Part-driven visual perception of 3D objects

Frithjof Gressmann, Timo Lüddecke, Tatyana Ivanovska, Markus Schoeler, Florentin Wörgötter
Georg-August-University, Göttingen, Germany
tiva@phys.uni-goettingen.de

Keywords: Object recognition, part perception, deep learning, neural networks, 3D objects

Abstract: During the last years, approaches based on convolutional neural networks (CNN) had substantial success in visual object perception. CNNs turned out to be capable of extracting high-level features of objects, which allow for fine-grained classification. However, some object classes exhibit tremendous variance with respect to their instances appearance. We believe that considering object parts as an intermediate representation could be helpful in these cases. In this work, a part-driven perception of everyday objects with a rotation estimation is implemented using deep convolution neural networks. The used network is trained and tested on artificially generated RGB-D data. The approach has a potential to be used for part recognition of realistic sensor recordings in present robot systems.

1 Introduction

The latest wave of artificial neural network methods which was triggered by Krizhevsky et al. (2012) has led to impressive progress on many computer vision problems. However, CNNs can not compete with humans in terms of generalization. This can be exemplified by looking at objects with a greatly varying appearance, e.g. power-drills. They come in vastly different forms and shapes (hand-held, on a stand, etc.) and when comparing two drill objects as a whole, many times there are only very few common visual features across instances which could support successful whole-object recognition with CNNs. The fact that humans can refer to these objects as drills, however, can be explained by looking at the object parts: On a part level the two drill types share comparable features, e.g. the boring bit and the power-switch and some others. This view suggests that a visual scene understanding and the class recognition in particular can be strongly guided by parts.

When it comes to handling and usage of objects, an adequate understanding of the main functional object parts seems to be indispensable. Particularly knowledge about which objects parts are suitable for grasping and manipulating forms are an important basis of the everyday life activities that robots are aiming for. In this regard, the recent achievements of convolutional neural networks suggest exploring the potential of an object perception on a per-part basis.

To the best of our knowledge, so far, object recognition with neural networks has been mainly focused on whole object, approaches explicitly addressing parts are rather uncommon, in particular in conjunction with 3D data, like depth images and normals maps as input.

2 Related Work

The idea of considering objects as composite structures involving many parts is not new to computer vision.

Perhaps most well known, Biederman (1987) suggested the recognition by components (RBC) theory. According to Biederman the human object recognition is carried out by separating objects into its main components which he calls geons (Biederman, 1987). An advantage of this approach is its economy since a small number of basic parts can form a huge number of possible objects. Also, part recognition is viewpoint-invariant to some degree since the properties of object parts do not change massively under different viewing angles. However, RBC provides a theoretical framework rather than an implementation, whereas this work presents a concrete algorithm. It shares with RBC the consideration of parts but our parts are neither geons nor is the number of part classes limited to 36. The segmentation of objects into parts is addressed by Schoeler et al. (2015). They propose a bottom-up algorithm to dissect objects into parts based on local convexity which does not require

annotated training data.

In the 2D domain, the implicit shape model (Leibe et al., 2004) employs parts in terms of codebook entries to determine the location of the central object by a voting procedure akin to Hough transform. Felzenszwalb et al. (2010) represents parts as linear filters which are convolved over the image to form feature maps which then are being employed by a deformable part model to predict the object location. Approaches involving neural networks, which explicitly focus on parts are rather rare, though. Zhang et al. (2014) proposes a part-based algorithm for fine-grained category detection based on the R-CNN by Girshick et al. (2014) which enriches CNNs with the capability of segmentation. Bottom-up region proposals of 2D images are assessed to be either parts or whole objects. Further approaches in the 2D domain which make use of dense prediction involve Tsogkas et al. (2015) and Oliveira et al. (2016). Our approach differs from them as it encompasses 3D information in form of depth images and normal maps.

The approaches by Gupta et al. (2015) and Papon and Schoeler (2015) use CNNs to predict poses of objects in RGB-D scenes. In contrast to our work, both operate on the object rather than the part level.

