Grasping of Unknown Objects using Deep Convolutional Neural Networks based on Depth Images

Philipp Schmidt, Nikolaus Vahrenkamp, Mirko Wächter and Tamim Asfour

***Abstract*— We present a data-driven, bottom-up, deep learning approach to robotic grasping of unknown objects using Deep Convolutional Neural Networks (DCNNs). The approach uses depth images of the scene as its sole input for synthesis of a single-grasp solution during execution, adequately portraying the robot's visual perception during exploration of a scene. The training input consists of precomputed high-quality grasps, generated by analytical grasp planners, accompanied with rendered depth images of the training objects. In contrast to previous work on applying deep learning techniques to robotic grasping, our approach is able to handle full end-effector poses and therefore approach directions other than the view direction of the camera. Furthermore, the approach is not limited to a certain grasping setup (e. g. parallel jaw gripper) by design. We evaluate the method regarding its force-closure performance in simulation using the KIT and YCB object model datasets as well as a big data grasping database. We demonstrate the performance of our approach in qualitative grasping experiments on the humanoid robot ARMAR-III.**

I. INTRODUCTION

Grasping an object is an everyday task, which humans and some animals perform subconsciously with both ease and reliability. By watching adults and by gathering own experiences, children rapidly learn how to grasp objects without the need to develop complex calculation models, solve complicated equations or remember every object encountered so far by heart. With robots becoming progressively more intelligent in interacting with their environment, the need for a robust solution for grasping everyday objects is of utmost importance. Nevertheless, robotic grasping still provides many challenges for researchers and is still among the most demanding problems in modern robotics. Finding a general solution would open up many new possibilities for robots to autonomously explore their environment and would enable them to perform better at assisting humans.

Common approaches for grasping objects with a robot hand or gripper are to calculate a list of good grasps offline in advance, based on full geometric information and the physical model of the object in question, see e. g. [1], [2], [3], [4], [5]. Those grasps are then stored in a database alongside a representation of the object itself. Hence, the main challenge in object grasping is reduced to object recognition and pose estimation during online execution. The need for a full description of the geometry of an object for grasp generation makes this approach unsuitable for exploration of a previously unknown scenes through image and depth sensors only, as they usually provide only partial and noisy information. More advanced and state-of-the-art data-driven methods are able to work on partial data [6], or even use a form of shape completion to handle partial views, while still handling novel objects [7].

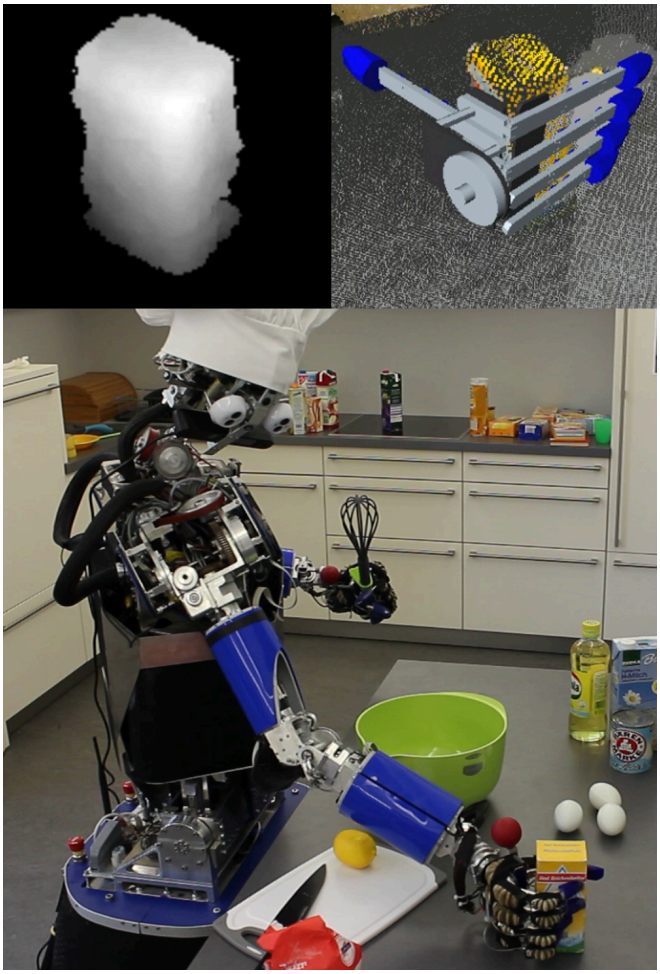

Fig. 1. Based on a segmented depth image of the robot's environment (top-left), a grasp for a previously unknown object is generated (top-right) using a Deep Convolutional Neural Network (DCNN). Based on this information, the humanoid robot ARMAR-III executes the grasp (bottom).

Recognising the need for more research regarding this topic, this work provides a bottom-up, data-driven, deep-learning approach for unknown objects utilising DCNNs. The pipeline uses depth-image sensor data as its only source of input, which adequately portraits the vision of a robot

The research leading to these results has received funding from the European Unions Horizon 2020 Research and Innovation programme under grant agreement No 643950 (SecondHands).

