

GraspME - Grasp Manifold Estimator

Janik Hager¹, Ruben Bauer¹, Marc Toussaint^{2,3}, Jim Mainprice^{1,2}

¹Machine Learning and Robotics Lab, IPVS, University of Stuttgart, Germany

²Max Planck Institute for Intelligent Systems ; IS-MPI ; Tübingen/Stuttgart, Germany

³Technische Universität Berlin ; TUB ; Germany

¹firstname.lastname@ipvs.uni-stuttgart.de ³lastname@tu-berlin.de

Abstract—In this paper, we introduce a Grasp Manifold Estimator (GraspME) to detect grasp affordances for objects directly in 2D camera images. To perform manipulation tasks autonomously it is crucial for robots to have such *graspability* models of the surrounding objects. Grasp manifolds have the advantage of providing continuously infinitely many grasps, which is not the case when using other grasp representations such as predefined grasp points. For instance, this property can be leveraged in motion optimization to define goal sets as implicit surface constraints in the robot configuration space. In this work, we restrict ourselves to the case of estimating possible end-effector positions directly from 2D camera images. To this extend, we define grasp manifolds via a set of keypoints and locate them in images using a Mask R-CNN [1] backbone. Using learned features allows to generalize to different view angle, with potentially noisy images, and objects that were not part of the training set. We rely on simulation data only and perform experiments on simple and complex objects, including unseen ones. Our framework achieves an inference speed of 11.5 fps on a GPU, an average precision for keypoint estimation of 94.5% and a mean pixel distance of only 1.29. This shows that we can estimate the objects very well via bounding boxes and segmentation masks as well as approximate the correct grasp manifold’s keypoint coordinates.

I. INTRODUCTION

As humans share tasks with robots that are increasingly more autonomous, it will become essential to provide user interfaces or robot behaviors that allow to flexibly define the task objectives. Hence in a human-robot collaborative manipulation task, knowledge of the entire object’s grasp manifold (i.e. suitable grasp candidates), provides a step in this direction (e.g. shared autonomy). Note that the rapid detection of grasp manifolds in image space can have other applications ranging from robot motion planning to character animation in video games.

In this paper, we present a grasp manifold estimator *GraspME*, based on the Detectron2 framework [2]. Our model estimates the grasp manifolds, classifies the objects and computes their bounding boxes and segmentation masks all at the same time from a 2D image. An outcome of such a grasp manifold estimation is depicted in Fig. 1. We train our model by supervised learning on simulation data from an environment we developed in PyBullet [3]. We devised two sets of objects: the first with simple geometry and the second with more complex geometry from the 50 category subset of 3DNet [4]. Our simulation environment generates RGB, depth and segmentation images together with bounding

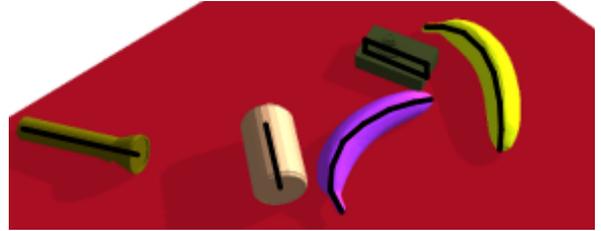


Fig. 1: Predicted grasp manifolds for complex objects using our approach, depicted as black lines.

boxes and grasp manifold keypoints for each object in a scene.

Object detection and semantic segmentation as well as grasp point localization have been improved steadily by the vision community which has led to a large list of baselines, e.g. Mask R-CNN [1] and several models based on its framework. Solutions based on grasp point detection often either rely on predefined grasp points or trial and error learning, which often proves to be rather unstable on unseen objects. An important aspect is often neglected, namely that for most objects, infinitely many grasp points exist instead of just a few predefined. This amount of grasp points can typically be defined by a manifold on a given object, mostly depending on the object geometry.

Thus, our contribution consists of

- 1) the introduction of the new problem setting of grasp manifold estimation on objects,
- 2) *GraspME*, a framework for object detection and grasp manifold keypoints estimation from 2D images and
- 3) a simulation environment to generate suitable scenes and data for this task.

Object’s grasp manifold provide more knowledge about the scene than simple grasp points. In human-robot collaboration this means providing more solutions for a handover. In space sharing scenario this grasp manifold may lead to more reactive behavior fallingback to different grasp solutions if the human moves and thus minimally disrupting the human.

