Automatic generation of realistic training data for learning parallel-jaw grasping from synthetic stereo images

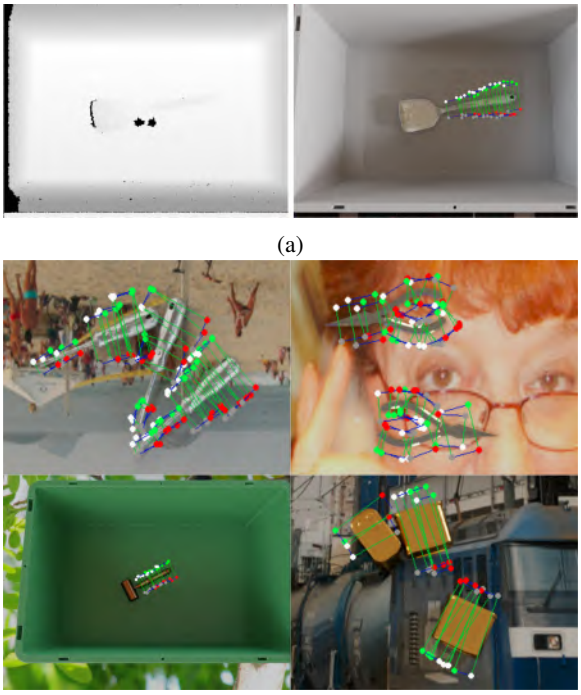
Justus Drögemüller[1], Carlos X. Garcia[2], Elena Gambaro[3], Michael Suppa[3], Jochen Steil[1], Máximo A. Roa[4]

***Abstract*— This paper proposes a novel approach to automatically generate labeled training data for predicting parallel-jaw grasps from stereo-matched depth images. We generate realistic depth images using Semi-Global Matching to compute disparity maps from synthetic data, which allows producing images that mimic the typical artifacts from real stereo matching in our data, thus reducing the gap from simulation to real execution. Our pipeline automatically generates grasp annotations for single or multiple objects on the synthetically rendered scenes, avoiding any manual image pre-processing steps such as inpainting or denoising. The labeled data is then used to train a CNN-model that predicts parallel-jaw grasps, even in scenarios with large amount of unknown depth values. We further show that scene properties such as the presence of obstacles (a bin, for instance) can be added to our pipeline, and the training process results in grasp prediction success rates of up to 90%.**

I. Introduction

A critical goal for current robotic systems is to achieve the highest possible flexibility for grasping any object, no matter its shape, size or material. Traditional approaches to define grasping points rely on significant amounts of a-priori knowledge (CAD model, center of mass) to analytically compute the location of fingers on the object surface. Recent research has focused on using learning algorithms to achieve the required flexibility for grasping applications. Learning-based grasp planning is classified according to [1] in three different categories: learning by human demonstration [2], learning by trial and error [3], [4] and learning with labeled training data [5], [6], [7]. Although results of these methods are just recently being ported to industrial processes, common success rates of above 80% indicate promising potential for real use of learning-based grasp planning approaches.

Supervised learning using labeled training data has become popular due to its high grasp success rates. In general, a Convolutional Neural Network (CNN) is trained with suitable data, and its performance is evaluated using evaluation datasets or images from real scenarios [5]. The most challenging part is often the acquisition of suitable training data [7]. Public datasets are available, consisting mostly of depth images with appropriate annotations for parallel-jaw grasps. To avoid expensive hand-labeling of images, utilization of synthetic data has gained traction, and the generalization from synthetic training data to real world grasping has been demonstrated with certain success [8], [9]. However, common datasets (e.g. [10]) provide in general lab-like, synthetic or ideal images that would not be found in real industrial applications where disturbances such as pose uncertainties of the camera or illumination variance are present and can greatly affect the quality of visual data acquired by the sensors. Also, perceptual artifacts that would normally appear in depth images (uncertainties, regions of invalid or missing depth values) are not considered in the training data, but they are effectively eliminated with image processing techniques (e.g. inpainting). Furthermore, training images usually contain a single object at a fixed distance from the camera, without variation of the camera pose. Dataset augmentation based on image processing (random zooms, crops) is commonly used to increase data variation [11], but does not include variations such as different camera tilts. Training a CNN with images that contain more than one object is not common, and if so, performance comparisons



(a)

(b)

Fig. 1: Exemplary images annotated with our pipeline: (a) SGM-based depth image of a spatula (left), and corresponding annotated image with multiple grasp options (right); (b) annotations for different objects: Screwdrivers, pliers, hammer, meat cans. The background is intentionally changed from image to image.

*This work has received funding from the European Union's H2020 programme under grant agreement No. 101017141, project ODIN.

[1]Technical University of Braunschweig, Institute of Robotics and Process Control.

[2]Technical University of Munich (TUM), Chair of Robotics, Artificial Intelligence and Real-time Systems.

[3]Roboception GmbH. Email: name.surname@roboception.de

[4]German Aerospace Center (DLR). Email: maximo.roa@dlr.de

to single object training are usually not carried out [9], [12]. Additional obstacles in the image, such as a bin in a typical bin-picking process, have not been considered so far.

