld, J. Kiverstein, A rich landscape of affordances, Ecological Psychology 26 (4) (2014) 325–352.

[8] D. G. Dotov, L. Nie, M. M. De Wit, Understanding affordances: history and contemporary development of Gibson's central concept, Avant: the Journal of the Philosophical-Interdisciplinary Vanguard .

[9] L. Lobo, M. Heras-Escribano, D. Travieso, The History and Philosophy of Ecological Psychology, Frontiers in Psychology 9 (2018) 2228, ISSN 1664-1078.

[10] W. H. Warren, Perceiving affordances: Visual guidance of stair climbing., Journal of experimental psychology: Human perception and performance 10 (5) (1984) 683.

[11] W. G. Cole, G. L. Chan, B. Vereijken, K. E. Adolph, Perceiving affordances for different motor skills, Experimental brain research 225 (3) (2013) 309–319.

[12] J. Franchak, K. Adolph, Affordances as probabilistic functions: Implications for development, perception, and decisions for action, Ecological Psychology 26 (1-2).

[13] J. B. Wagman, S. E. Caputo, T. A. Stoffregen, Hierarchical nesting of affordances in a tool use task., Journal of Experimental Psychology: Human Perception and Performance 42 (10) (2016) 1627.

[14] M. Stark, P. Lies, M. Zillich, J. Wyatt, B. Schiele, Functional object class detection based on learned affordance cues, in: International Conference on Computer Vision Systems (ICVS), Springer, 2008.

[15] Y. Zhu, A. Fathi, L. Fei-Fei, Reasoning about Object Affordances in a Knowledge Base Representation, in: European Conference on Computer Vision (ECCV), Springer, 408–424, 2014.

[16] C. Ye, Y. Yang, C. Fermüller, Y. Aloimonos, What Can I Do Around Here? Deep Functional Scene Understanding for Cognitive Robots, in: ICRA, vol. abs/1602.00032, 2017.

[17] J. Sawatzky, A. Srikantha, J. Gall, Weakly supervised affordance detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[18] J. Sawatzky, M. Garbade, J. Gall, Ex Paucis Plura: Learning Affordance Segmentation from Very Few Examples, in: German Conference on Pattern Recognition (GCPR), 2018.

[19] A. Myers, C. L. Teo, C. Fermüller, Y. Aloimonos, Affordance Detection of Tool Parts from Geometric Features, in: ICRA, 2015.

[20] A. Nguyen, D. Kanoulas, D. G. Caldwell, N. G. Tsagarakis, Object-based affordances detection with convolutional neural networks and dense conditional random fields, in: Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, IEEE, 5908–5915, 2017.

[21] A. Gupta, S. Satkin, A. A. Efros, M. Hebert, From 3D Scene Geometry to Human Workspace, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

[22] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, J. Sivic, People watching: Human actions as a cue for single view geometry, International Journal of Computer Vision (IJCV) 110 (3).

[23] H. Kjellström, J. Romero, D. Kragic, Visual object-action recognition: Inferring object affordances from human demonstration 115 (1).

[24] X. Wang, R. Girdhar, A. Gupta, Binge watching: Scaling affordance learning from sitcoms, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[25] M. Savva, A. X. Chang, P. Hanrahan, M. Fisher, M. Nießner, SceneGrok: Inferring action maps in 3D environments, ACM transactions on graphics (TOG) 33 (6).

[26] N. Rhinehart, K. M. Kitani, Learning action maps of large environments via first-person vision, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[27] A. Roy, S. Todorovic, A Multi-scale CNN for Affordance Segmentation in RGB Images, in: European Conference on Computer Vision (ECCV), Springer, 2016.

[28] T.-T. Do, A. Nguyen, I. Reid, AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection, in: International Conference on Robotics and Automation (ICRA), 2018.

[29] H. Min, C. Yi, R. Luo, J.-H. Zhu, S. Bi, Affordance Research in Developmental Robotics: A Survey, IEEE Transactions on Cognitive and Developmental Systems 8.

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

[31] K. F. Uyanik, Y. Calskan, A. K. Bozcuoglu, O. Yuruten, S. Kalkan, E. Sahin, Learning social affordances and using them for planning, in: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 35, 2013.

[32] A. Saxena, J. Driemeyer, A. Y. Ng, Robotic grasping of novel objects using vision, The International Journal of Robotics Research 27 (2) (2008) 157–173.

[33] R. Detry, D. Kraft, A. G. Buch, N. Krüger, J. Piater, Refining grasp affordance models by experience, in: Robotics and automation (icra), 2010 ieee international conference on, IEEE, 2287–2293, 2010.

[34] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, J. Piater, Learning grasp affordance densities, Paladyn, Journal of Behavioral Robotics 2 (1) (2011) 1–17.

[35] A. Ückermann, C. Elbrechter, R. Haschke, H. Ritter, 3D scene segmentation for autonomous robot grasping, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, ISSN 2153-0866, 1734–1740, 2012.