Another area that could benefit from grasp manifolds is Task and Motion Planning (TAMP). For example, it could be used in Logic Geometric Programming [5], where optimization over continuously many grasp locations in a manifold - in contrast to fixing a specific grasp - could lead to a better trajectory with respect to the used control cost

or even lead to more feasible and stable solutions of the problem.

This paper is structured as follows: In Section II, we present relevant related work. We then formulate the problem of grasp manifold estimation from images and introduce notation in Section III. Before introducing our GraspME framework and implementation in Section IV, we present the dataset we work with to train our estimator in Section V. Finally, Sections VI and VII present our experiments, results and conclusions.

II. RELATED WORK

Our work merges the two research fields of object detection and grasp point detection. The first part is done by following Mask R-CNN [1] while we combine it with a keypoint detection approach for the second.

Mask R-CNN: Mask R-CNN is a successor of Faster R-CNN [6] and Fast R-CNN [7] and addresses the problem of object detection. It consists of two parts, a backbone to generate region proposals and a head to solve the actual task. To detect objects, Mask R-CNN’s head is composed of two computation branches of which one is responsible for classifying the object and generating an axis aligned bounding box while the second branch estimates the object’s segmentation mask. Mask R-CNN has also been used before to detect unseen objects in simulation in [8]. Regarding keypoint estimation, Mask R-CNN suggested the extension of their framework with a third branch to estimate human poses using keypoints. Our own framework extends this approach by applying it on a new problem of predicting grasp manifolds and their corresponding keypoints.

Grasp Point Detection: A common approach to detect grasp points is to predefine fixed points on objects and localize them to plan a grasp trajectory, e.g. [9]. These methods often lack in generalizability, as they cannot work properly on unseen objects without grasp point labels. Another possibility is to let the model learn to differentiate between good and bad grasp points and extrapolate the knowledge to unseen objects. To be able to do this distinction, the model can either rely on predefined grasp points or more common by executing random grasps on the objects and learn by trial-and-error [10], [11], [12], [13].

A benchmark for object affordances has been recently introduced [14] to evaluate point cloud deep learning networks. Our framework complements the aforementioned approaches by introducing manifolds that contain possible grasp points from which an algorithm can sample and execute a grasp.

Keypoint Detection: The topic of keypoint detection is often associated to Human Pose Estimation. There are also several datasets for this problem, e.g. MPII [15] or COCO [16]. Several approaches address Human Pose Estimation via keypoint detection and incorporating domain knowledge, e.g. the COCO 2016 keypoint detection winner [17].

However, keypoints detection has lately also being used to tackle other problems such as improving the quality of image generation [18] or object detection by estimating the

object’s center as keypoint [19]. We use keypoints to describe the grasp manifolds and estimate them during inference.

III. GRASP MANIFOLDS

A. Definition

We formalize the concept of an object’s grasp manifold as a region GM with a continuous closed border. Any point $p \in GM$ defines a potential grasp on the object, such that the closing point of the gripper is the same as this point p .

To simplify the problem, we approximate such a grasp manifold by a set of keypoints, to which we will refer to as *grasp manifold keypoints* kp_i . Typically, they are chosen as corner points that span the corresponding manifold, thus approximating it as close as possible.

B. Object and Gripper Geometry

Depending on the object’s geometry, the corresponding grasp manifold can be defined as a line or a whole surface (see next subsections).

We assume the usage of a parallel two finger gripper, i.e. to perform a grasp using an object’s grasp manifold, the gripper should be aligned parallel to the manifold such that the closing point of the gripper would be on the chosen grasp point on the manifold. We expect that this can easily be extended to other types of gripper.

C. Simple Objects

We assume that the simple objects have a lengthy shape, i.e. one of the sides along the axes is longer than the other, i.e. the object’s *main axis*. In most cases, the main axis defines already the grasp manifold, e.g. for cylinders and capsules. If we neglect any other situational circumstances, it is possible to perform a grasp on a cylinder or a capsule if the gripper is aligned parallel to the main axis of the object.

Therefore, the grasp manifold for capsules and cylinders is defined by the starting and the ending keypoint of the object’s main axis going through the object’s center, resulting in a line. An example of the grasp manifold for these two object types can be seen in Fig. 2 as a red line.