The authors are with the High Performance Humanoids Technologies (H^2T) Lab, Institute for Anthropomatics and Robotics (IAR), Karlsruhe Institute of Technology (KIT), Germany

exploring a scene, and therefore does not need the full information about the geometry and physical model of objects. By training this DCNN with examples of good grasps, in combination with the associated depth-images of the scene, this technique is able to directly generate grasping actions from depth information in one single step. The grasp examples are generated by classic, state-of-the-art grasp planners, while using simulated sensor data fulfils the need for large amounts of training data for deep-learning. In contrast to many classic grasp planners or grasping pipelines, this approach is online-capable and is intended to be used as an on-the-fly grasp solver rather than an offline grasp planner.

Deep-learning techniques have already shown the ability to outperform classic approaches, without the necessity to invest a lot of effort into modeling the system and finding computational and algorithmic solutions [8]. One of the most problematic requirements of deep learning lies in the large amount of data needed to perform training. Harvesting this data from big existing databases or datasets (e.g. Cornell Grasping Dataset [9], [10]) can be time consuming and might not be sufficient for the problem. Generating training data in simulation becomes increasingly interesting for complex problems, when only expensive hand-labeled data could be used otherwise [11]. In image recognition and classification tasks, DCNNs have shown up to superhuman performance (e.g. [12], [13]) in the ImageNet challenge [14]. Those neural networks utilise classic image features in combination with deep neural networks. As such, they are capable of "understanding" the structure of an object in an image and also the spatial arrangement of geometric features. Classic grasp planners have shown to be able to use this grasp related local geometric information to improve the quality of the calculated grasp [6]. DCNNs therefore have promising preconditions for the use in grasping tasks.

In this work we show that deep learning techniques are capable of transferring and utilising grasp knowledge, gained in a training process, to different unknown objects only based on a depth image of the object and the analysis of its geometric features.

II. RELATED WORK

The survey paper [4] by Bohg et al. provides a set of criteria to distinguish between the various approaches of robotic grasping, as well as their capabilities and methodology. The authors divide all methods into *analytic* and *data-driven* approaches. Analytical methods most commonly assume full knowledge about geometry and physics of the object, but a robot exploring its environment relies on sensory input, which generally provides only partial and noisy information. Simple analytical approaches are therefore commonly used to generate grasps prior to execution, establishing a *Prior Object Knowledge*. Data-driven methods on the other hand "place more weight on the object representation and the perceptual processing, e.g., feature extraction, similarity metrics, object recognition or classification and pose estimation" and their "parameterization [...] is less specific (e.g. an approach vector instead of fingertip positions)" [4]. The authors further distinguish between grasping *known*, *familiar* or *unknown* objects. Following this classification, this work provides a data-driven approach for unknown objects.

A. Classical Data-Driven Grasping of Unknown Objects

Data-driven approaches to Grasp Synthesis usually focus on describing the sensor input of a scene in a way, which allows for easy generation of grasps. Using 2D sensor images [15], 3D stereo images [6] or depth-images as input, they extract important features, like edges, surfaces, approximated normals or even complete shape primitives. The Early Cognitive Vision (ECV) System [6] works on 3D stereo image input and segments the sensor input into a hierarchical representation of edge and texture information. The system extracts sparse local features and categorises them into edges, textured surfaces and junctions. Those features get combined to form contours and so called surflings. On this level of segmentation, relations between those features can lead to even higher abstractions. Many surflings, which share similar position and orientation, might e.g. get combined to form a surface. With this new description of the scene, grasp synthesis becomes a matter of extracting and arranging enabling factors for grasping, like co-planarity of surfaces. Two different approaches are presented, using edge features and surface information. Using this technique, the generation of two-fingered and three-fingered grasping actions is demonstrated. While the grasping actions are tested in a dynamic simulated environment, the input images are real world examples to account for the noisy and uncertain input of an applied application.

B. Deep Learning Data-Driven Grasping of Unknown Objects

Most of the effort of classic approaches lies in designing hand-engineered features for detection of grasps. Learning these features would reduce the amount of necessary hand-engineering and would imply less generalisation. Lenz et al. [10] demonstrate the use of DCNNs on RGBD images to find the optimal grasp for an object in an image of the Cornell Grasping Dataset[1]. The grasping configuration is fully described by a five-dimensional rectangle $(x, y, height, width, rotationangle\ \alpha)$ in the image plane [16] and represents the grasp a parallel gripper would perform in the scene. A two stage model is used, in which the first network predicts the top T rectangles with the highest probability of a successful grasp, before the second one consecutively selects the single best one. While tackling many of the issues mentioned in the introduction, this technique underlies certain limitations. Limiting the approach direction for the end-effector to the normal of the image plane would be a substantially restricting constraint for interactions with the robot's environment. Furthermore, the rectangle configuration limits the possible design types of end-effectors to parallel grippers or similar setups, whereas a state-of-the-art robot hand is able to perform a variety of different grasp