In this paper, we propose a novel software pipeline for automatic generation of training data for learning parallel-jaw grasping. The produced training images are automatically labeled with multiple parallel-jaw grasps using analytical metrics and without any dependency on a specific gripper model. Our pipeline provides a higher degree of realism compared to most traditional synthetic images, both through the use of the Semi-Global Matching (SGM) algorithm [13] to obtain depth images, and also because image features (e.g. unknown depth spots, realistic shadows) and camera characteristics (e.g. field of view, different camera poses) are effectively simulated. Simultaneously, our pipeline provides flexibility for creating multiple training datasets, for instance images containing multiple objects, bins, or other specific scene properties, while allowing simultaneous variation of camera poses. Exemplary images are shown in fig. 1.

The main contribution of this paper is the creation of a pipeline for automatic generation of labeled images that 1) allows generation of data that mimics quite well the visual perception in real scenarios (including projected textures, artifacts such as regions with unknown depth values, or foreign objects such as a bin), thus closing the gap from simulation to real execution, 2) avoids any manual image pre-processing steps, 3) allows multiple grasp annotations per object, with single or multiple objects per scene without any significant quality loss of the generated grasp labels, and 4) can train a network that generalizes the grasp prediction to any stereo-based sensor. We evaluate our SGM-based datasets using the CNN proposed in [11]. Our focus is the evaluation of the network performance if trained with non-optimized depth images, i.e. we make the network directly learn depth invariances and discontinuities. The evaluation is performed using reference datasets as well as real sensor data captured with an rc_visard[1].

II. Related Work

Grasp planning for two-finger, parallel-jaw grippers has been traditionally based on analytical algorithms. These algorithms rely mostly on the object shape to define suitable grasping configurations, and they form the base for today's popular learning methods. Comprehensive overviews for such analytical approaches are provided in [1], [19].

One of the earliest works in grasp learning used a Support-Vector-Machine (SVM) trained with a set of human-labeled ranked grasps on RGB-D stereo images [17]. In robot experiments, they achieve a grasp success rate of up to 87.9% on isolated objects. This work also introduced the rectangle representation for two-finger grasps, which has been adopted by many researchers (including us) due to its simplicity and portability. The SVM was replaced by two neural networks in [18]: The first one finds a number of grasp rectangles in the image, and the second one selects the optimal grasp from the output of the first network. As training data, this work uses the Cornell Grasp Dataset, consisting of 1280 RGB-D images with human-labeled grasp rectangles. The images are captured with a Microsoft Kinect, and a grasp success rate of up to 89% for grasping isolated objects is achieved. The Cornell Dataset is one of few public datasets for grasp learning[2], thus it has been frequently used by many researchers [11], [14], [15], [16]. The Generative Grasping-CNN (GG-CNN) was presented in [11] for pixel-wise predictions of grasp quality, i.e. the network outputs a "grasp map" directly, encoding each pixel of the input depth image with a predicted grasp quality, an angle for the gripper, and a gripper opening width. Training is performed on Cornell and Jacquard [10] datasets. From the network output, the best suited grasp rectangle can be easily computed with a low inference time (19ms), achieving a grasp success rate of more than 80% with a real robot.

The learning process can also be entirely based on point cloud processing. For instance, the work in [7] relies on the BigBird database [20], a synthetic dataset of pointclouds obtained from mesh representations. The annotation procedure is based on analytical evaluations. With a final training dataset of 55 objects and 300k unique scenes, they achieve a grasp success rate on cluttered scenes of up to 93%. The mean curvature skeletons [21] of objects from the YCB [22] and KIT [23] datasets are computed in [8] to analytically generate hand poses in synthetic depth images as training data. They use about 5k depth images of 19 different objects, and evaluate the approach within a simulation by analyzing force-closure. Their grasp success rates with real robot trials highly depend on the object shapes (e.g. 90% for spherical objects, 12% on a bag of chips). Up to now, the largest training dataset for parallel-jaw grasp prediction and ranking is Dex-Net [5]. The dataset contains 6.7M unique, synthetic depth images of about 1500 objects, with one parallel-jaw grasp labeled per scene. Their learning approach comprises the sampling of a number of grasps in the depth image and a subsequent ranking performed by a Grasp Quality-CNN (GQ-CNN). They achieve grasp success rates of more than 80% (and up to 91%) on real world grasp trials, including picking from clutter or with adversarial objects.

While all of the approaches above use simple and structured training data for learning grasp prediction, [9] creates a set of more diverse training images using random scenes in a simulation with up to 5 out of 381 3DNet objects [24] simultaneously present. For each scene, 40 depth images are rendered from different camera perspectives. They use their training set to make a network learn a visuomotor controller for grasping, i.e. besides force-closure grasps they associate each annotation with Euclidean distances to the camera frame in order to learn stable gripper movements in dynamic environments. For evaluation, they compare the performance of their approach to [25] in dynamic scenarios, i.e. the objects to be grasped are moved while computing a grasp location. They significantly outperform the network

[1] https://roboception.com/en/rc_visard-en/

[2] https://www.kaggle.com/oneoneliu/cornell-grasp

TABLE I: Overview of previous work in learning-based grasping.