3 Materials and Methods

3.1 Data Generation

In contrast to whole-object training data, there are no annotated real world data sets of sufficient size to train deep networks with ground truth for object parts being available today. Nonetheless, there are different smaller data sets for the development and the valuation of object segmentation algorithms. While part level annotation of large real data sets is not feasible, we decide to use segmented models of a small data set to generate large annotated data sets suitable for deep learning. In this context, the shape COSEG dataset (COSEG) [1] provided by Wang et al. (2012) is the most eligible one because it is segmented in not too fine parts, which mostly have some functional meaning. Alternative data sets like the Princeton Benchmark for 3D Mesh Segmentation provided by Chen et al. (2009) not so useful because of their finer segmentation in many quite small parts. The COSEG dataset provides 190 everyday object instances separated into eight classes: *Chairs, Lamps, Candles, Guitars, Goblets, Vases, Irons, and Four-legged animals*.

[1] available here:
http://irc.cs.sdu.edu.cn/ yunhai/public_html/ssl/ssd.htm

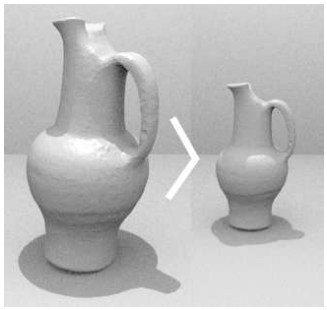


Figure 1: Example of vase object pose alignment and size normalization.

The COSEG models are neither aligned to a reference pose nor normalized to a common scale. Thus, to accomplish that different models with the same rotation or scale are represented in a comparable way, their minimum bounding boxes (MBB) and normalized size and rotation is determined. We also ensure that one characteristic part of the object will always be on the right hand side of the scene to avoid a misalignment by 180 degree. An example of the performed normalization is illustrated in Figure 1.

The creation of the annotated training data set is realized by building on the rendering techniques presented by Papon and Schoeler (2015): To convert the un-textured normalized models into RGB data, the Blender Graphic Engine is employed. Depth information is produced using the BlenSor sensor simulation toolbox[2] by Gschwandtner et al. (2011), which allows reproducing the noisy depth measurement of a Microsoft Kinect RGB-D sensor in a realistic way. Despite the fact that due to lack of texture the RGB data misses hue information, the generated RGB-D are close to realistic recordings of a Kinect Camera used by many robot systems. To obtain a part mask the corresponding object model is segmented into parts using the COSEG ground truth information. Subsequently, each of the part objects were pulled into the Blender scene and aligned with the object. Using the Blender coordination transformations it is possible to calculate the part mask entries. Finally, in order to possibly enrich the scene information we decide to use structured RGB-D data to compute surface normals using the technique presented by Holzer et al. (2012).

3.2 Network Structure

As the approach to recognize object parts with neural networks is rather new, there are no proven standard network architectures to use in this case. However, it seems reasonable to view the part recognition prob-

[2] available here: http://www.blensor.org/

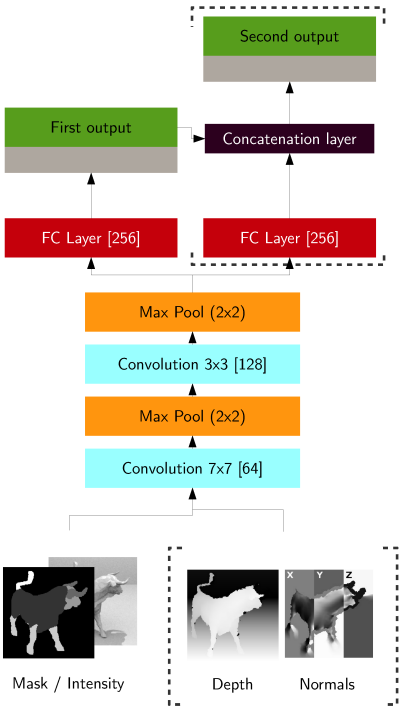


Figure 2: Basic network architecture for the various prediction tasks: The numbers in brackets denote the number of filters of the convolution layers, the number of nodes of the fully connected layers (FC layers) or the pooling size of the pooling stages. Moreover the convolutional modules are labeled with the used filter size. The network input consisted of a 96×96 part mask stacked together with the intensity image and optionally depth and surface normal information. For the multi-output models the first network output is merged into the second output branch. Dotted brackets indicate optional components of the network that are added depending on the task.

