An Efficient Pick-and-Place Pipeline based on Quantized Segmentation

Nuno Pereira
Departamento de Informática and NOVA LINCS,
Universidade da Beira Interior
Covilhã, Portugal
nuno.pereira@ubi.pt

Luís A. Alexandre
Departamento de Informática and NOVA LINCS,
Universidade da Beira Interior
Covilhã, Portugal
luis.alexandre@ubi.pt

***Abstract*—Pick-and-place tasks in unconstrained environments use two or three deep learning methods to execute an object grasp, thus requiring highly capable computation devices. We propose a new pipeline that solves the pick-and-place tasks in unconstrained environments. Our pipeline only uses one deep learning method to make object grasping possible. It leverages quantization methods to speed up its inference time and improve its memory efficiency. We provide multiple experiments that compare the different methods and provide results for using this pipeline in the real-world. A new concept named Hybrid-QaT is introduced. Hybrid-QaT uses GPUs' power to speed up the neural networks' converging initially. Then in the last training epochs, it starts the quantization-aware training process to fine-tune the neural network's weights in an 8-bit integer representation. After the training, the conversion of the method to quantize can be done without losing accuracy.**

***Index Terms*—Deep Learning, Robotics, Quantization, Semantic Segmentation, Pick and Place, Grasping**

I. Introduction

Unconstrained picking is being used more in the industry 4.0. To enable robots to have pick-and-place tasks without requiring vibrating tables or proper palletization one possibility is the use of artificial intelligence methods to guide robots in this process. The objective in unconstrained picking is for the robot to be able to pick known objects from multiple places without any control of the environment.

To solve this type of task, a combination of multiple deep learning methods needs to be used in a pipeline. The pipelines that solve this task usually have the following sub-tasks: capturing the data, object detection or image segmentation, object 6D pose estimation, object grasping detection, motion planning, and motion execution. To capture data, the most common method used is 3D cameras (RGB-D cameras). For the object detection, methods based on bounding box detection and object classification [1] [2] are the most common. Meanwhile, the deep learning image segmentation can use two types of architectures, semantic segmentation [3] [4] [5] [6] [7] or instance segmentation [8]. The use of instance segmentation over semantic segmentation depends if there are multiple instances of the same object pilled together with occlusion. For the 6D pose estimation there are two different possible approaches to solve it. Feeding RGB images to a neural network to extract keypoints and then use the Perspective-

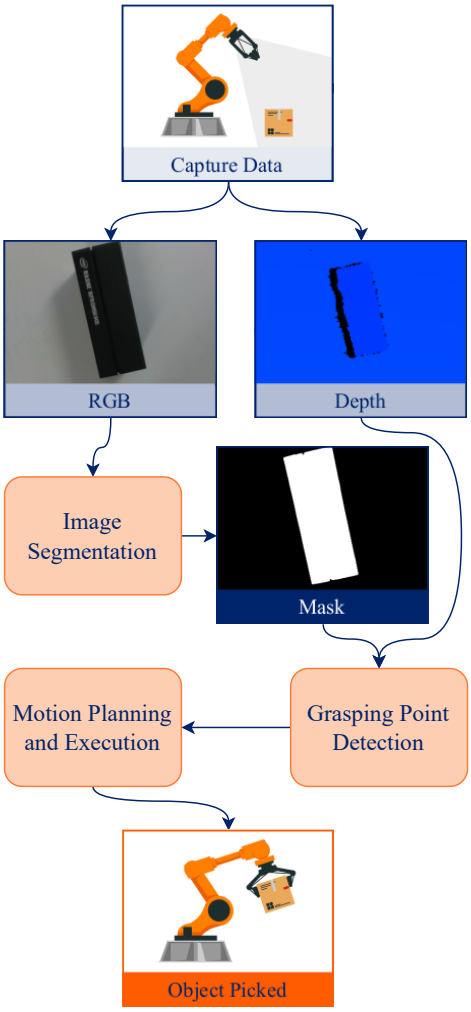


Fig. 1. Proposed efficient pick-and-place pipeline based on quantized segmentation.