Work	Hand		Training Data		Network characteristics			Grasp Representation	Success rate	
	2-finger gripper	Multi-finger gripper	Synthetic	Real	Network parameters	Inference time	Input		Dataset	Robot Experiments
[5]	√	-	√	-	18M	1-3s	Depth	Other	85.7%	80±11%
[7]	√	-	√	-	N/A	N/A	Point Cloud	Other	90%	93%
[14]	√	(√)	-	√	N/A	20ms	RGB	Circle	96.5%	-
[11]	√	-	√	√	62k	20ms	Depth	Rectangle	80%	81±8-100%
[15]	√	-	-	√	N/A	0.5s	RGB	Rectangle	88.7%	-
[16]	√	-	-	√	RGB	20ms	RGB	Rectangle	99.2%	93.8%
[9]	√	√	√	-	$\sim$ 125k	0.2s	Depth	Other	-	77.3-97.5%
[8]	-	√	√	-	61.5M	N/A	Depth	Other	-	12-90%
[17]	√	-	-	√	N/A	N/A	RGBD	Rectangle	96.8%	87.9%
[18]	√	-	-	√	$\sim$ 5k	13.5s	RGBD	Rectangle	-	84-89%
[3]	√	-	-	√	$\sim$1M	0.2-0.5s	RGB	5D vector	-	$\sim$ 80%

described in [25] (77.3% vs. 22.5% grasp success rate). Another approach using camera variations is presented in [26], however, they are focused on learning the 3D shape of the object and use a random exploration for generating the grasp annotations, while we are interested in generating high quality grasp annotations for any object shape.

Table I provides an overview of previous work in learning-based grasping. In general, we observed a lack of diversity in the training sets in most published works, which restricts its subsequent applicability. Except for [9], all authors used images of single objects with static camera poses, and do not consider real effects in depth images (e.g. artifacts or discontinuities) when they are used in the training phase. The most frequently used training dataset is Cornell, potentially due to its availability and easily adaptable grasp annotations, although some other datasets are also publicly available. Also, the presence of obstacles such as bins in training images and how this affects the grasp learning capabilities has not been addressed so far. Note also that the evaluation results highly depend on the setup used by the authors, making the numbers shown in the table only partially comparable.

III. Dataset generation pipeline

Synthetically rendered images are a feasible alternative for generating the large amounts of training data required for learning-based approaches that can seamlessly generalize to real world scenarios. However, the gap between simulation and reality can degrade the performance in real world applications. Here, we propose generation of annotated synthetic data based on mimicked realistic output of a stereo camera. Depth images (or disparity maps) coming from these sensors typically have areas of unmatchable pixels, which we will refer to as *stereo discontinuities*. These appear for instance at blind spots (i.e. areas that cannot be observed by both cameras of the stereo system), surfaces with light reflections, or textureless areas. Our software pipeline, represented in fig. 2, consists of three parts: Raw data rendering, computation of grasp points, and adjustment to the required data format, i.e. transformation of grasp points in 3D Cartesian coordinates to grasp rectangles in 2D image coordinates.

1) Raw data rendering: The first part of the pipeline produces raw data with a simulated stereo sensor using the SGM algorithm [13]. We use Blender with its Python API and the Cycles engine to render synthetic scenes. Mesh models from [22], [24] and Roboception's internal database of objects serve as inputs. The object models are loaded into a physics simulation and are dropped from a certain height to a solid plane. After a predefined time, the sensor captures the scene, illuminated with a random dot projector, from a number of viewpoints. The projector is simulated as it is commonly used in real installations to increase the structure in the image, thus allowing higher quality of stereo data. For each scene, we extract the object poses with respect to the camera frame. For higher realism, and to increase the diversity of the data, a random material is assigned to each object. This step ensures varying amounts of discontinuities in the depth images, which better mimics real world scenarios. The full output of our raw data generation step for one object comprises a left and right stereo image pair (I_l, I_r), an image mask I_m, an SGM-based depth image I_z, and a text file containing relevant scene information including camera and object poses.

In order to get raw depth images, we use the SGM algorithm [13]. As SGM produces a disparity map from the left and right intensity image of the stereo system, we compute the real depth z using

$$z = \frac{b f_l}{D}, \tag{1}$$

where b is the camera's baseline, f_l its focal length, and D the computed disparity at any given pixel. In disparity maps created with SGM, unmatched pixels receive an assigned value of infinity (i.e. the corresponding pixel is right in front of the image sensor). To retain this relation in the computed depth images, we assign to those pixels a real depth of 0.

2) Generic Grasp Point Detection: We recreate the rendered raw scenes with a pointcloud representation of the corresponding object meshes (fig. 3). Our approach to find grasp points for 2-finger grippers relies on the principle of antipodal contact points, i.e. points p_c such that the line l_c connecting them lies inside both friction cones, using a friction coefficient μ [27] (fig. 4). Such grasp is force-closure, i.e. the gripper can apply opposite and co-linear forces to the object to be grasped at these points, so that any force or torque perturbation is compensated by the applied forces.