For the cuboids, the manifold containing possible grasp points is much larger. The main axis of the cuboid is still the key to define it. However, it can expand in the upper and lower direction as well, creating a plane parallel to the cuboid’s faces, as can be seen on the left in Fig. 2 in red. For simplicity, we reduced the grasp manifold for cuboids to a line (the main axis), such that only two keypoints define the grasp manifold for simple objects.

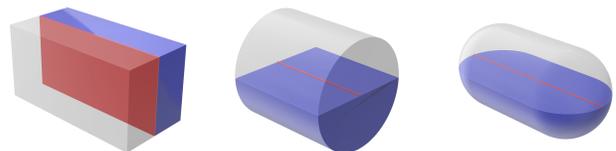


Fig. 2: The three object models in blue and transparent with the corresponding grasp manifold in red: cuboid, cylinder, capsule (left to right).



Fig. 3: The complex object models taken from 3DNet [4].

D. Complex Objects

For objects with less trivial geometries we defined the grasp manifold manually by specifying the corresponding keypoints. To simplify the problem, we restrict ourselves to a maximum of K keypoints per object which approximates the real grasp manifold.

The approximated grasp manifold is defined by connecting the sequence of keypoints $\{kp_i\}_{i=1}^k$ with $k \in [2, K]$, conditioned by the number of keypoints used to approximate the grasp manifold for the corresponding object. The grasp manifold of objects like bananas or bottles is a line, while it is a surface for objects like cameras and guitars.

IV. THE GRASPME MODEL

Our proposed model estimates the grasp manifold from 2D images via keypoints localization instead of directly computing the object’s pose, as mentioned in III.

Since this procedure is analog to human pose estimation, we use the method of Mask R-CNN [1] and the corresponding Detectron2 framework [2] as basis for our model.

A. Architecture

Similar to Mask R-CNN, our framework also consists of two main parts. First, a backbone model, the Region Proposal Network (RPN) from Faster R-CNN [6], is used on the whole image to generate region proposals. These are fed into the network’s region of interest (ROI) head for the actual task: the classification, the bounding box detection, the mask prediction and the keypoints estimation. The overall framework architecture can be seen in Fig. 4.

For the backbone, we use the ResNet-FPN variant from Mask R-CNN [1]. It consists of a ResNet [20] of depth 50, denoted as ResNet-50, or of depth 101, denoted as ResNet-101, with a Feature Pyramid Network (FPN) [21] on top. The rest of the architecture follows the suggestions from Mask R-CNN and the Detectron2 framework [2] for the Human Pose Estimation. For our experiments, we use only the 2D RGB images as input.

Depending on the object types, we set the number of keypoints to detect to $K = 2$ for the simple objects and to $K = 10$ for the complex objects. We chose this number because we found that we do not need more than 10 to approximate manifolds for our objects. Extending to more keypoints would be possible. Since not every complex object type needs 10 keypoints to define the corresponding grasp

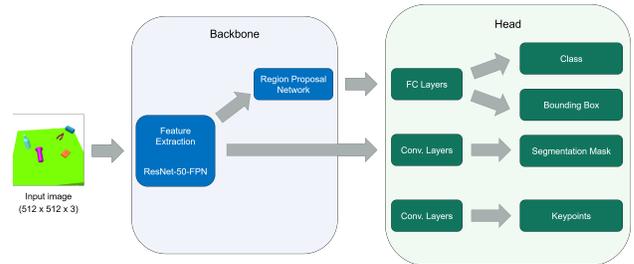


Fig. 4: GraspME framework for grasp manifold estimation.

manifold, we add extra keypoints at the object’s origin with a visibility flag equal to 0 such that each object type has a set of 10 keypoints and these additional keypoints will be neglected during the training.

Since our approach should be able to detect unknown objects, all object types belong to the same category “object” which leads to a class-agnostic object detection task. However, we additionally train a network with different object classes for comparison.

V. TRAINING DATA GENERATION

Our simulation environment is based on PyBullet [3]. The generated data should be diverse enough such that the trained model generalizes to unseen poses and objects, and to real data.

We consider the scenario where a robot grasps objects from a flat table surface while observing the scene from above. The camera’s position is sampled from a hemisphere around the tabletop’s center. The camera records RGB and depth data together with corresponding segmentation images of the observed scenes. The segmentation images are automatically generated by the PyBullet simulation.