[1] http://pr.cs.cornell.edu/grasping/rect_data/data.php

types. Using the Cornell Grasping Dataset, this approach also depends on hand-labeled training data, which reduces the effective scalability. Replacing this two stage model with a single DCNN results in increased accuracy while lowering computation time [9]. The approach most comparable to ours is the one presented in [11]. In this approach, finger contact patches are predicted for a grasp based on an RGBD image. They also use deep learning for the prediction of the finger contact patches, but also use segmentation and re-meshing algorithm to produce the final grasp poses. This approach is one of few, which does not grasp through the image plane only. A quantitative comparison is unfortunately not very meaningful, since they use an unknown object set and do not provide success rates. In [7], deep learning is used to complete the shape of object extracted from an RGBD image to employ classical grasp planners. Another approach learning to grasp with a gripper in the image plane is proposed in [17]. They identify several grasp candidates based on an input RGBD image and use a DCNN to rank the grasp candidates. A similar approach for grasping with a 1-DoF gripper is proposed in [18], with the difference that their method provides 6D grasp poses. Minimising the impact of imprecisions in the execution of planned grasps already during planning is proposed by [19].

III. METHOD

The following sections will outline the methods used in this work, covering the process of generating training samples, the architecture of the DCNN and the execution of the inferred grasps in simulation and on the humanoid robot ARMAR-III.

A. Objects and Database

Training a DCNN requires a significant amount of training data. Generating this large dataset in a simulation provides us with the flexibility and scalability necessary to perform the training process. For this purpose, our experiment uses object models from the KIT object models database [20], the YCB object and model set [21] and the big data grasping database in [22]. They provide laser-scanned, textured 3-D meshes of everyday objects which vary in shape, size and form. Figure 3 shows a few exemplary objects of the database. The databases provide visual meshes for rendering, reduced models for collision detection and physics parameters to perform simulation (e. g. close the end-effector around the object). The collision models and physics parameters also enable us to perform classic top-down grasp planning with the object models, using classic, existing grasp planners. [23]

B. Grasp Generation

For grasp generation we use a grasp planner [23] of the toolset Simox [24] to provide a set of force-closure grasps for training data generation. Due to the modularity of our grasping pipeline, this grasp planner is exchangeable for any other grasp planner. We use the Simox-based grasp planner as an exemplary setup to automate effortless generation of grasps. Those grasps are distributed over all different parts of the objects, which enables us to grasp them from many different approach directions. Figure 2 shows a few exemplary results of the grasp planner. Grasp planning is performed for every object and the list $\Theta = \{\Theta_0, ..., \Theta_n\}$ of force-closure grasps is appended to the corresponding object information in the database.

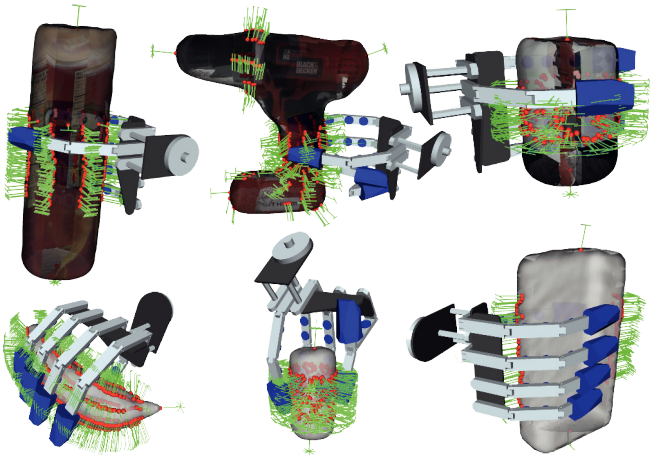

Fig. 2. Exemplary objects with results of the Simox-based grasp planner. [23] Every marker on an object represents a different grasp. We only generate Force Closure Grasps.

C. Generation of Training Samples

To generate training samples, the visual models retrieved from the object databases are placed into the simulation in front of the robot's simulated camera and the visual models are rotated and translated randomly, to generate different views. The distance and distance variance to the camera can be configured. We use a distance of $42cm$, with a distance variance of $5cm$. Rotating and translating the models also transforms the corresponding precomputed grasps in Θ accordingly. For the training process of the DCNN, only one single suitable ground-truth grasping configuration per sample of the grasps in Θ is selected, which would be considered a sufficient result if inferred by the DCNN during execution for the current random pose of the object. By limiting the output of the DCNN to one favorite grasping pose for a given input image, we reduce the complexity of the planning process by limiting the application to a specified scenario. E.g. in our case we assume that the robot is supposed to grasp an object in a table top scenario with the right hand. This complexity reduction makes the training process of the network much easier, since only one grasp has to be generated by the DCNN.