We start by computing the principal curvature vector $\vec{p}$ at each point. Using their z-component, we find points whose normal is perpendicular (within a threshold) to the camera's

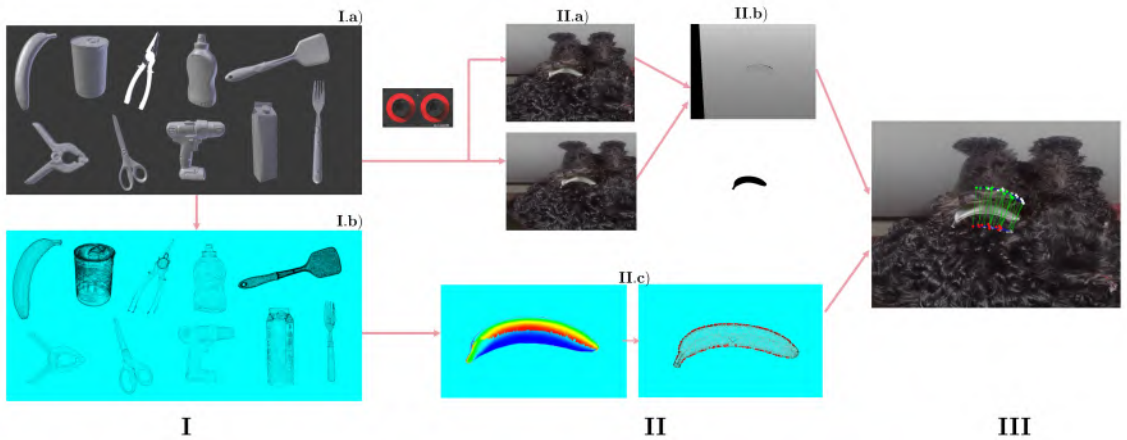
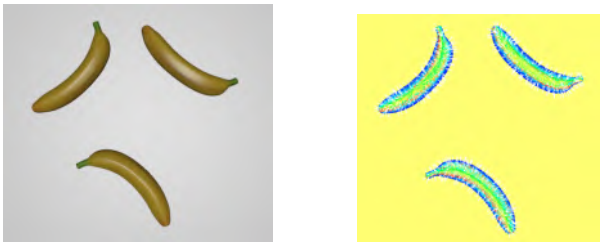
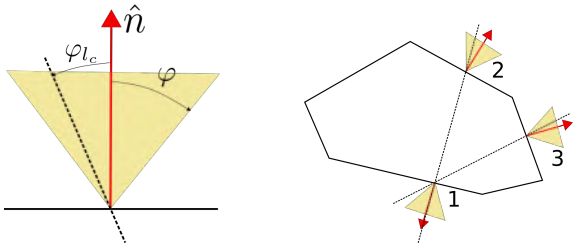


Fig. 2: Overview of the complete training data generation pipeline. We use mainly household object meshes from [22] or [24] (I.a) to generate appropriate point clouds (I.b). A simulated rc_visard stereo sensor is used to generate raw data, shown here for a banana with an arbitrary background: left and right images (II.a), depth image using a projected pattern (II.b top), and object binary mask obtained directly from Blender (II.b bottom). The rendering omniscience is used to create generic grasp point pairs based on antipodality. The raw scenes are re-created, surface normals and principal curvature are estimated, and points of high probability of being antipodal are selected (II.c). Finally, grasp rectangles are generated and synthetic datasets for network training (using the Cornell format) are produced (III).



(a) (b)

Fig. 3: Rendered RGB image (a) and corresponding recreation of the scene with point clouds (b). Dark blue areas indicate areas of high probability for finding antipodal point pairs.



(a) (b)

Fig. 4: Exemplary evaluation of antipodality: a) antipodality is fulfilled if $\varphi \geq \varphi_{lc}$ for both contact points, b) p_c 1 is antipodal to p_c 2, but not to p_c 3.

z-axis (i.e. the axis pointing towards the objects). On areas of points with relatively high $\vec{p}_z$-values, we uniformly sample N points. We then evaluate the antipodality for all possible pairs of points, as represented in Fig. 4, where $\hat{n}$ is the estimated surface normal at a given contact point p_c. The angles φ_{lc} formed between l_c and $\hat{n}$ at both p_c, are compared to the opening angle $\varphi = \arctan(\mu)$ of the friction cones.

We define a grasp $\mathcal{G}$ as a set of two antipodal contact points p_c. Each $p_c \in (x, y, z)$ has an associated φ_{lc}, which could be interpreted as the "degree of antipodality" as the estimated quality of the grasp increases when $\varphi_{lc}(p_c) \to 0$. We define a grasp rectangle $\mathcal{G}_\mathcal{R}$ that corresponds to a 2D representation in image coordinates of $\mathcal{G}$, $\mathcal{G}_\mathcal{R} := \{\tilde{c}_i, i \in [1, 4]\}$, where $\tilde{c}_i$ is the i-th corner of $\mathcal{G}_\mathcal{R}$. With $\tilde{c}_i$, any other property of $\mathcal{G}_\mathcal{R}$ can be easily obtained, such as its orientation θ or its width $\tilde{w}$ and height $\tilde{h}$. The transformation from a grasp in 3D to a grasp in image space is performed with a function $g : \mathcal{G} \to \mathcal{G}_\mathcal{R}$ given by

$$g = f(p_c, f_l, I_h, I_w). \tag{2}$$

where f_l is the focal length of the camera, and I_h, I_w correspond to the image resolution (i.e. height and width). Note that this method can be used for the synthetic data generation, as we have full object knowledge. However, computation of antipodal points on partial (real) views of the object is not straightforward due to the lack of information especially in the regions of interest for antipodality (fig. 3).