Additionally, we store the axis aligned bounding boxes per object and the keypoints that define the grasp manifold for the corresponding object.

A. Bounding Boxes

The bounding boxes are computed by using the minimal and maximal x- and y-coordinates of the object’s segmentation mask. This way, we define the bounding box with the lower point (x_{min}, y_{min}) , its width $w = x_{max} - x_{min}$ and its height $h = y_{max} - y_{min}$. Due to occlusions, the segmentation mask and the bounding box are computed only for the object’s visible part.

B. Keypoints

We define the keypoints for each object type beforehand, and project them during the simulation on the image plane using the full projection matrix of the camera, giving us the absolute position of a grasp manifold keypoint $kp_i = (x_i, y_i)$.

Since a keypoint could be invisible due to being occluded or outside of the camera’s view, we set a visibility flag v_i for them using the COCO format [16] for keypoint detection, i.e. $v_i = 0$ if the keypoint does not exist on the object, $v_i = 1$ if the keypoint exists but is not visible and $v_i = 2$ if the

TABLE I: An overview over the training parameters chosen for the different architectures and experiments. The model name is a concatenation of abbreviations of the training parameters in the following order: backbone (R50/R101), training data (S/C/P), class agnostic (merged classes = M) or classification (C), iterations (40/80)

Model Name	Backbone	Class Agnostic	Training Data	Test Data	Iterations		
R50-S-M-40	ResNet-50-FPN	True	Simple	Simple	40k		
R50-S-C-40		False			40k		
R50-S-M-80		True			80k		
R101-S-M-40	ResNet-101-FPN	True			40k		
R101-S-M-80		True			80k		
R50-C-M-40		True			40k		
R50-C-C-40	ResNet-50-FPN	False	Complex	Complex	40k		
R50-C-M-80		True			80k		
R101-C-M-40		True			40k		
R101-C-M-80	ResNet-101-FPN	True			40k		
R50-P-M-40		ResNet-50-FPN			True	Part	40k
R50-P-M-80					ResNet-50-FPN	True	
	Complex						
	Part	80k					
R50-P-M-80	ResNet-50-FPN		True	Unseen			
				Complex			
		Complex					

keypoint is visible. Furthermore, for the simple objects only, if kp_1 is not visible but kp_2 is, we swap kp_1 and kp_2 and their corresponding visibility flags such that kp_1 should always be visible. This is possible due to the symmetric properties of those objects.

C. Object Shapes and Randomization

For cuboids, cylinders and capsules, the sizes are randomized for each object instance before rendering them, i.e. the cuboid’s length, width and height are chosen such that the length has the largest value with a small probability of generating cubes. The same holds for the capsule’s and the cylinder’s length and radius. Typical samples during the simulation can be seen in Fig. 5a.

For complex objects, we use a small subset of 11 objects from the 50 category subset of 3DNet [4]: apple, banana, bottle, camera, can, grenade, guitar, gun, maglite, mug and pliers. These synthetic 3D mesh models are rescaled and transformed from the original versions to fit our simulation. The object models are depicted in Fig. 3 while some samples from the simulation with complex objects can be seen in Fig. 1 and 5b.

To increase the diversity of the simulated dataset, we incorporate domain randomization techniques which have been proposed by Tobin et al. [22]. Hence, we randomize for each scene separately, the camera’s view point, the lighting conditions with one light source, the table’s color, the total amount of objects and the objects’ colors, positions and orientations before dropping them from above the tabletop using simulated physics.

D. Training

The training procedure for the RPN, the classification, the bounding box detection, the mask prediction and the keypoint estimation are adopted from the Mask R-CNN which is why we refer readers for further details to [1] and its predecessors Fast R-CNN [7] and Faster R-CNN [6].

During training, we apply some randomly chosen online augmentations to improve the generalizability of our model.

TABLE II: The results on the simple and complex test data w.r.t. AP in percent (%), IoU of the grasp manifold in percent (%) and mean pixel distance between keypoints on the three main tasks of bounding box, mask and keypoint estimation.