Hence, the training process gets configured with a single pose Φ, representing the most comfortable pose for the end-effector of the robot . Comfortable may e.g. take into consideration a safety distance from poses of the kinematic chain, which can are prone to IK singularities, or preferred approach direction for the end-effector in a grasping task. This pose is therefore a design variable and can be chosen. For a given object pose, we select the best suitable grasp

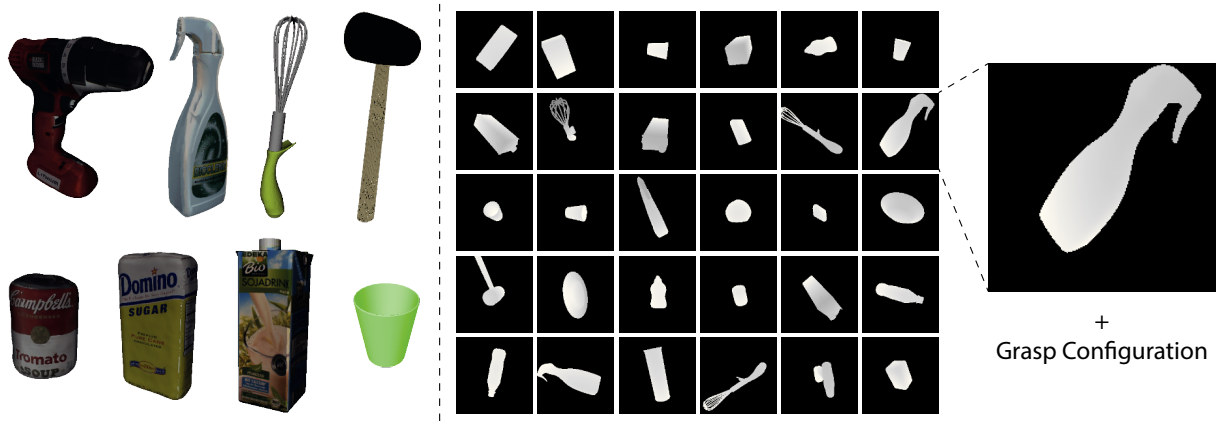

Fig. 3. Illustration of the training data rendering process. The left part of the image displays some of the objects used in our experiments. The object poses are transformed randomly during rendering, which results in multiple samples per object. The right side of the image shows finished training samples. Only depth information is rendered, resulting in a single channel 200x200 image per sample. The depth information is displayed in greyscale representation. Every sample has meta information attached, which labels it with a suitable grasping configuration. For debugging purposes and for later evaluation, object name and pose are attached as well.

Θ_{best}, by evaluating every grasp in the list of available grasps Θ. Those grasps undergo the same transformation as the object and are furthermore transformed into camera space. After this transformation, the following penalty metric determines the ranking:

$$penalty(\Theta_i) = \frac{\|\Theta_i - \Phi\|}{\theta} + \frac{axisAngle(\Theta_i, \Phi)}{\omega}$$

$$\Theta_{best} = \Theta_{argmin(penalty(\Theta))}$$

The function $axisAngle()$ computes the axis-angle difference between two poses. θ and ω act as normalisers to weight rotation and position difference against each other. In our experiments we chose $\theta = 0.04m$ and $\omega = \frac{\pi}{16}$. A rotation difference of $\frac{\pi}{16}$ now leads to the same penalty as a position difference of $40mm$ would do. Those values are chosen manually, are considered a design variable and can be adjusted if necessary. The renderer now renders a single image of the scene and extracts the depth information. The depth image and the selected grasp combined end-up in one sample and are added to the dataset, like shown in Figure 3. The rendering is repeated until the required amount of samples has been computed. One single object of the database may lead to potentially many samples in the dataset, due to the random transformation of the object pose and the selection of a suitable grasp. This approach enables us to generate large training datasets even from a rather small amount of available objects and precomputed grasps.

Note, that this process of generating training samples and selecting ground-truth grasps from a precomputed list Ψ_{res} strongly depends on the complete setup used, especially the used robot and applied task, but can generally be accomplished for a variety of different setups as it comes with no restrictions or limitations. What we show with the humanoid robot hand of ARMAR-III in the following could also be achieved with a different robot setup, a different object database and a different grasp planer. The system has to be retrained though, to work in a different setup.

D. Architecture of the DCNN

The DCNN used for this experiment has a six layer architecture, with two Convolutional Layers and three Fully Connected Layers. The input layers size fits the image size of 200 by 200 pixels and accepts floating point values, which encode the depth at this position. Both Convolutional Layers have a kernel size of *5x5* with a stride of *1x1* and each include 64 independent convolutions. Pooling and Normalisation is used after each Convolutional Layer. The Pooling Layers use Max-Pooling with a kernel size of *3x3* and a stride of *2x2*. The Normalisation parameters are $\alpha = 0.001/9.0$, $\beta = 0.75$, $bias = 1.0$ with a depth radius of 4. See [13] for additional information about Normalisation. The last Convolutional Layer is followed by a Fully Connected Layer of size 384, which subsequently connects to another Fully Connected Layer of size 192. The final output layer connects the last Fully Connected Layer as a linear combination ($Wx+b$) with 6 output neurons. The 6 output neurons encode the grasp in camera coordinates, using a Roll-Pitch-Yaw representation.