At this stage we consider each object separately, so no grasp points are found that would attempt to grasp multiple objects at the same time. However, potential grasp points could be located in areas of low probability of success, e.g. areas where objects are in contact. We catch these problematic situations with the last part of the software pipeline.

3) Grasp Rectangle Generation: This part converts the generic grasp point pairs to grasp rectangles, following the format of the Cornell dataset [6]. For each scene of raw data for a given object with computed antipodal grasp points, all grasp rectangles are generated at first by mapping the point pairs onto the raw images. Next, unwanted rectangles are pruned. We verified empirically that this step is the most important in order to achieve high quality grasp annotations. To influence the number and distribution of grasp rectangles per scene, we use a pruning algorithm based on image tiles. We split the image in $N \times M$ tiles $\mathcal{T}$ and allow only one

rectangle's center point $c_{\mathcal{R}}$ to be present in each $\mathcal{T}$. As a quality measure, a scoring function S is used to locally rank $\mathcal{G}_{\mathcal{R}}$, as follows:

$$S = w_1 \tilde{d}_C + w_2 \overline{\varphi},$$

where the distance $\tilde{d}_C$ to the object's center of mass and the average degree of antipodality $\overline{\varphi}$ of $\mathcal{G}_{\mathcal{R}_i}$ are combined with suitable weights w_1 and w_2. The ideal rectangle would minimize S, i.e. it would correspond to a grasp at the object's center of mass and with $\overline{\varphi} \to 0$. A specific gripper model is not defined; however, the maximum width of $\mathcal{G}_{\mathcal{R}}$ is adjusted dynamically depending on the shape of the object to be grasped. This approach preserves the quality of the annotations and simultaneously grants flexibility in terms of the gripper model that will be used for real world grasping. Additionally, if more than one object is present in a scene, using a pruning based on the image mask I_m prevents our pipeline from annotating grasps that would be in collision with other objects.

4) Performance: Our pipeline effectively provides annotations for a large number of objects. The generic nature of our grasp point detection together with the flexibility due to variable gripper openings allows us to annotate thin, spherical, cylindrical, polyhedral or bulky objects. The quality of the annotations is visually comparable to other publicly available datasets, e.g. Cornell and Jacquard. On the other hand, some irregularly shaped objects and hollowed objects (e.g. a bowl) were found to be problematic for the pipeline. This behavior is directly related to the computation of point features (point normal, principal curvature), which might lead to errors for those types of surfaces.

The pipeline's execution time for one scene is dominated by the raw image rendering when using Blender with full details (about 25s, roughly 80% of the total time). The creation of the depth image I_z (matching and conversion) takes another 5s, the annotation process normally takes less than 2s. All in all, the computation of a single data point (depth image and corresponding grasp annotations) requires about 32s. Rendering time measurements were performed on a Xeon 3GHz CPU system with four cores, 16GB RAM and a GeForce 1080 GPU with 8GB RAM; the remaining times were obtained on a system with a 3.4GHz CPU with four cores, 8GB RAM and no GPU. Note however that we did not focus on the optimization of these computational times, as the generation of training data is mainly an offline process.

Our approach is completely independent of scene properties such as camera pose, number of objects in the scenes, presence of bins or load carriers, or light conditions, depth image quality or other disturbances, making it extremely flexible and robust compared to other approaches.

5) Final Dataset: We used the described pipeline to obtain a dataset of around 15k unique depth images created with SGM and annotated with grasp rectangles. The set, hereafter called rc_49, is publicly available. It consists of 49 unique objects of different shapes and sizes. We cover variation of the training images in terms of object pose, camera pose (distances to the object between 30 and 80 cm, and camera tilt of up to 35°) and the amount of discontinuities in the depth image. Note that this basic set provides only images with one object present in each scene.

IV. Evaluation

To verify the benefits of our data generation, we use our synthetic SGM datasets to train a GG-CNN, and we evaluate its performance with synthetic and real images of novel objects captured with the rc_visard. The performance is compared with the same network trained on the Cornell and Jacquard datasets.

A. Network Training

We train the GG-CNN for 30 epochs, with the same basic parameters described in [11]. However, some changes are applied: First, random zooms, crops and rotations of the training images are disabled; they are not required, as our data already provides enough variation of images. Second, we adjust the depth normalization for stereo depth images with discontinuities: instead of subtracting the mean depth from the input image, pixels with 0-value are not considered, i.e. they are ignored for depth normalization. At the end of each epoch, the Intersection-over-Union (IoU) score ([10], [11]) is evaluated against a random set of the training images. The highest achieved score is called *training performance*.

B. Initial evaluation

We randomly select 10 of the 49 objects from rc_49 and exclude their scenes from the dataset. The network is trained subsequently on scenes of the remaining 39 objects, i.e. 12k images. The scenes of the excluded objects serve as evaluation test datasets. We run the experiment 5 times, and results are given in table II. A grasp prediction is considered successful if the area of overlapping of the predicted and the ground-truth rectangle is 25% or greater with respect to their area of union, and the rotation angle difference is smaller or equal to 30° [11]. The GG-CNN trained on our synthetic SGM data is able to predict a correct grasp rectangle in an average of 81.8% of the cases. The performance drop in run 2 can be traced to the particular split of the objects used for training and evaluation. The results indicate that our data can be effectively used for training the desired network, endowing it with a good generalization capability.