[36] H. Guo, Y. Meng, Distributed reinforcement learning for coordinate multi-robot foraging, Journal of intelligent & robotic systems 60 (3-4) (2010) 531–551.

[37] S. Griffith, V. Sukhoy, T. Wegter, A. Stoytchev, Object categorization in the sink: Learning behavior–grounded object categories with water, in: Proceedings of the 2012 ICRA Workshop on Semantic Perception, Mapping and Exploration, Citeseer, 2012.

[38] V. Tikhanoff, U. Pattacini, L. Natale, G. Metta, Exploring affordances and tool use on the iCub, in: Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on, IEEE, 130–137, 2013.

[39] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, R. Dillmann, Object–Action Complexes: Grounded abstractions of sensory–motor processes, Robotics and Autonomous Systems 59 (10) (2011) 740 – 757, ISSN 0921-8890, doi:https://doi.org/10.1016/j.robot.2011.05.009, URL http://www.sciencedirect.com/science/article/pii/S0921889011000997.

[40] O. Yürüten, E. Şahin, S. Kalkan, The learning of adjectives and nouns from affordance and appearance features, Adaptive Behavior 21 (6) (2013) 437–451.

[41] A. Stoytchev, Behavior-grounded representation of tool affordances, in: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, IEEE, 3060–3065, 2005.

[42] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, L. De Raedt, Learning relational affordance models for robots in multi-object manipulation tasks, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 4373–4378, 2012.

[43] E. Ugur, J. Piater, Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, 2627–2633, 2015.

[44] S. Fichtl, D. Kraft, N. Krüger, F. Guerin, Bootstrapping Relational Affordances of Object Pairs Using Transfer, IEEE Transactions on Cognitive and Developmental Systems 10 (1) (2018) 56–71, ISSN 2379-8920, doi:10.1109/TCDS.2016.2616496.

[45] E. Uğur, E. Şahin, Traversability: A case study for learning and perceiving affordances in robots, Adaptive Behavior 18 (3-4) (2010) 258–284.

[46] E. Ugur, M. R. Dogar, M. Cakmak, E. Sahin, The learning and use of traversability affordance using range images on a mobile robot, in: Robotics and Automation, 2007 IEEE International Conference on, IEEE,

1721–1726, 2007.

[47] J. Sun, J. L. Moore, A. Bobick, J. M. Rehg, Learning visual object categories for robot affordance prediction, The International Journal of Robotics Research 29 (2-3) (2010) 174–197.

[48] M. A. Lewis, H.-K. Lee, A. Patla, Foot placement selection using non-geometric visual properties, The International Journal of Robotics Research 24 (7) (2005) 553–561.

[49] C. Barck-Holst, M. Ralph, F. Holmar, D. Kragic, Learning grasping affordance using probabilistic and ontological approaches, in: Advanced Robotics, 2009. ICAR 2009. International Conference on, IEEE, 1–6, 2009.

[50] A. Bierbaum, M. Rambow, T. Asfour, R. Dillmann, Grasp affordances from multi-fingered tactile exploration using dynamic potential fields, in: Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on, IEEE, 168–174, 2009.

[51] J. Baleia, P. Santana, J. Barata, On exploiting haptic cues for self-supervised learning of depth-based robot navigation affordances, Journal of Intelligent & Robotic Systems 80 (3-4) (2015) 455–474.

[52] J. T. Carvalho, S. Nolfi, Behavioural plasticity in evolving robots, Theory in Biosciences 135 (4) (2016) 201–216.

[53] D. Song, N. Kyriazis, I. Oikonomidis, C. Papazov, A. A. Argyros, D. Burschka, D. Kragic, Predicting human intention in visual observations of hand/object interactions, in: ICRA, 2013.

[54] D. Song, C. H. Ek, K. Huebner, D. Kragic, Task-based robot grasp planning using probabilistic inference, IEEE transactions on robotics 31 (3) (2015) 546–561.

[55] A. Aldoma, F. Tombari, M. Vincze, Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 1732–1739, 2012.

[56] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al., Shapenet: An information-rich 3d model repository, arXiv preprint arXiv:1512.03012 .

[57] C. Chen, A. Seff, A. L. Kornhauser, J. Xiao, DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving, 2015 IEEE International Conference on Computer Vision (ICCV) .

[58] O. Kroemer, J. Peters, A flexible hybrid framework for modeling complex manipulation tasks., in: ICRA, 1856–1861, 2011.

[59] P. Kaiser, D. I. Gonzalez-Aguirre, F. Schultje, J. B. Sol, N. Vahrenkamp, T. Asfour, Extracting whole-body affordances from multimodal exploration, 2014.

[60] P. Kaiser, M. Grotz, E. E. Aksoy, M. Do, N. Vahrenkamp, T. Asfour, Validation of whole-body loco-manipulation affordances for pushability and liftability, in: Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, IEEE, 920–927, 2015.