The network is trained with a batch size of 128. All together, the DCNN contains approximately 61.5 million trainable weights, which results in a memory size of 246MB for network parameters. During training, a specialised loss function is used. Similar to the selection of the best grasp in subsection III-C, rotation and position get normalised against each other. Assuming a batch size S:

$$\text{Inference Tensors:} \quad \Theta^{Pos}_{i=1...S} \text{ and } \Theta^{Rot}_{i=1...S}$$

$$\text{Ground Truth Tensors:} \quad \Psi^{Pos}_{i=1...S} \text{ and } \Psi^{Rot}_{i=1...S}$$

$$\text{Normalisers:} \quad \theta = 0.02m \text{ and } \omega = \frac{\pi}{16}$$

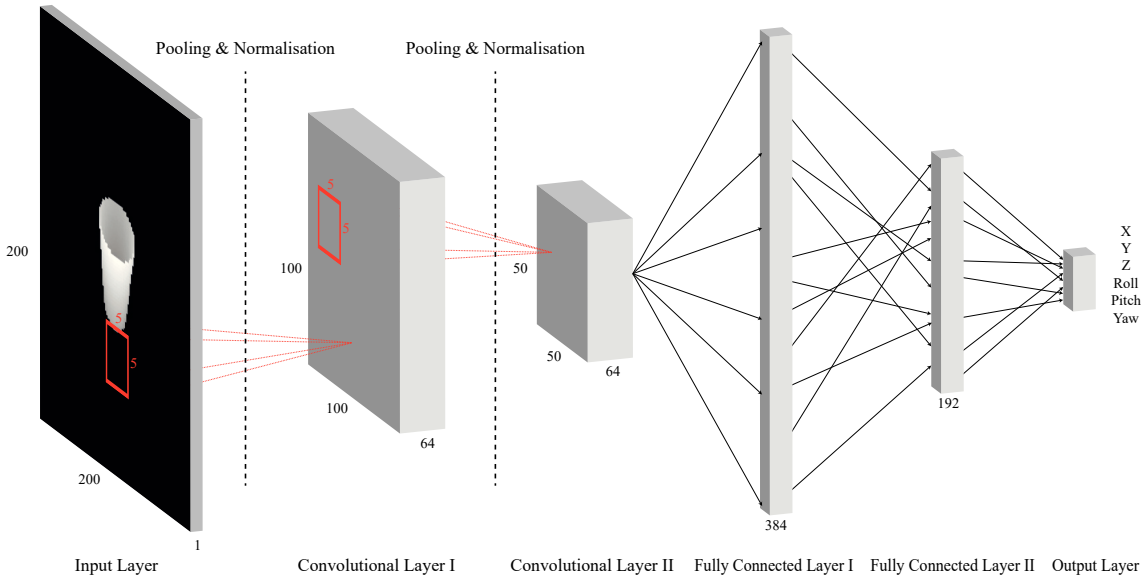

Fig. 4. The network architecture contains 6 different layers. The input layer contains the input image of size 200x200x1. Both Convolutional Layers have a kernel size of *5x5* with a stride of *1x1* and each include 64 independent convolutions. Pooling and Normalisation is used after each Convolutional Layer. The Pooling Layers use Max-Pooling with a kernel size of *3x3* and a stride of *2x2*. The last Convolutional Layer is followed by a Fully Connected Layer of size 384, which subsequently connects to another Fully Connected Layer of size 192. The final output layer connects the last Fully Connected Layer as a linear combination ($Wx + b$) with 6 output neurons. The 6 output neurons encode the grasp in camera coordinates, using a Roll-Pitch-Yaw representation.

$$\Delta Pos = \left(\frac{1}{S} \sum_{i=1}^{S} \left(\frac{\|\Theta_i^{Pos} - \Psi_i^{Pos}\|}{\theta} \right) \right)^2$$

$$\Delta Rot = \left(\frac{1}{S} \sum_{i=1}^{S} \left(\frac{\|\Theta_i^{Rot} - \Psi_i^{Rot}\|}{\omega} \right) \right)^2$$

$$Loss = \Delta Rot + \Delta Pos$$

E. Training

For training, we use 19 different objects of the databases and generate approximately 250 grasps per object with the Simox-based grasp planner. Generating 5,000 samples per object leads to 95,000 total samples and a training dataset size of 14.4 GB. Our network needs approximately 36 hours to finish training on a single Titan X (Pascal) GPU.

F. Execution

During execution, few basic pre-processing and post-processing steps are applied to increase accuracy. Training images in the data generation process are rendered in a fixed distance of $42cm$ to the camera, only applying a random distance variation of $5cm$. Evaluation and real-world application images usually leave these boundaries, unsurprisingly leading to bad grasp accuracy on objects farther away or closer to the camera. *Z-Compensation* normalises input images before they are being fed into the DCNN. For this purpose, it computes the average distance in the input image and shifts all pixels with a single subtraction operation by the difference to the desired average distance of $42cm$. The resulting grasp for the image in question is adjusted accordingly, by adding the same amount onto the Z component. As the grasp configuration is encoded in camera coordinates, this just moves the grasp farther away or into the cameras image plane. This technique enables the network to be trained on real, non-normalised distance values, while still delivering robust results for out-of-bound distances.