C. Cross-evaluation

A cross comparison of test performance while trained on a different data set, as proposed in [10], is performed for the GG-CNN using Jacquard, Cornell and rc_49 training. Table III summarizes the results, and indicates the average

TABLE II: Initial evaluation using rc_49 training data.

Run	Correctly predicted grasps	[%]
1	2636/3080	85
2	2051/2777	73
3	2388/2790	85
4	2545/3080	82
5	2525/2984	84
Average		**81.8**

TABLE III: Grasp prediction success rate (in percentage) using cross-evaluation among the three available datasets.

Training Dataset	*Evaluation dataset*			
	Cornell	Jacquard	rc_49	**Mean**
Cornell	**73**	67.6	64.3	68.3
Jacquard	21.6	**90**	48.6	53.4
rc_49	34.3	42.5	**87**	54.6
Mean	42.9	66.7	66.6	

(a) (b)

Fig. 5: Test object set, which includes objects that have been presented to the model during training (a) and novel objects (b).

IoU scores of the differently trained datasets on three subsets of the evaluation datasets with 100 images each. The numbers on the diagonal correspond to the networks' training performances, a first indicator for the expected performance. We find that training on Cornell data leads to best prediction results in average, while the GG-CNN with Jacquard training has the weakest performance.

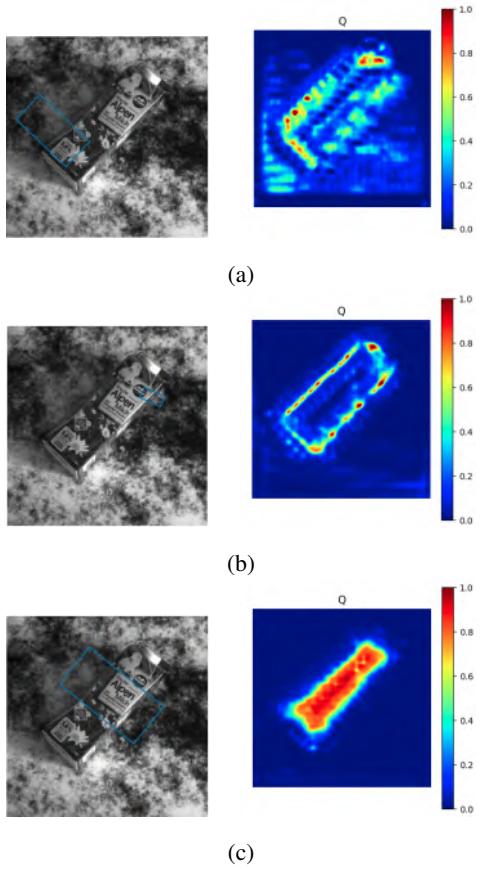
D. Evaluation with real datasets

To compare the grasp prediction performances of the differently trained networks on real stereo data, we create a new set with 25 objects, shown in Fig. 5. The set includes objects from the rc_49, and objects that are novel for all datasets, trying to cover many different shapes, sizes and surfaces. The final evaluation dataset consists of 132 real scenes of isolated objects (including 56 of "known" objects) with varying object poses (2-7 images per object). As our annotation pipeline requires explicit knowledge of the object pose in camera coordinates for annotations, we cannot use it to label and evaluate these images. Therefore, we evaluate the predicted grasp rectangles by visual inspection. To minimize bias introduced by the human evaluation, we follow a strict protocol for this procedure. It requires to label with "fail" if uncertain or unsure that the prediction will lead to success, in order to minimize false positives.

Table IV displays the evaluated prediction performance of the GG-CNN trained on Cornell, Jacquard and rc_49 on the dataset of real SGM data captured with an rc_visard65, with 65mm baseline. We also use a combined dataset with Cornell and rc_49, as both use the same format. The CNN trained on our synthetic, SGM-based dataset rc_49 significantly outperforms the other networks. Especially for bulky objects like the milk carton, the networks without adequate training data appeared to have significant problems to provide a good grasp prediction. Fig. 6 displays one exemplary comparison

TABLE IV: Grasp prediction success rate (in percentage) of the differently trained network models.

Training dataset	**Known objects (Fig. 5a)**	**Novel objects (Fig. 5b)**
Cornell	51.8	60.5
Jacquard	41.1	51.3
rc_49	94.6	84.2
rc_49 & Cornell	87.5	71



(a)

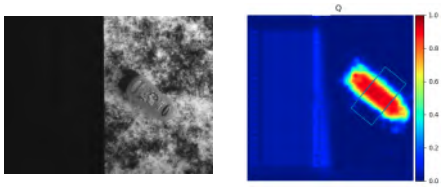
(b)

(c)

Fig. 6: Predictions of the trained CNNs on a scene of a milk carton. The left column shows the monochrome images with the predicted grasp rectangle, the right column shows the predicted pixel-wise grasp quality. Training on Cornell (a), Jacquard (b), rc_49 (c).