lem as an object recognition problem, where small objects (the parts) in close proximity to a larger object (the object as a whole) need to be recognized. Following this line of thought it makes sense to be inspired by standard architectures for object perception and to test part recognition performance of different versions of them.

As basis for each of the perception networks the *ReLU-Conv-Pool* network modules suggested by Krizhevsky et al. (2012) turn out to be a good choice. The module consists of a 2D-convolutional layer followed by a 2D-Maximum-Pooling layer where both use the recommended ReLU nonlinearity. Following the approach of Papon and Schoeler (2015) two of these Conv-Pool modules were stacked with the parameters of their most successful model. To prevent from overfitting all layers were adjusted to use the

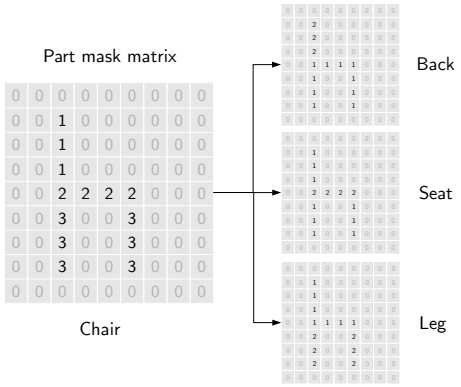


Figure 3: Illustration of the part ground truth encoding: Part information is encoded into a matrix with the dimensions of the input image, where each entry contains the part number of the corresponding image pixel. Consequently, different object parts are highlighted by different numbers.

dropout technique proposed by Hinton et al. (2012). An overview of the resulting architecture is depicted in Figure 2.

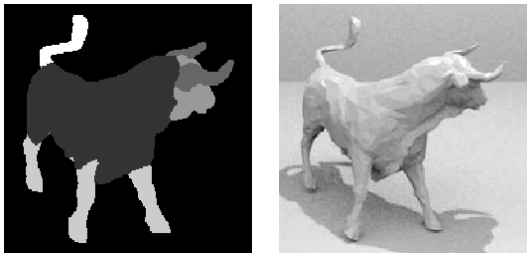
3.2.1 Network Inputs and Preprocessing

The input of the networks are 96×96 real-valued images including the intensity information and, optionally, depth and surface normal vectors in x-, y- and z-direction. To ensure standardized inputs all channels were zero-centered by subtracting the mean across every individual feature in the data. Moreover, in order to level the different scales and units the channels were normalized so that the minimum and maximum along the dimension is -1 and 1 respectively.

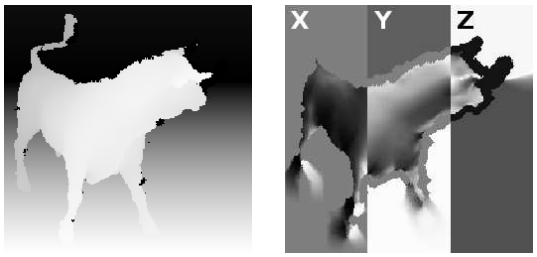
We assume that the object's parts had been segmented previously and focus our approach on classifying these segmented parts. The part ground truth is encoded into a matrix having the same size as the RGB-D image with each entry containing the part number of the corresponding image pixel. Hereafter it is referred to as part mask. As illustrated in Figure 3, the matrix encodes the parent object with a 1, the part in question with a 2 and the background with a 0 digit. The network is then requested to predict the correct label of the part or the area which is designated with 2. This way, by subsequently masking every unclassified part with 2, all object parts can be assigned a label. An overview of all inputs is presented in Figure 4.