Model Name	AP ^{bb}	AP ^{seg}	AP ^{kp}	IoU _{clip}	IoU _{full}	mDist
Random	-	-	0.0	0.6 ± 1.5	0.6 ± 1.5	38.84 ± 8.01
R50-S-M-40	93.3	93.2	22.7	39.7 ± 32.8	39.7 ± 32.8	9.18 ± 9.93
R50-S-C-40	93.8	93.6	24.4	39.8 ± 32.4	39.8 ± 32.4	8.82 ± 9.44
R50-S-M-80	93.3	93.5	23.5	39.9 ± 32.5	39.9 ± 32.5	8.33 ± 9.35
R101-S-M-40	94.2	94.1	23.0	39.2 ± 32.8	39.2 ± 32.8	9.33 ± 10.19
R101-S-M-80	93.9	94.0	24.6	40.2 ± 32.5	40.2 ± 32.5	8.12 ± 9.90
Random	-	-	0.0	8.2 ± 10.9	10.1 ± 11.0	26.61 ± 8.68
R50-C-M-40	96.4	93.1	93.5	64.8 ± 29.6	26.1 ± 22.2	1.52 ± 2.79
R50-C-C-40	94.9	91.2	91.5	64.4 ± 29.7	28.4 ± 24.1	1.57 ± 3.09
R50-C-M-80	96.5	93.1	93.4	64.8 ± 29.5	25.3 ± 21.5	1.51 ± 2.85
R101-C-M-40	96.7	93.4	93.7	65.2 ± 29.8	25.8 ± 25.2	1.53 ± 2.92
R101-C-M-80	96.5	93.2	93.2	65.4 ± 29.8	25.6 ± 25.0	1.48 ± 2.87

We use the implemented augmentations from the Detectron2 framework [2] like flipping and changes of lighting, saturation, brightness and contrast, from which 2 augmentations are chosen at random for each batch.

a) *Dataset*: We generated 40,000 synthetic scenes per object set. The model is then trained on 80% of the data, i.e. on 32,000 data points, while the remaining data is withheld for validation and testing, each containing 4,000 data points.

The images are of size 512×512 pixels which we keep fixed as input for our models. We collect these amounts of data for three datasets: the first one contains only simple objects (called “Simple”), the second one contains only complex objects (called “Complex”) and the third contains 8 out of 11 complex objects (called “Part”). A fourth dataset including 4,000 data points each for validation and for testing contains the remaining three objects that did not appear in any scene in “Part” (called “Unseen”).

b) *Hyper parameters*: We chose an image batch size of 10 while the batch size per image is 64 for the RPN and 128 for the ROI head due to memory restrictions. As we cannot compare our method to any baseline due to lacking one, we use different training parameters as follows to get a better overview over the effects of these parameters.

The models are trained for 40k and 80k iterations with a base learning rate of 0.001. We decrease the learning rate by a factor of 10 after 30k iterations if trained for 40k iterations and additionally after 60k if trained for 80k iterations. Due to the similar approach, we use the two pretrained models from Mask R-CNN for the Human Pose Estimation experiments as initialization and finetune them on our datasets and problem setting. The training of the models is performed on two GeForce GTX 1080 Ti GPU. An overview over all model configurations can be found in Table I.

VI. EXPERIMENTS

The experiments are conducted on the two scenarios with simple objects and with complex objects. Since there does not exist any baseline yet that estimates whole grasp manifolds from 2D images, we can only compare different architectures and training setups of our models. An overview of the performed experiments for all of the model’s outputs can be seen in Table I. Overall, we achieve an average speed