This work was supported by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT-Fundação para a Ciência e a Tecnologia, through national funds, partially supported by project 026653 (POCI-01-0247-FEDER-026653) INDTECH 4.0 – New technologies for smart manufacturing, cofinanced by the Portugal 2020 Program (PT 2020), Compete 2020 Program and the European Union through the European Regional Development Fund (ERDF) and by project PTDC/EEI-HAC/30485/2017.

n-Point (PnP) algorithm to obtain the final pose estimation of the object [9] [10] [11] [12] [13] [14]. The other type of solution is the use of RGB-D data. In this case the data is fed to a deep learning method that regresses the 6D pose estimation [15] [16] [17]. One recent approach that solves both semantic segmentation and 6D pose estimation in just one neural network was proposed in [18]. This method collects the data and outputs the rotation matrix and translation vector for each object present in the scene. Finally, for the object grasping there are methods that set multiple fixed grasping position for each known object and others find the best possible grasping position [19] [20] [21]. Depending on the robotic arm and robot end-factor different libraries or APIs need to be used to plan their movement and execute them. The most common combination is the use of the Robot Operating System (ROS) with the MoveIt library. This combo is one that supports most robotic arms and can be used with a custom robotic arm if you develop your corresponding configuration file.

The pipelines that solve this task rely heavily on deep learning methods that require high computational power and need large datasets to be used during the training phases of the deep learning methods. Every time a new object is required to be picked, all the previous training processes must be done to retrain all the methods used in the pipeline. To tackle these problems in prior pipelines, we propose a novel method that speeds up the pick-and-place task by removing sub-tasks from the original pipeline and using quantization methods on the required deep learning methods. Our proposal only uses image segmentation to detect and classify objects and find grasping points. The proposed pipeline has four sub-tasks: capturing the data, image segmentation, object grasping detection, motion planing and execution. One of the advantages of our method is that only one of these sub-tasks uses deep learning (image segmentation) and the deep neural network used is quantized in order to execute as fast as possible and enable the method's deployment on edge computing devices. The previous pipelines used to execute unconstrained pick-and-place tasks require high-end GPUs during the inference to execute the predictions. In our pipeline the objective was to enable unconstrained pick-and-place without requiring high-end computers or GPUs.

The main contributions of the paper are:

- The proposal of a new approach to train models with quantization, Hybrid-QaT, that enables the reduction of models size and the increase in speed with a negligible impact on accuracy;
- The proposal of a complete pipeline for object pick-and-place, that is efficient in that it does not require a GPU after training and can hence be used with devices on the edge.

The next section contains an overview of quantization methods. The third section discusses the data sets and metrics used to evaluate our work. The fourth section shows a comparison between semantic segmentation methods used. In the fifth section we present our method followed by the experiment results using quantization and using our pipeline in the real-world. The paper ends with some conclusions in section 7.

II. Quantization

Neural network quantization reduces the precision of the used weights, biases, and activations such that they consume less memory. In other words, quantization reduces the size of a neural network's parameter representation, from typically 32-bit floats, to smaller, 8-bit integers. One clear advantage of quantization is a considerable decrease in memory use. For instance, going from 32-bit to 8-bit would result in a model size reduction of a factor of 4.

Quantization also has the potential to reduce network latency and improve power efficiency. Because operations can be carried out using integers rather than floating-point data types, network speed is increased. Most CPU cores, including those in micro-controllers, need fewer cycles to perform these integer operations that what they would require to make similar floating-point operations. Since the processing and memory access budgets are reduced, the overall power efficiency is enhanced.

Despite these advantages, the trade-off of quantization is that since neural networks are not representing information as accurately, their accuracy may suffer. However, it has been shown that quantization can frequently result in a minimal loss of accuracy [22].

A. Quantization Types

In practice, there are two main approaches to quantization: post-training quantization and quantization-aware training. Post-training quantization, as the name suggests, is a technique in which the neural network is trained using floating-point computing and then quantized. To be quantized, the neural network is frozen after training, preventing further updates to its parameters, and those parameters are therefore converted, usually, to 8-bit integers. The post-training parameters do not change; instead, the quantized model is deployed and used to make the inference.

This strategy, despite being straightforward, may result in an increased accuracy loss because all quantization-related errors happen after training and cannot be compensated.

Quantization-aware training (QaT) [22] works to compensate for the quantization-related errors. It uses 32-bit floats during training, but the actual values used are a representation of 8-bit integers as 32-bit floats. By training the neural network using this representation in the forward pass during training, the quantization-related errors will accumulate in the total loss of the model during training, and the training optimizer will work to adjust parameters accordingly and reduce the error.