3.2.2 Network Outputs

The network is trained and evaluated in different settings, each predicting different quantities. First, there



(a) Part mask. (b) Intensity Image.



(c) Depth Image. (d) Surface normals.

Figure 4: Compilation of the different network input types, which were generated for the training. The part and intensity information is obtained using the Blender graphic engine. The depth channel is generated with the help of the BlenSor sensor simulation toolbox by Gschwandtner et al. (2011). Finally, the object surface normals were computed using the technique presented by Holzer et al. (2012).

is a standard output to predict the eight classes of the parent objects. Second, there are two more output modules for part classification: One 'full-part output', which predicts each of the 28 parts in the eight parent object classes; and alternatively, a 'reduced-part output', which condenses semantically similar parts from the parent object classes resulting in only 16 distinct part predictions, e.g. *LampesStand* and *VasesStand* become *Stand*. In the latter case the output therefore predicts parts regardless of the actual class of the parent object. Finally, there is an output layer to predict the rotation angle of the parent object.

Rotation output In principle, rotation output can be realized in two different ways. The obvious possibility is the use of a single output neuron, the activation value of which represents the predicted rotation angular. Hereafter this solution is called 'regression model'. On the other hand it is possible to cast the rotation estimation into a classification task. To achieve this the possible value range is divided into a number of small value ranges which are being associated with a class name. The network is then requested to predict the correct range. A disadvantage of this approach is that the network cannot distinguish rotational values which lie in the same value range. The rotation estimation by class is hereafter referred to as 'binned model'. In contrast to the aforementioned class outputs, where the predictions were simply compared to the correct solution, for the regression output it is necessary to define a proper error metric, e.g. the difference between predicted and correct value. Evidently, the maximum possible error in rotation estimation is 180 degree. To reflect this, the error $\varepsilon_{\triangleleft}$ between the predicted angular α and the actual rotation β is defined as follows:

$$\varepsilon_{\triangleleft}(\alpha,\beta) = \min(|\alpha-\beta|, 360-|\alpha-\beta|) \qquad (1)$$

However, it is important to notice that the use of this error metric might cause problems in the learning process. The parameter adjustments of the output neuron depend on the gradient ∇C of the cost function. In fact, the calculation therefore includes a derivation with respect to the neuron parameters Θ. Particularly, if the angular error $\varepsilon_{\triangleleft}$ is used in the cost function, the neuron output $\alpha = \sigma(x,\Theta)$ with the activation function σ has to be differentiated:

$$\nabla C \propto \nabla\varepsilon_{\triangleleft}(\alpha,\beta,\Theta) \propto \pm\frac{(\sigma-\beta)}{|\sigma-\beta|}\cdot\sigma'$$

The gradient expression then contains the derivation of the activation function $\sigma'(w,b)$. It is therefore essential to remember that the activation function is a rectified linear unit. As a result, σ' vanishes if its argument $\vec{w}^T\vec{x}+b$ is smaller than zero. In consequence, if the weights and the bias adapt to improper values, ∇C vanishes as well and the learning process stops.

To use the regression model it had therefore been ensured that $\vec{w}^T\vec{x}+b$ is always positive. A simple way to avoid a vanishing gradient was to initialize the weights and the bias with $w_i = 0$ and $b = 90$. This way the training always sets off from $\vec{w}^T\vec{x}+b = 90 > 0$, converging safely, even if $\vec{x}$ was assigned with an improper value at the beginning of training.

3.3 Training and Testing Methodology

To find optimal learning rates for each training run, six learning rates between 0.1 and 1×10^{-6} were tested. Moreover, the learning rate was automatically reduced if the validation performance stopped improving (*adaptive learning rate*). To avoid overfitting training was canceled using the *early stopping* technique on the validation set.