To further increase the grasp quality, without relying on additional information other than the sensor input, we introduce a technique called Approach Vector Post Processing (AVPP). Using the camera lens physics (focal length, field of view, etc), a single pixel in a two-dimensional depth image can be mapped back into the three dimensional camera space. By placing small collision primitives, like cubes or spheres, at the positions of the mapping, collision detection with the robot hand can be performed. This assumes full knowledge about the geometry of the robot hand, which in contrast to geometric models of the objects to grasp, is known and available prior to the exploration of the scene. In Figure 5 (2) the Grasp Center Point (GCP) of the robot hand is shown. The orientation of the GCP was chosen intentionally, aligning the Z axis of the GCP with the estimated optimal grasping approach direction (see [25]). The AVPP technique uses the inference result of the DCNN to position the robot hand into the scene and checks for collisions with the mapped collision primitives of the sensor input image. In case of a collision, the robot hand is shifted along the negative Z axis of the GCP, by a small configurable distance, and the collision gets re-evaluated. This is repeated, until the collision is finally resolved.

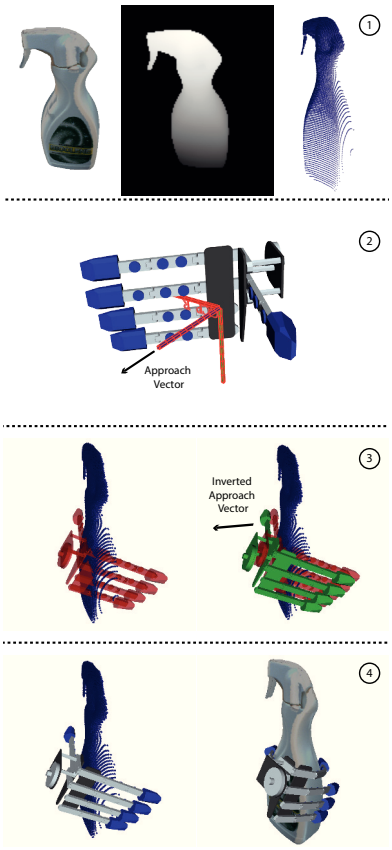

Fig. 5. Approach Vector Post Processing (AVPP) Illustration. (1) The left image in the top row shows the the object Spraybottle in the scene. The image in the middle shows the respective depth-image input of the DCNN. By mapping every pixel in the depth image back into camera space, an approximated collision model of small cubes can be generated, which is shown in the right image. The object model itself is not used for the AVPP technique! (2) The GCP of the ARMAR-III robot hand. The blue Z-axis of the GCP represents the approach direction of the TCP during a grasping task. (3) In case of a collision between the generated collision mesh and the preliminary inference of the DCNN, the end-effector is shifted along the inverted approach vector, which is the inverted Z-axis of the GCP, until the collision is resolved. (4) The final grasping configuration can now be executed and evaluated.

Figure 5 illustrates the process and shows the relation between robot hand and object before and after this post-processing step. We would like to emphasize again, that this technique does not use any information other than the raw sensor input, except the collision model of the robot hand and a valid GCP definition, which we presume as available and given prior to execution.

IV. EVALUATION

Evaluation is performed on objects, which have not been used for training before. The objects are therefore unknown to the DCNN. Similar to the process of generating training data, the evaluation objects are transformed randomly before getting rendered. Additionally saving the resulting pose of this transformation in the meta-data of the samples enables us to place the respective object into the simulation and match it with the current sample image. Using the full model of the object and the tool set Simox [24], force-closure analysis

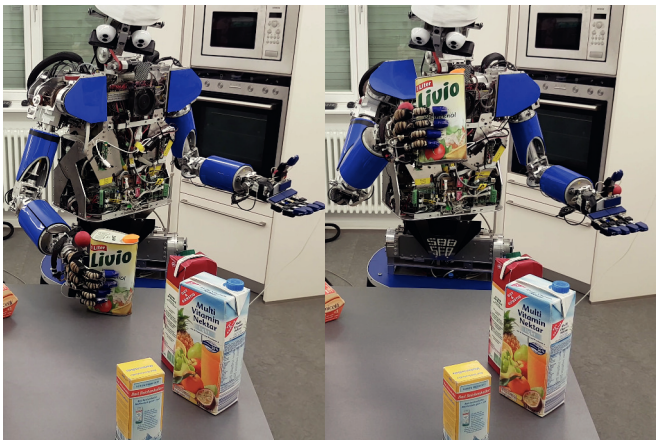

Fig. 7. ARMAR-III grasping the oil can and lifting it.

can now be performed. The meta-data of the sample is not accessible by the DCNN, which still performs inference solely on the depth-image. Table I shows the results of this evaluation on different objects with 256 executed grasps per object. We consider the grasp a success, if it is a force-closure grasp.

Since it is not practically possible nor useful to evaluate grasps in the simulation, which are in collision with the object in the first place, any inference of the DCNN with subsequent AVPP, which places the end-effector in collision with the object is consequently evaluated as non force-closure and therefore unsuccessful. This state of collision between object and end-effector is only possible because the AVPP does not work on the full object model, but only the mapped depth image collision mesh. The evaluation shows, that the approach performs very well on round objects like the tennisball or softball, which are easy to align the orientation of the end-effector with. On more box-shaped objects like salt, oil or apple tea, it is still able to achieve up to 70% force-closure performance. On more cylindrical-shaped objects, like the bleach, the approach even achieves up to 92%. Figure 6 illustrates some of the evaluation samples and the generated grasps.