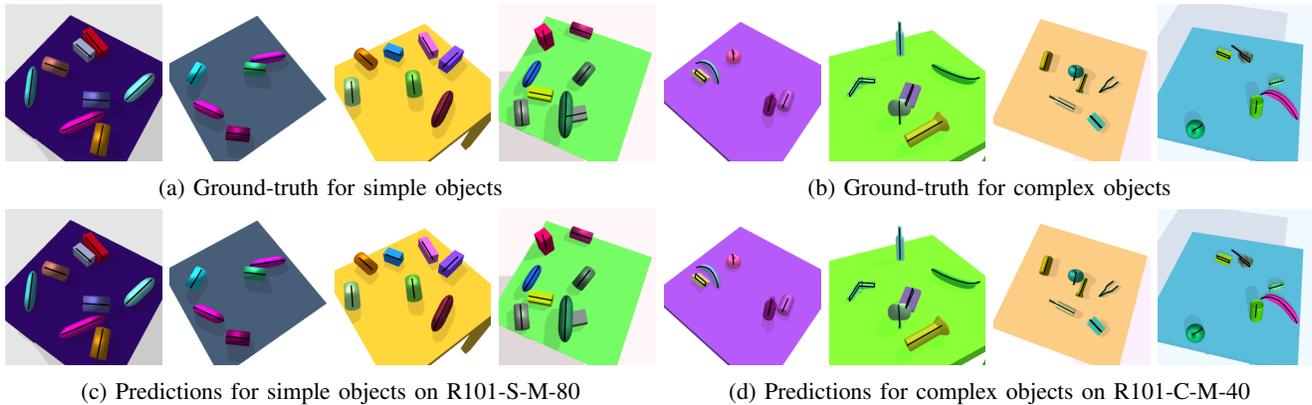


Fig. 5: Results of the predictions (bottom) in comparison to the ground-truth (top) for simple (left) and complex objects (right). The corresponding grasp manifolds are depicted as black lines.

TABLE III: The results on the complex test data including unseen objects w.r.t. average precision (AP) in percent (%) on the three main tasks of bounding box, mask and keypoint estimation.

Data	Iter.	AP ^{bb}	AP ^{seg}	AP ^{kp}	IoU _{clip}	IoU _{full}	mDist
Part	Rand.	-	-	0.0	9.6 ± 12.1	12.0 ± 12.1	26.34 ± 9.07
	40k	96.6	92.4	94.5	68.7 ± 31.2	30.5 ± 23.4	1.30 ± 3.04
	80k	96.5	92.3	94.3	69.1 ± 31.1	28.9 ± 21.8	1.29 ± 2.96
Unseen	Rand.	-	-	0.0	5.0 ± 6.4	6.0 ± 6.0	27.76 ± 7.99
	40k	60.9	86.0	1.9	13.6 ± 15.6	12.1 ± 9.7	21.09 ± 8.12
	80k	57.4	84.9	1.8	13.4 ± 15.7	12.1 ± 9.8	21.43 ± 8.20
Comp.	Rand.	-	-	0.0	0.6 ± 1.5	0.6 ± 1.5	38.84 ± 8.01
	40k	89.4	92.2	56.4	51.0 ± 37.7	24.9 ± 22.1	7.77 ± 7.04
	80k	88.0	91.8	54.1	51.1 ± 37.7	23.6 ± 20.6	7.88 ± 7.48

of 11.5 frames per second on one of the before mentioned GPUs which makes our framework suitable for real time applications.

A. Metrics

We evaluate our models using the standard COCO metrics [16] regarding average precision (AP) for the three outputs bounding box (bb), segmentation mask (segm) and keypoints (kp). As the metric for the evaluation of the latter output is optimized for human pose estimation, we additionally compute the mean Intersection over Union (IoU) of the ground-truth grasp manifold with the predicted one by using the same number of keypoints as the ground-truth (clip) or the full set of predicted keypoints (full) as well as its standard deviation.

We also measure the mean pixel distance (mDist) of the predicted keypoints to the ground-truth ones by matching them first with the closest ground-truth keypoint set per object. Afterwards, we conduct some ablation studies on the different model architectures and training setups of our models. Due to a lacking baseline, we compare our model to the Random baseline, i.e. we randomly sample keypoints from the predicted bounding boxes from the models R50-S-M-40 for the “Simple” test dataset and R50-C-M-40 for the “Complex”, “Part” and “Unseen” test datasets.

B. Simple Objects

As can be seen from the upper part of Table II, we achieve a very high accuracy with all our models for the bounding box detection and the segmentation mask estimation while the AP for keypoints is rather low. The reason for this is the objects’ symmetry, which makes it harder to estimate the correct keypoints without any additional information. Flipping the keypoint labels during training could solve this problem, by specializing on certain image regions, i.e. kp_1 could always be rather on the left of the object while kp_2 could always be on the right.

As can be seen in Fig. 5c, if we detect an object, we indeed predict the keypoints and the corresponding grasp manifolds very well in comparison with its ground-truth in Fig. 5a. The models with the ResNet-101-FPN backbone achieve the highest accuracies as well as the best values for our own metrics.

We get values around 40% for the grasp manifold IoUs, which proves that our assumptions about the low AP^{kp} values is correct. This might also be the reason for the rather high mean pixel distances between the ground-truth and the predicted keypoints. As all of the objects had two ground-truth keypoints which were fully used, the values for IoU_{clip} and IoU_{full} are the same. Our model outperforms the Random baseline by far in all of the keypoint related metrics.