The separation into training, validation, and test sets were done as follows: randomly chosen 20% of the models of each class were excluded from training. From these models, the test data set with 1000 items was generated using rotation and scaling. Hence, the

network encounter completely new object instances in the test. With the 80% remaining object models the training and validation data sets were generated in proportion 95% and 5%, respectively.

Each of the generated training examples consists of a randomly chosen object model placed in the center of the scene with random illumination to simulate shadow effects. In the first instance the objects were put to the ground and rotated around the z-axis to guarantee realistic poses. As the number of instances in each object class differs, an approximately equal number of scenes for each class has been generated. This led obviously to more frequent occurrences of object instances of smaller classes throughout the training. In total, 25 000 scenes were generated as training set, in which every single object instance occurred at least in 100 different representations.

4 Results and Discussion

The main focus is here on training and evaluation of the synthetic data sets in order to achieve a basic understanding of how well part perception with such a neural network works.

4.1 Classification

The quantitative results in the following section are obtained using the *test data set* containing 1000 scenes with *unseen* object models, i.e. with object shapes not seen during training at all.

4.1.1 Classification (whole object)

For the first step and to establish a baseline, the convolutional network is trained to predict the eight different whole-object classes based on the intensity channel only (first network output only). With an overall accuracy of 85.3%, the classification proves to be reliable, but it is worth noting that the performance significantly correlates with the number of class instances in the data set. While the perception accuracy of the 44 *Guitar* instances is close to perfection (0.97 F1 score) the classification is less reliable for the *Goblets* which only counts 12 instances (0.64 F1 score).

4.1.2 Rotation (whole object)

As a next step the model is extended with the second output to predict the rotation of the whole object around the z-axis. To measure accuracy for each output the symmetric angular error $\varepsilon_{\sphericalangle}$ is calculated as defined by Equation 1. The Area Under Curve (AUC) accuracy is calculated as

$$\varepsilon_{AUC} = 1 - \frac{1}{N}\sum_{i}^{N}\frac{\varepsilon^{i}_{\sphericalangle}}{180^{\circ}}.$$

Using this error metric we find the binned rotation model being able to predict the z-rotation with an accuracy of 69.5% which clearly outperforms the rotation regression model by 7.7%. In order to examine the performance in more detail the representation by Gupta et al. (2015) is used: It determines the fraction of objects for which the model is able to predict the rotation with an error less than ε_{ϑ} plotting it as a function of ε_{ϑ}. Figure 5a documents the comparison of the binned and the regression model using this representation.

Figure 5a shows that the binned model outperforms the regression model: For the binned model 36.8% of all predictions lie under 10° error whereas this is true for only 11.4% of the regression predictions. Moreover it is obvious that for a significant fraction of predictions the angular error is greater than 100° and therefore not usable in a real world application.

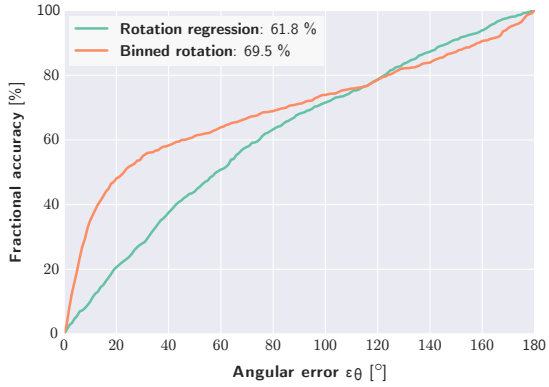
Examining the rotational estimation for each object class, we find that for both models the accuracy rates are not homogeneous. In fact, as presented in Figure 5b, rates vary from 94.7% in *Guitars* class to 50.9% in the *Goblets* class with 83.8% to 5.8% under 10° error, respectively. The scores correlate with the number of different object instances within classes. The per-class analysis also discloses that for the three classes *Guitars*, *Four-legged* and *Chairs*, the rotation estimation is reliable in the sense that more than 75% of all predictions have an error less than 25°.