Quantization-aware training has the same benefits as post-training quantization but usually has a much lower accuracy loss than post-training quantization. The main disadvantage of quantization-aware training is that the model needs to be trained with this method using the CPU, thus requiring more time to train.

III. Data sets and Metrics

A. Data sets

1) Our data set: To test our framework in a real-world environment, and then control a real robot to pick-and-place objects in an unconstrained environment, a data set was created. The classes present in the data set are, bearing (car bearing), disk (car disk), adapter (outlet plug adapter), box (cardboard box). For the adapter and box class the data set has variations of the objects present in them, thus having different colors and textures. For example the adapter class has two variations, one white with a gray texture other pure black. Fig. 2c show RGB images present in the data set.

To acquire the data set, two Intel RealSense Depth Cameras, model D415 and D435, were used to capture RGB and depth images of the scene. The semantic segmentation ground truth was manually annotated using Computer Vision Annotation Tool (CVAT). From this annotation it is also possible to obtain bounding box annotations for the data set.

The data set is composed of 5100 RGB and Depth images, 6D pose and Semantic Segmentation annotations for each object present in the scenes.

To create a robust data set that can simulate multiple real-world problems, a system to capture the data was created (Fig. 2a, 2b). It contains multiple light sources, some of them can change color. It also has a mechanical system that can move the cameras on a 2D plane and a projector that can be used to project patterns and serve as another light source.

A system like this enables the environment to be changed in term of light positions and ambiance colors. Since two of the objects that are present in the data set are metallic objects, the light projection over them can difficult their detection and also interferes with the laser pattern of the RGB-D cameras like the ones used, thus creating points of failure in depth distance readings. Also these objects can change its colors when the light source color changes. Initially, this might not seem like a big problem but, specially in the factory setting that this work will be used, is a problem having multiple different light sources. For example, if there is a siren/hazard light in the factory signaling something nearby these objects they can change from gray to red or orange thus creating problems with the depth capture and with the algorithms used to detect and estimate their pose.

The data set is divided into three subsets, train, validation and test. There are 4000 scenes in the train subset, 1000 scenes in validation subset, and 100 scenes in the test. For the train and validation subsets the scenes were captured in the system shown in Fig 2a. For the test subset, the scenes were captured in a different environment, where the camera is attached to a gripper of an Universal Robot 3 arm in a position that we consider as home. The position of the arm is assigned as a starting position for each round of movement and the position of the arm puts the end-factor 90º above the working scene and as far as possible.

The test set is similar to the factory setting where this system will be deployed and completely different from the training and validation data. With this approach it is possible to have a better estimate of the performance in the real-world.

2) LineMOD: LineMOD [23] is the most used data set to tackle the 6D pose estimation problem. It has 15 low-textured objects (although only 13 objects are used) in over 18000 images and has the ground truth pose annotated. The 13 used objects are: ape, bench-vise, camera, can, cat, driller, duck, egg-box, glue, hole-puncher, iron, lamp, and phone. Most methods only use these 13 objects due two some missing meshes in the 3D CAD files for the other two objects.

3) Cityscapes: The Cityscapes data set [24] is used for semantic interpretation of urban street scenes. This data set is the most used data set to evaluate image segmentation methods. It has 30 classes with multiple presences in a diversity of images captured over 50 cities during all four seasons and ranging from mild to moderate weather conditions. The data set has 25000 annotated images for multiple task benchmarks like semantic segmentation, instance segmentation and panoptic segmentation.

Some authors only use the 19 classes that have more instances in the data set but in this work we use all 30 classes since our main objective is not to compare with other semantic segmentation works but to evaluate different quantization methods.

B. Evaluation Metrics

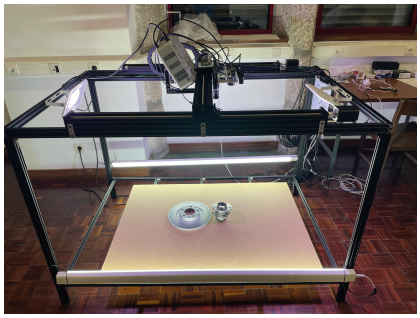
1) Pixel-wise Accuracy: Pixel-wise accuracy is the most used metric to measure the performance of methods that tackle the semantic segmentation task. This metric consists in measuring the percentage of pixels in the image that were correctly classified. The report usually consists in the percentage per class and then an average for all classes present in the data set.