C. Complex Objects

The results of the bounding box detection and the segmentation mask estimation, reported in the lower part of Table II, are of similar quality as for simple objects. However, due to the unique shape of the objects, the AP^{kp} values are much higher as the keypoints are much easier to identify on the objects.

This is also reflected in the low mean pixel distance between ground-truth and prediction of around 1.6 pixels which also leads to a grasp manifold IoU of 65%. Though using the full set of available keypoints does not seem to be helpful as the IoU is much smaller.

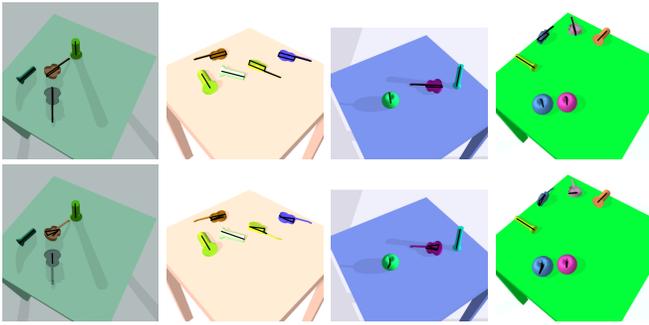


Fig. 6: Results of the predictions (bottom) in comparison to the ground-truth (top) for unseen objects on R50-P-M-40. The corresponding grasp manifolds are depicted as black lines.

Objects with less keypoints do not benefit from the additional keypoints as their grasp manifold is approximated already well enough. Due to the larger number of keypoints, the Random baseline achieves higher scores with our metrics but is still a lot worse than our framework. Overall, the best results are again achieved by the models with the ResNet-101-FPN backbone. These accurate results can also be seen in Fig. 5d, where we predict all of the keypoints nearly perfect in comparison with its ground-truth in Fig. 5b.

D. Unseen Objects

For the third experiment, we want to evaluate the generalizability of our models by using the dataset “Part” to train on a subset of the objects and test on the dataset “Unseen” containing some withheld objects. Additionally, we provide results on the full dataset “Complex” with all objects. We report the corresponding overview in Table III.

As can be seen, the models trained and evaluated on the “Part” dataset achieve similar performance to the models trained on the “Complex” dataset with respect to the COCO metrics while getting even better scores with our own metrics in terms of mean IoU with nearly 70% and mean pixel distance to the ground-truth keypoints of around 1.3 pixels. This could be due to the withheld objects that might belong to the objects that are more difficult.

Even though the model has never encountered objects from the “Unseen” dataset, it could still segment most of them from the images and estimate corresponding bounding boxes. However, computing the expected keypoints from the ground-truth was not possible, following the COCO metrics and the high pixel distance. There is some intersection of the grasp manifolds though, as can be seen from the IoU values, from which we can assume that a grasp manifold has still been found by the models.

Using the full amount of keypoints decreases the quality of the results. We assume that these low values overall come from the very different object shapes in comparison to the known object’s shapes and that it was difficult for the model to approximate the expected grasp manifold and the corresponding keypoints. However, the model might have predicted another unintentional grasp manifold that is still



Fig. 7: Results of the predictions for unseen real data on R101-C-M-40. The corresponding grasp manifolds are depicted as black lines.

a valid grasp manifold. By extending our approach to find several grasp manifolds or using more than one ground-truth grasp manifold per object, we might be able to achieve better results.

To emphasize this hypothesis, we compare some of the results on the unseen objects in Fig. 6. As can be seen, the result’s quality depends on the object. The grasp manifolds for the maglites are quite accurate while the grasp manifolds for the apples also seem to be rather close to the ground-truth. As the guitar is the most difficult of these objects regarding shape, the results are not so good in comparison with the ground-truth. However, the model still often predicts a grasp manifold by focusing on the guitar’s corpus which seems to be valid, even if it is different from the expected one. We conclude that our framework is able to estimate grasp manifolds also on unseen objects.

As the “Complex” dataset contains both seen and unseen objects, the COCO scores are obviously lower as for the models trained on all objects but we still achieve very good results in all three categories and also estimating the grasp manifolds well enough as can be seen from the IoU. Therefore, having some unseen objects mixed with known objects does not decrease the results too much, except for the mean pixel distance. The model might even benefit from having these mixed scenes and hence, achieve better results. Even for the “Unseen” dataset as well as for the “Part” and the “Complex” datasets, our framework outperforms the Random baseline.