Motivated by these results all of the following simulations (if not stated otherwise) are performed using the binned rotational estimation.

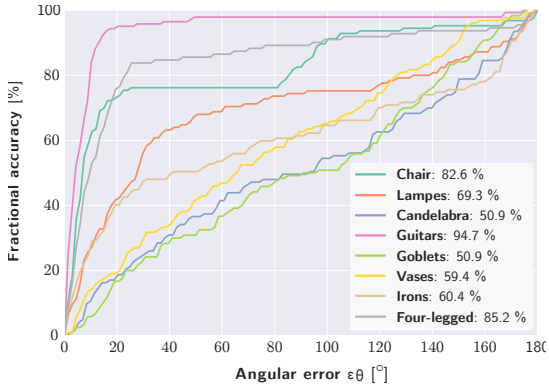
4.1.3 Part recognition

First, the network is trained to predict the full part label, which contains the parent object's class name, e.g. *VasesContainer* contains *Vases*. For this only the two-dimensional part masks are used as input. This method achieved an overall classification accuracy of 82.3%. The corresponding confusion matrix is documented in Figure 6.

The performance for different parts varies in a noteworthy manner ranging from 53% for *LampesLamp* up to even 100% for *GuitarsBody*. Notably the network tended to confuse *Goblet* parts with *Lamp* parts and had problems keeping the parts of *Vases* - *Handle*, *Stand* - apart. In particular, in half of all



(a) Rotation estimation comparison of binned and regression model: The binned rotation model outperforms the rotation model in terms of overall accuracy as well as share of marginal errors.



(b) Comparison of the rotation estimation performances broken down by class: Evidently, a large variation in accuracy is observed ranging from 94.7% for the guitars class to 50.9% for the goblets class.

Figure 5: Results of the rotation estimation: Following Gupta et al. (2015), the depiction represents the fraction of objects for which the model is able to predict the rotation with an error of less than ε_ϑ, plotted versus ε_ϑ.

classes an accuracy of over 90% is achieved, while the *Goblet* part classification falls under 50%.

One remarkable observation is that the network confuses *GobletsStand* with *LampesStand* and also *GobletsHandle* with *LampesHolder* in roughly one half of the predictions performed on these classes. These findings suggest that the network recognized semantic part features, e.g. *Stands* and *Handles*, rather than the class of the parent object and its parts. Therefore it seems reasonable to train a network for these particular part classes. As *Stands* and *Handles* are not part in all of the eight parent object classes, this part-focused classification consequently reduces the number of classes, hence it is called reduced part classification. Experiments indicate that employing such a reduced classification model turned out to be a reasonable choice. The overall accuracy of the reduced part perception lies at 85.4% and, therefore, on the same scale as the full class part recognition. The rates ranged from 62% for the *Holder* class to 92% for the *Seat* class. Remarkably, like the full model, the reduced one confuses the *Handle* with *Holder*, with both part classes having the same function. It is not possible to compare the scores of the full and reduced models in a direct manner since the 16-reduced-classes identification task is easier than that for the 28-full-classes problem. Nevertheless, the good accuracy of the reduced model supports the presumption that the network is able to identify parts regardless of the parent object class.

4.1.4 Part-driven object classification

Provided with the correctly classified parts, it might be possible to derive the class name of the parent object. Next, we investigate if this procedure leads to a better performance compared to directly predicting the object class. To realize such a part-driven object classification, the object in question has to be segmented into parts, which then themselves are classified. From the resulting part labels the parent object class is inferred. For example, if the part classification yields *ChairSeat*, *LampesStand* and *ChairBack* it is likely that the parent object is a *Chair*. This technique is implemented for all models with part recognition which did not provide a class output natively. For the part recognition model the mask input was used as discussed in the previous section. This leads to a classification accuracy of 88.2%. Figure 8 shows the corresponding confusion matrix. Compared to the original, direct classification model, which uses intensity information and is only and explicitly trained for object classification, this is an improvement of 2.9%. The fact that objects can be recognized just from their parts using CNN could be considered an important observation.