When considering the per-class pixel accuracy, the ground truth consists in binary masks for each class, where a true positive represents a pixel that is correctly predicted to belong to the ground truth mask for the given class, whereas a true negative represents a pixel that is correctly identified as not belonging to the given class. This metric can sometimes provide misleading results when the class prevalence is small within the image.

2) Mean Intersection over Union: The Intersection over Union (IoU) quantifies the percentage overlap between the ground truth mask (*target*) and our predicted output (*prediction*). It divides the total number of pixels across both masks by the number of pixels shared by the ground truth and predicted masks. The overall IoU score is calculated independently for each class and then averaged across all classes to provide a mean IoU (mIoU) score for semantic segmentation.

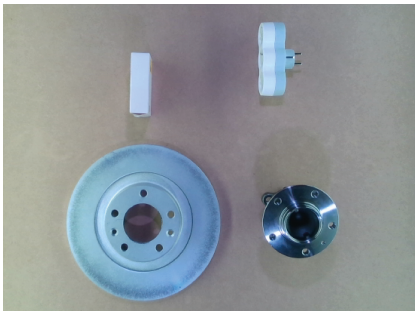
IV. Semantic Segmentation Model

We compared DeepLabV3 [6] with U-Net [3], SegNet [4] and FCN (ResNet101) [7] in our data set to choose the method that could achieve the best performance in it, to further apply multiple quantization methods and use it in the real-world. Table I shows the results of the comparison that we have made



(a) Capture system.

(b) Capture system using projections.



(c) Data set example

Fig. 2. Our data set capture system and data example.

TABLE I
ACCURACY RESULTS OBTAINED TO COMPARE DIFFERENT METHODS FOR SEMANTIC SEGMENTATION. BOLD VALUES REPRESENT THE HIGHEST ACCURACY.

Model (Backbone)	mIoU (%)	Pixel Acc (%)
SegNet	63.61	89.81
FCN (ResNet101)	64.80	90.23
U-Net	70.94	92.57
DeepLabV3	**82.33**	**98.25**

with our data set. We trained each model for 100 epochs and then the weights that performed the best in the validation sub-set were saved and loaded to execute in the test sub-set of our data set to obtain the table results. We can see that DeepLabV3 was the best performing method in the data set achieving the highest mIoU 82.33% and 98.25% of pixel-wise accuracy.

V. PROPOSED METHOD

A new pipeline was developed to solve unconstrained pick-and-place tasks. This pipeline consists of four sub-tasks where just one of the sub-tasks uses a deep learning method. The four sub-tasks are: capturing the data, image segmentation, object grasping detection, robot planing and execution. The sub-task that was solved with deep learning was the image segmentation task. To improve even further the speed of this sub-task, quantization methods were applied.

We use the DeepLabV3 model to execute the image segmentation sub-task because it achieved the highest accuracy in the comparison tests that were made in our data set that tries to represent as closely as possible our pick-and-place task.

Besides using the DeepLabV3 with 32-bit float representation we also tested the use of 8-bit integer representation of the parameters of the DeepLabV3 model. We have used both static quantization and quantization-aware training available in the PyTorch library. We did not use the dynamic quantization since it does not support 2D convolutions and the model mainly uses this type of layer. For the quantization-aware training we implemented a new hybrid approach. PyTorch quantization does not support GPU's and the model takes a long time to train in CPU. To speed up the training process we developed an hybrid method that can train the neural network mainly in GPU and then fine-tune the model parameters with quantization-aware training, thus enabling us to have the advantages of the quantization-aware training while not taking much more time to train the model for this quantization method. To test the Hybrid-QaT we setup three tests. One that relies on training 99% of the epochs using GPU and only 1% using QaT. Second test trains 95% of the epochs with GPU and 5% with QaT. Finally, 90% of the epochs train in GPU and 10% using QaT.

For the object grasping detection sub-task we start by finding the contours of the segmented object (object mask). With these contours we can find the center point of the segmented object. For this we use the OpenCV *moments* method. At this point we have the cx, cy Cartesian coordinates for the object's center. Since our approach for picking an object is keeping the gripper pointed downwards towards the object, we are just missing the end-factor rotation, θ, which we obtain with:

$$\theta = \frac{1}{2} \times \arctan\left(\frac{2 \times M[mu11]}{M[mu20] - M[mu02]}\right) \quad (1)$$

With the *moments* that enabled us to find the cx, cy of the object projected into a 2D plane and the θ angle we can find the last needed coordinate cz by the depth data in the cx, cy pixel. With this approach we can have a robust grasping method without using any deep learning technique.