[61] B. Akgun, N. Dag, T. Bilal, I. Atil, E. Sahin, Unsupervised learning of affordance relations on a humanoid robot, in: Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on, IEEE, 254–259, 2009.

[62] W. P. Chan, Y. Kakiuchi, K. Okada, M. Inaba, Determining proper grasp configurations for handovers through observation of object movement patterns and inter-object interactions during usage, in: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, IEEE, 1355–1360, 2014.

[63] D. I. Kim, G. S. Sukhatme, Semantic labeling of 3d point clouds with object affordance for robot manipulation, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, Citeseer, 5578–5584, 2014.

[64] A. Stoytchev, Robot tool behavior: A developmental approach to autonomous tool use, Ph.D. thesis, Georgia Institute of Technology, 2007.

[65] C. Wang, K. V. Hindriks, R. Babuska, Robot learning and use of affordances in goal-directed tasks, in: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, 2288–2294, 2013.

[66] F. Cruz, S. Magg, C. Weber, S. Wermter, Training agents with interactive reinforcement learning and contextual affordances, IEEE Transactions on Cognitive and Developmental Systems 8 (4) (2016) 271–284.

[67] C. Castellini, T. Tommasi, N. Noceti, F. Odone, B. Caputo, Using object affordances to improve object recognition, IEEE Transactions on Autonomous Mental Development 3 (3) (2011) 207–215.

[68] E. Ugur, J. Piater, Emergent structuring of interdependent affordance learning tasks, in: Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on, IEEE, 489–494, 2014.

[69] E. Ugur, E. Oztop, E. Sahin, Goal emulation and planning in perceptual space using learned affordances, Robotics and Autonomous Systems 59 (7-8) (2011) 580–595.

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

[71] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, T. Brox, What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?, International Journal of Computer Vision 126 (9) (2018) 942–960, ISSN 1573-1405.

[72] F. S. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, J. M. Alvarez, Effective Use of Synthetic Data for Urban Scene Semantic Segmentation, The European Conference on Computer Vision (ECCV) .

[73] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, A. Torralba, Semantic understanding of scenes through the ade20k dataset, arXiv preprint arXiv:1608.05442 .

[74] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. Zitnick, Microsoft COCO: Common Objects in Context, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), European Conference on Computer Vision (ECCV), vol. 8693, Springer International Publishing, ISBN 978-3-319-10601-4, 2014.

[75] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[76] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, IEEE Transactions on Pattern Analysis and Machine Intelligence 40.

[77] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[78] P. O. Pinheiro, T.-Y. Lin, R. Collobert, P. Dollar, Learning to Refine Object Segments, in: European Conference on Computer Vision (ECCV), 2016.

[79] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009, IEEE, 248–255, 2009.

[80] T. Lüddecke, F. Wörgötter, Learning to Segment Affordances, in: International Computer Vision Conference Workshops (ICCVW), 2017.

[81] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, in: NIPS Workshops, 2017.

[82] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning 4 (2).

[83] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, S. Kamali, M. Malloci, J. Pont-Tuset, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, K. Murphy, OpenImages: A public dataset for large-scale multi-label and multi-class image classification., Dataset available from https://storage.googleapis.com/openimages/web/index.html .

[84] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from RGBD images, in: European Conference on Computer Vision, Springer, 746–760, 2012.

[85] KUKA AG, LWR, URL `https://www.kuka.com/en-de/products/`, accessed 08.04.2019.

[86] Schunk, SDH Hand, URL `https://schunk.com/pl_en/gripping-systems/ser` accessed 08.04.2019.

[87] M. J. Aein, E. E. Aksoy, M. Tamosiunaite, J. Papon, A. Ude, F. Wörgötter, Toward a library of manipulation actions based on semantic object-action relations, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 4555–4562, doi:10.1109/IROS.2013.6697011, 2013.

[88] M. J. Aein, E. E. Aksoy, F. Wörgötter, Library of Actions: Implementing a Generic Robot Execution Framework by Using Manipulation Action Semantics, IJRR (in press) .

[89] A. J. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), vol. 2, 1398–1403, doi:10.1109/ROBOT.2002.1014739, 2002.

[90] T. Kulvicius, K. Ning, M. Tamosiunaite, F. Worgötter, Joining Movement Sequences: Modified Dynamic Movement Primitives for Robotics Applications Exemplified on Handwriting, IEEE Transactions on Robotics 28 (1) (2012) 145–157, doi:10.1109/TRO.2011.2163863.

[91] J. Papon, M. Schoeler, Semantic pose using deep networks trained on synthetic RGB-D, in: Proceedings of the IEEE International Conference on Computer Vision, 774–782, 2015.

[92] A. Collet, M. Martinez, S. S. Srinivasa, The MOPED framework: Object recognition and pose estimation for manipulation, The International Journal of Robotics Research 30 (10) (2011) 1284–1306.