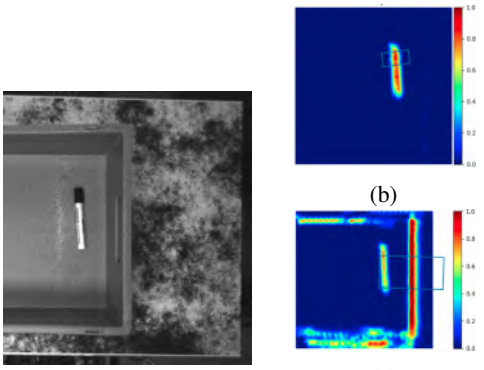
of such predictions on one scene of the milk carton. Overall, most failure cases have been detected for the glue bottle object, potentially caused by light reflections on its surface.

We further evaluate the robustness of the GG-CNN trained on our SGM data, considering now changing camera poses (i.e. tilts and distance to objects) and increasing amounts of invalid pixels (discontinuities). We consider the latter aspect of high interest for real world grasping using stereo perception, as every image contains a strip of unmatchable pixels due to the stereo system itself (i.e. the right camera cannot observe the far left, when the left camera is used as reference). Therefore, we want to explore the network's behavior if objects of interest are placed close to that "invalid strip". We generally find a high robustness to camera tilts and heights even outside the range of the training data. Also for thin objects like the clothespin, the CNN is able to predict good grasps independently of the camera height (with our



(a) (b)

Fig. 7: Scene of a shampoo bottle placed close to the edge of the table, on a scene with large amounts of unmatched pixels (a). The CNN trained on our SGM dataset is still able to predict good grasp rectangles with reasonable grasp quality (b).



(a) (b) (c)

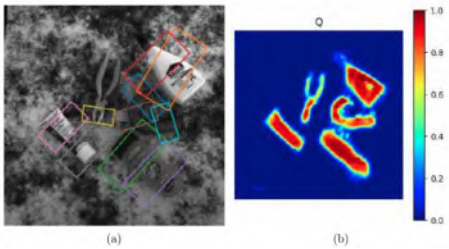
Fig. 8: Predicted grasp quality on a scene with a marker in a load carrier (a). (b): CNN prediction with training data that includes the bin. (c): CNN prediction with regular training data, which leads to collisions with the bin wall.

physical setup we changed the height in a range from 40 to 87cm). Moreover, with variation of the amount of unmatched pixels to a very high level (of about 50%, of the image), the CNN was still able to predict high quality grasps (fig. 7).

E. Presence of a load carrier

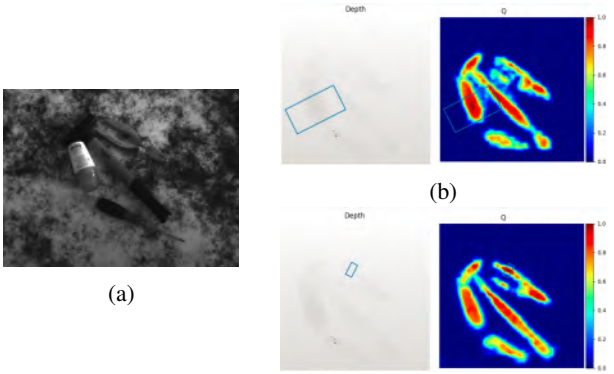
We created a training dataset with scenes containing a load carrier, which is critical for realistic bin-picking applications. Our load carrier dataset contains 10k scenes. The variations in height are the same as stated above, but the tilt variations are now limited to 15°, otherwise the sensor would not be able to see the objects inside the bin. We observed that the new scenes do not cause quality loss of the labels that our annotation pipeline produces. The most relevant change is the increase in rendering time, as ray-tracing intensifies because of the presence of the load carrier.

For evaluation, a new test setup with an rc_visard160 (160 mm baseline) is used. We capture a small dataset with 10 of the objects from fig. 5, and vary the objects' distances to the walls of the load carrier. We find that the GG-CNN with regular rc_49 training is completely unable to predict good grasp rectangles due to the presence of the carrier walls, which distract the network, as they are processed as if they were an additional object. An example can be seen in fig. 8. With our load carrier training data, the CNN was able to predict 100% good grasp rectangles on objects at the load carrier's center and 90% on objects close to its walls (76% and 0% for the network with regular rc_49 training). The



(a) (b)

Fig. 9: Exemplary multiple grasp predictions for a scene with multiple objects using the GG-CNN trained on rc_49. Intensity image (a) and predicted grasp quality (b).



(a) (b) (c)

Fig. 10: Different grasp predictions for the same scene with multiple objects. Monochrome sensor image (a), (b) inference result of the CNN trained on single objects only, (c) inference result of the CNN trained on 50% single object and 50% multi-object scenes. Note that adequate training data prevents the gripper jaws from being in collision with other objects.

only failed predictions appeared due to collisions with the load carrier walls in real grasp trial, i.e. gripper jaws would collide with the load carrier. Note that the synthetic dataset is generated with an rc_visard65, i.e. 65mm of baseline, while the evaluation dataset is obtained with a sensor of 160mm baseline. This implies that our pipeline is generic enough to allow application to different stereo sensors.