To demonstrate real world application, we wrote a full implementation for the ArmarX-Framework [26] and executed it on the robot ARMAR-III. By using the already existing mapping of depth pixels into camera space of the AVPP with the information about the global pose of the camera on the robot, segmentation of e.g. the table under the objects becomes trivial. Objects can be separated from the table they stand on by cutting away every pixel with a global frame height coordinate less than the height of the table. Despite the camera noise of a real-world depth-camera, which was never included in the training data, the network generates grasps which allow the robot to grasp objects and lift them off the table. Figure 7 shows ARMAR-III grasping the oil can, a previously unknown object.

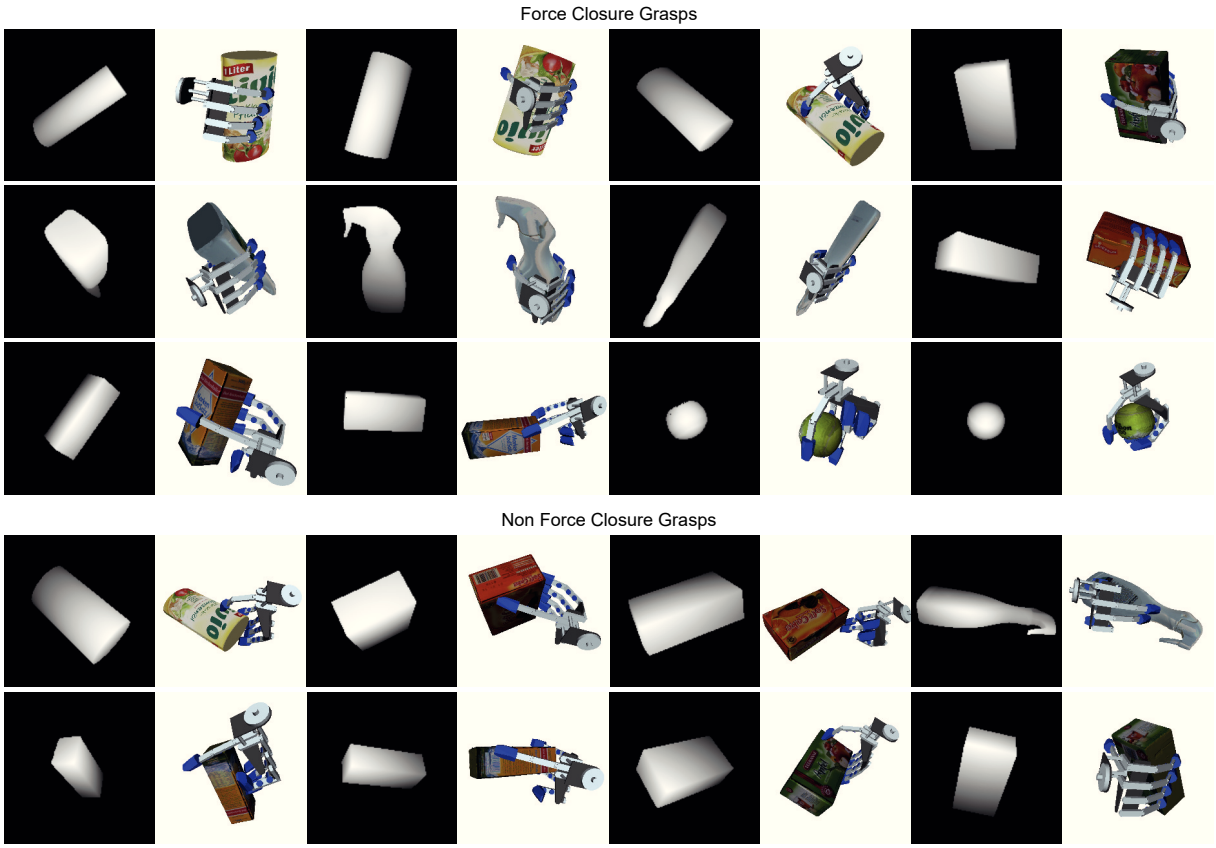

Fig. 6. Exemplary evaluation samples. The left part of every sample shows the input of the DCNN. The right part shows the resulting grasp. This illustration clearly shows, that the network has learned to align the end-effector with the objects. It simultaneously takes the reachability of this pose for the robot into account, by trying to select grasps which are similar to the grasping configuration Φ, representing the most comfortable grasp in camera space during training data generation.