5 Conclusions and Future Work

The presented work proves that it is possible to realize part perception of everyday objects with synthetically trained deep neural networks. Two aspects are potentially of more far reaching interest. 1) We showed that parts are recognized independently of their parent object class. This kind of generalization is a typical human trait. For us a leg remains a leg whether it comes from a chair or a table. 2) We could also demonstrate that object recognition can

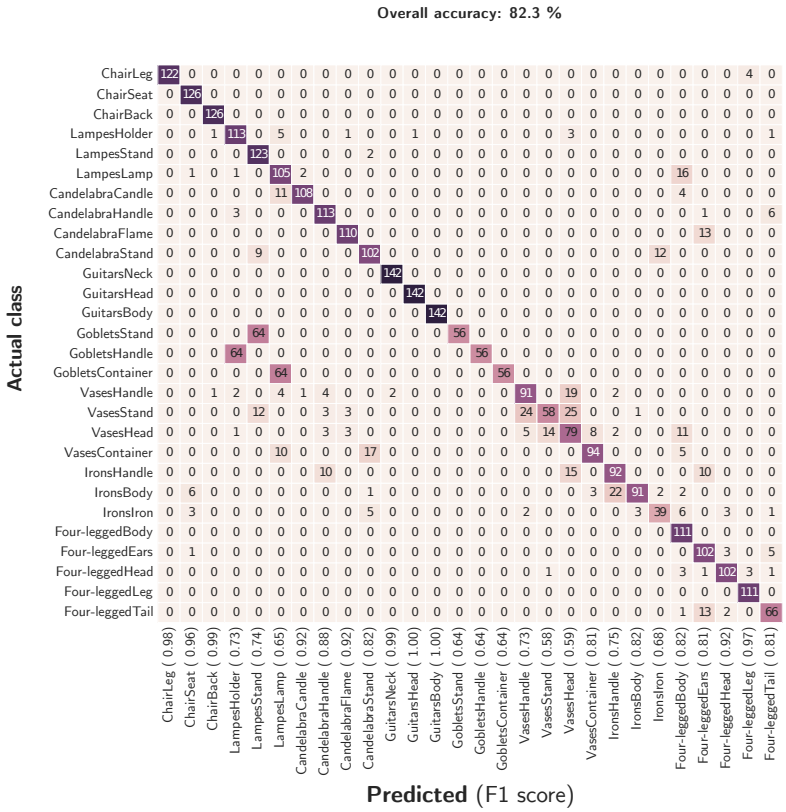


Figure 6: Confusion matrix of the full part prediction model, which is trained on the part mask channel. The network is able to predict parts without major problems but tends to confuse both vases' and goblets' components.

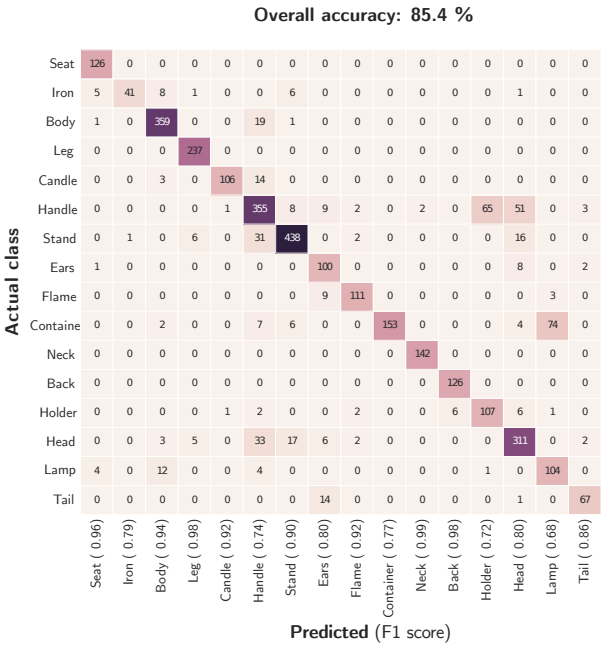


Figure 7: Confusion matrix of the reduced part classification carried out by a network trained on the part mask input channel only. Problems occurred only in the prediction of handle and container parts.