F. Multi-object grasp detection

Most deep learning approaches use as a prior that every image contains a single object, and provide a single grasp target. In general, this assumption does not hold in practice, as a scene might contain multiple objects and require multiple grasp options to choose from during execution time. The network trained with rc_49 was also able to predict good grasp rectangles in scenes with multiple objects (fig. 9). However, collisions of the gripper jaws with other objects in the scene eventually appear in a real execution. To improve this, we also evaluated the performance when scenes of multiple objects are included in the training dataset (fig. 10), in a combination of 7.5k images with single and 7.5k images with max. 3 objects each. Our pipeline produced the required annotated data without a problem. Our results showed a decrease of about 30% of predicted grasps with potential collisions with other objects.

V. Discussion

This paper presented a pipeline that is able to compute and label parallel-jaw grasps in point cloud representations of synthetic images. These images are gathered in a synthetic data set (rc_49) that is employed for analyzing the influence of training data variations on the performance of a CNN for grasp learning. For depth computation, we use the SGM algorithm that runs also on the rc_visard stereo sensors, which allows us to obtain synthetic images that mimic real images obtained with the sensor, thus closing the sim to real gap, and allowing a direct grasp execution with the robot. Our pipeline contains elements from [10] (e.g. Blender Cycles, same grasp representation, stereo depth computation, input meshes from [24]) and we use a computation of the generic grasp points similar to [25].

The proposed approach allows the generation of annotated synthetic images for training a grasp CNN, fully avoiding any pre-processing step or the addition of artificial noise in the images before the inference. The generated images mimic real sensor images (while other datasets include mainly ideal depth images) thanks to the use of the SGM algorithm with different camera poses (tilts and distance to objects). Thus, our method allows a network to learn image disturbances directly as they naturally appear on real stereo sensor data. Using images with large discontinuities in other methods requires some pre-processing such as noise removal or in-painting, which can be computationally expensive as iteration over image pixels is necessary, and might lead to image errors. Our experiments show that learning discontinuities directly can lead to robust grasp predictions even when a very high amount of unmatched pixels is used. Furthermore, the network trained on synthetic images generated for one stereo sensor can be applicable to real images coming from any other stereo sensor thanks to the use of the SGM algorithm for image processing in the generation of training data.

The pipeline also provides high quality annotations even if multiple objects or load carriers are present in the scenes. We used our automatic labeling pipeline to train a CNN for robust prediction of grasp rectangles under stereo discontinuities, varying camera poses, inclusion of multiple objects, and even presence of load carriers in the scenes. The predicted grasps are directly executable on a real robotic system without loss of performance.

Our datasets are publicly available[3] for further training and comparison with other learning approaches. As a next step, we want to evaluate the prediction performance with real data coming from vision sensors with different sensing principles, to verify the network's generalization capabilities using our synthetically generated training data.

[3] https://roboception.com/en/innovation-en/#datasets

References

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

[2] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal, "Learning of grasp selection based on shape-templates," *Auton. Robots*, vol. 36, no. 1-2, pp. 51–65, 2014.

[3] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2017.

[4] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 3406–3413.

[5] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Robotics: Science and Systems (RSS)*, 2017.

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

[7] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *Int. J. Rob. Res.*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[8] P. Schmidt, N. Vahrenkamp, M. Wachter, and T. Asfour, "Grasping of unknown objects using deep convolutional neural networks based on depth images," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 6831–6838.

[9] U. Viereck, A. ten Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images," in *Proc. Conf. on Robot Learning (CoRL)*, 2017.

[10] A. Depierre, E. Dellandrea, and L. Chen, "Jacquard: A large scale Dataset for robotic grasp detection," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3511–3516, 2018.

[11] D. Morrison, J. Leitner, and P. Corke, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," in *Proc. Robotics: Science and Systems (RSS)*, 2018.

[12] F. Chu, R. Xu, and P. Vela, "Real-world multi-object, multi-grasp detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018.

[13] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, 2008.

[14] Y. Xu, L. Wang, A. Yang, and L. Chen, "GraspCNN: Real-time grasp detection using a new oriented diameter circle representation," *IEEE Access*, vol. 7, pp. 159 322–159 331, 2019.

[15] H. Karaoguz and P. Jensfelt, "Object detection approach for robot grasp detection," *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 4953–4959, 2019.

[16] D. Park, Y. Seo, and S. Chun, "Real-time, highly accurate robotic grasp detection using fully convolutional neural networks with high-resolution images," *CoRR*, vol. abs/1809.05828, 2018.

[17] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 3304–3311.

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

[19] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous Robots*, vol. 38, no. 1, pp. 65–88, 2014.

[20] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "BigBIRD: A large-scale 3D database of object instances," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 509–516.

[21] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, "Mean curvature skeletons," *Comp. Graph. Forum*, vol. 31, pp. 1735–1744, 2012.

[22] B. Calli, A. Walsman, A. Singh, S. S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set," *IEEE Robotics and Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[23] 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," *Int. J. Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012.

[24] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3DNet: Large-scale object class recognition from CAD models," *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 5384–5391, 2012.

[25] A. ten Pas and R. Platt, "Localizing antipodal grasps in point clouds," *CoRR*, vol. abs/1501.03100, 2015.

[26] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-DOF grasping interaction via deep geometry-aware 3D representations," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018.

[27] V.-D. Nguyen, "Constructing force- closure grasps," *Int. J. Robotics Research*, vol. 7, no. 3, pp. 3–16, 1988.