TABLE I

RESULTS OF THE EVALUATION

Object Name	Grasp is valid	Average Force Closure	Quality	AVPP necessary
bottle3	82%	40%	0,12	42%
salt	94%	63%	0,29	57%
oil	91%	74%	0,19	40%
hammer10	91%	37%	0,11	75%
cup	100%	78%	0,32	51%
appletea	81%	50%	0,20	59%
apple	100%	90%	0,35	6%
cheerios	66%	12%	0,04	89%
bottle4	96%	80%	0,36	32%
bottle5	93%	82%	0,32	22%
banana	96%	43%	0,15	53%
bleach	96%	84%	0,32	18%
hammer11	93%	46%	0,10	68%
spraybottle	88%	58%	0,13	48%
softcake	66%	22%	0,11	85%
hammer8	89%	27%	0,06	83%
softball	95%	74%	0,27	37%
spam12_oz	81%	63%	0,18	54%
hammer7	89%	26%	0,06	78%

A grasp is valid, if the end-effector is not in collision with the full geometric object model after AVPP. If it is in collision, it is evaluated as non force-closure (force closure 0), no matter how good the evaluation of this grasp might be. The Average Force Closure column is the average score for the robustness evaluation of the metric described in [23]. We evaluate 300 slightly and randomly displaced grasps for each grasp solution of the network and evalute how many of those have force closure property. Higher scores indicate a more stable grasp. Grasp quality scores are explained in [24]. AVPP is not applied, if there is no initial collision between the end-effector and the mapped depth-image pixels. The last column therefore evaluates, how many grasps were post-processed.

V. CONCLUSION

This work introduces a data-driven robotic grasping approach for unknown objects using only depth-image perception as input for grasp synthesis. Motivated by the success of DCNNs in image classification tasks, we leverage similar existing techniques for grasping tasks. By training in a simulated environment, with grasps generated by classic, analytical grasp planners, the approach automatically learns grasping-relevant features. In contrast to previous work on applying deep-learning techniques to robotic grasping, this approach is not restricted to grasp through the image plane, which results in only a single approach direction. On the contrary; it is able to handle full end-effector poses and therefore arbitrary approach directions. Furthermore, it can be applied to generic end-effector setups, instead of only parallel grippers, and uses a flexible and scalable training process, without the need for hand-labeled training data. By only selecting specific suitable grasps, we are moreover able to take preferred approach direction and pose of the end-effector into account during training, to allow for flexible application of subsequent tasks, like e.g. inverse kinematics. Our approach currently only generates a single-grasp solution, binding the geometry of an object to a single grasp. By training more than one network with different approach direction setups, this limitation could be circumvented. Future work will pick up this idea and evaluate its viability.

Using ARMAR-III's right robot hand as an exemplary setup, we evaluated our work on its force-closure performance by grasping objects in simulation. We also demonstrated the real-world application on the robot ARMAR-III as can be seen in the video attachment. Both experiments show the capability of the approach to apply training experience to unknown objects while generating quality grasps. For future work, we will take advantage of the scalability of our approach and extend the training by more objects and grasps, and conduct further experiments with different DCNN architectures and hyperparameters for increased and fine-tuned performance.

REFERENCES

[1] V.-D. Nguyen, "Constructing stable grasps," *The International Journal of Robotics Research*, vol. 8, no. 1, pp. 26–37, 1989.

[2] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *The International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.

[3] A. Morales, E. Chinellato, P. Sanz, A. Del Pobil, and A. H. Fagg, "Learning to predict grasp reliability for a multifinger robot hand by using visual features," *AISC proceedings*, 2004.

[4] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[5] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann, "Grasping known objects with humanoid robots: A box-based approach," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*. IEEE, 2009, pp. 1–6.

[6] M. Popović, G. Kootstra, J. A. Jørgensen, D. Kragic, and N. Krüger, "Grasping unknown objects using an early cognitive vision system for general scene understanding," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 987–994.

[7] J. Varley, C. DeChant, A. Richardson, A. Nair, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," *arXiv preprint arXiv:1609.08546*, 2016.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and Others, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

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

[10] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[11] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, 2015, pp. 4415–4420.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and Others, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[15] Y.-H. Liu, "Computing n-finger form-closure grasps on polygonal objects," *The International journal of robotics research*, vol. 19, no. 2, pp. 149–158, 2000.

[16] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3304–3311.

[17] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[18] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 598–605.

[19] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4461–4468.

[20] A. Kasper, Z. Xue, and R. Dillmann, "The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012.

[21] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols," *IEEE Robotics and Automation Magazine*, pp. 36–52, Sept. 2015.

[22] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4304–4311.

[23] N. Vahrenkamp, E. Koch, M. Wächter, and T. Asfour, "Planning high-quality grasps using mean curvature object skeletons," *CoRR*, vol. abs/1710.02418, 2017. [Online]. Available: http://arxiv.org/abs/1710.02418

[24] N. Vahrenkamp, M. Kröhnert, S. Ulbrich, T. Asfour, G. Metta, R. Dillmann, and G. Sandini, "Simox: A robotics toolbox for simulation, motion and grasp planning," in *International Conference on Intelligent Autonomous Systems (IAS)*, 2012, pp. 585–594.

[25] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schröder, and R. Dillmann, "Toward humanoid manipulation in human-centred environments," *Robotics and Autonomous Systems*, vol. 56, pp. 54–65, 2008.

[26] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour, "The robot software framework ArmarX," *Information Technology*, vol. 57, no. 2, pp. 99–111, 2015.