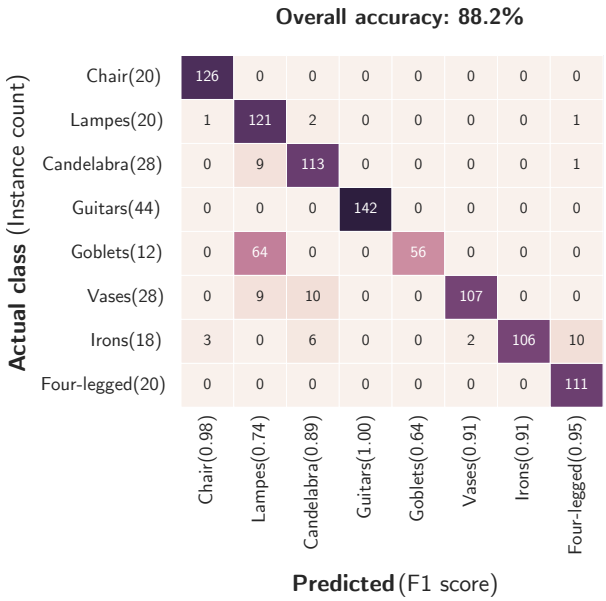


Figure 8: Confusion matrix of the COSEG object classification, which was derived from the recognized part. The method tends to confuse lampes with goblets but outperforms the standard classification model, which was trained to recognize the objects as-a-whole.

be achieved from recognizing part combinations from which the object class is then inferred. Here, we found that for our system part-driven object classification, where the parent object class is derived from the object parts, outperformed a comparable standard object classification network which does not use object parts as an intermediate representation.

In our future work, we plan to test the part-based perception approach on a bigger data set also using more advanced network structures as well as extend the approach to infer not only objects, but also their functions (affordances). The method performance will be compared to the existing techniques and validated on real RGB-D data. In order to make predictions more robust, more rotation axes and view angles could be included in the training. Additionally, artificial noise and visual obstacles could be applied on the training data to increase robustness even further.

REFERENCES

Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115.

Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3).

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9).

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. IEEE.

Gschwandtner, M., Kwitt, R., Uhl, A., and Pree, W. (2011). Blensor: Blender sensor simulation toolbox. In *Advances in Visual Computing*, volume 6939 of *Lecture Notes in Computer Science*, chapter 20. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Gupta, S., Arbeláez, P., Girshick, R., and Malik, J. (2015). Aligning 3d models to rgb-d images of cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Holzer, S., Rusu, R. B., Dixon, M., Gedikli, S., and Navab, N. (2012). Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2684–2689. IEEE.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Leibe, B., Leonardis, A., and Schiele, B. (2004). Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2.

Oliveira, G. L., Valada, A., Bollen, C., Burgard, W., and Brox, T. (2016). Deep learning for human part discovery in images. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Papon, J. and Schoeler, M. (2015). Semantic pose using deep networks trained on synthetic rgb-d. In *IEEE International Conference on Computer Vision (ICCV)*.

Schoeler, M., Papon, J., and Worgotter, F. (2015). Constrained planar cuts - object partitioning for point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tsogkas, S., Kokkinos, I., Papandreou, G., and Vedaldi, A. (2015). Semantic part segmentation with deep learning. *arXiv preprint arXiv:1505.02438*.

Wang, Y., Asafi, S., van Kaick, O., Zhang, H., Cohen-Or, D., and Chen, B. (2012). Active co-analysis of a set of shapes. 31(6):165:1–165:10.

Zhang, N., Donahue, J., Girshick, R., and Darrell, T. (2014). Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision*. Springer.