E. Ablation Studies

To further evaluate our framework, we compare different model architectures and training setups as ablation studies. The first part is about the model’s architecture in terms of the backbone. Regarding the results of our experiments, we can see that models with the ResNet-101-FPN backbone achieved better results than with the ResNet-50-FPN backbone.

Training the model by additional 40k iterations does not seem to improve the results as much as expected regarding the COCO metrics. For the ResNet-101-FPN backbone, the model trained for only 40k instead of 80k iterations achieves even better results. The higher number of training iterations is only noticeable with respect to the IoU and the mDist.

The difference between using separate classes for each object type or simply having one for all, i.e. having the class-agnostic case, is rather insignificant. For the complex objects, the class-agnostic model achieves better results by nearly 2 points in all three categories of the COCO metrics and slightly better values for the IoU and the mDist metrics.

Therefore, we recommend the class-agnostic model as it can also be used for unseen objects without additional training.

F. Real Data

We present initial results on real camera images. However, as ground truth labels were not available, we could only evaluate the resulting images. Some of the better results are depicted in Fig. 7, which seems promising regarding the usage in real applications, as in most cases the object's main axis is predicted.

VII. CONCLUSION

Overall, we showed that our models achieve good results in terms of object detection for both simple and complex objects. Even though the values for evaluating the models on unseen objects are low, we can see that our framework could still partially generalize to these shapes and predict a grasp manifold. Thus, our model can support other methods for finding suitable grasp points on objects by spanning a whole manifold of possibilities. By having a frame rate of 11.5 fps, we expect to use our approach for real time applications.

We plan to provide a proof of concept by integrating it into a trajectory optimization framework and demonstrate our model's usage to perform human-robot-collaboration tasks, e.g. via shared autonomy and autonomous environment interactions in real scenarios. The sampled grasps in these scenarios will also be compared to those provided by other algorithms mentioned in Section II. Furthermore, we want to extend our framework in taking advantage of additional depth data to gain valuable information about the scene.

As this problem setting is unknown yet, we hope to draw interest to this scenario and encourage other researchers to approach it and use our framework as baseline.

ACKNOWLEDGMENT

We want to thank the authors of [1] and [2] for making the code of their framework publicly available. This work was conducted while Ruben Bauer was performing his Masters dissertation in the Machine Learning and Robotics Lab, University of Stuttgart, Germany. This work is partially funded by the research alliance "System Mensch".

REFERENCES

- [1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017.
- [2] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." <https://github.com/facebookresearch/detectron2>, 2019.
- [3] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." <http://pybullet.org>, 2016–2019.
- [4] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *IEEE Int. Conf. Robotics And Automation (ICRA)*, 2012.
- [5] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning.," in *IJCAI*, 2015.
- [6] S. Ren and et. al., "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2016.
- [7] R. Girshick, "Fast R-CNN," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2015.
- [8] M. Danielczuk and et. al., "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data," in *IEEE Int. Conf. Robotics And Automation (ICRA)*, 2019.
- [9] F. Spenrath and A. Pott, "Gripping point determination for bin picking using heuristic search," *Procedia CIRP*, 2017.
- [10] S. Levine and et. al., "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The Int. Journal of Robotics Research*, 2018.
- [11] J. Mahler and et. al., "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.
- [12] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.
- [13] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The Int. Journal of Robotics Research*, 2020.
- [14] S. Deng, X. Xu, C. Wu, K. Chen, and K. Jia, "3d affordancenet: A benchmark for visual object affordance understanding," *arXiv preprint arXiv:2103.16397*, 2021.
- [15] M. Andriluka and et. al., "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [16] T.-Y. Lin and et. al., "Microsoft coco: Common objects in context," in *European Conf. on Computer Vision (ECCV)*, Springer, 2014.
- [17] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [18] X. He, B. Wandt, and H. Rhodin, "Latentkeypointgan: Controlling gans via latent keypoints," *arXiv preprint arXiv:2103.15812*, 2021.
- [19] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [21] T.-Y. Lin and et. al., "Feature pyramid networks for object detection," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [22] J. Tobin and et. al., "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ Int. Conf. on Intel. Robots And Systems (IROS)*, 2017.