VI. EXPERIMENTS

A. Semantic Segmentation with Quantization

The computer specs used for these experiments are: AMD Ryzen 5 3600, NVIDIA GeForce GTX 1080 Ti, 32GB of RAM and NVME SSD.

Table II shows all the results obtained for the DeepLabV3 method. We used the same settings for all three data sets.We trained the method for 100 epochs for each experiment and we retrieved the time that it took to train (Training Time). Then during inference time, in the test subset of each data set, we measured the inference time per image (Inference Time) and we show the accuracy for two evaluation metrics mIoU and pixel-wise accuracy (Pixel Acc). In the last column of the table we show how much memory the model requires.

Based on the obtained results from our tests, using the Hybrid-QaT technique can improve in some cases the accuracy while using less resources and being fast during inference. Static quantization is an alternative when the accuracy is not

TABLE II
EXPERIMENT RESULTS FOR 100 TRAINING EPOCHS. (*) REPRESENT ESTIMATION VALUES THAT WERE CALCULATED WITH THE TIME THAT ONE EPOCH NEEDED, MULTIPLIED BY THE 100 EPOCHS REQUIRED TO TRAIN THE METHOD.

	Pixel Acc (%)	mIoU (%)	Inference Time (s)	Training Time	Size (mb)
Data set			Our data set		
GPU	98.25	82.33	0.01	05h39	225
CPU	98.25	82.33	2.14	3 days 13h43	225
Static	98.00	82.15	1.15	05h39	58
QaT	98.20	82.24	1.14	4 days 20h40	58
Hybrid-QaT 1%	98.81	85.59	1.14	06h46	58
Hybrid-QaT 5%	98.21	81.06	1.14	11h12	58
Hybrid-QaT 10%	98.78	85.41	1.14	16h45	58
Data set			LineMOD		
GPU	97.19	85.67	0.01	24h00	225
CPU	97.19	85.67	2.09	15 days 04h05	225
Static	96.87	84.99	1.16	24h00	58
QaT	96.98	85.28	1.14	20 days 15h33	58
Hybrid-QaT 1%	97.24	85.53	1.14	1 day 04h43	58
Hybrid-QaT 5%	97.18	85.53	1.14	1 day 23h35	58
Hybrid-QaT 10%	97.20	85.60	1.14	2 days 23h10	58
Data set			CityScapes		
GPU	88.57	41.94	0.01	11h50	225
CPU	88.57	41.94	1.67	7 days 11h30	225
Static	88.54	41.71	1.12	11h50	58
QaT	88.56	41.83	1.11	10 days 04h20	58
Hybrid-QaT 1%	89.03	40.88	1.11	14h10	58
Hybrid-QaT 5%	88.99	41.20	1.11	23h28	58
Hybrid-QaT 10%	88.99	41.36	1.11	1 day 11h05	58

TABLE III
REAL-WORLD RESULTS OF THE PROPOSED PIPELINE IN AN UNCONSTRAINED PICK-AND-PLACE TASK FOR FOUR OBJECTS.

	Avg. Inference Time (s) [STDev]	Size (mb)	Grasping Success (%)
CPU	9.93 [0.10]	225	96
Static	5.61 [0.05]	58	96
QaT and Hybrid-QaT	5.59 [0.09]	58	96

critical, since there is no additional training cost to produce the quantized network.

When comparing QaT with Hybrid-QaT it is possible to see that Hybrid-QaT can achieve better results while using less time to train the neural network and having no drawbacks.

B. Real-World Object Grasping

We deployed the proposed pipeline in the real-world to execute the pick-and-place task. For the real-world experiments we used and Universal Robot 3 arm, an Intel RealSense D415 and small computer device with the following specs: Intel i5-4300U CPU, 8GB of RAM and a SATA SSD.

In the real-world experiments we measured the average inference time of the method when running on CPU, when using static quantization and quantization-aware training. This average was collected during 50 executions of the pipeline, these executions had the object to grasp or simulate the grasping of the objects present in the data set that we created. For these 50 executions we counted how many resulted in successful grasps, and in Table III we show the obtained results.

In Table III, it is possible to see that in the real-world experiments the difference in accuracy's obtained when using quantization methods was not relevant since the robot could achieve the same object grasping success rate. We can conclude that using quantization speeds up the pipeline while not loosing grasping rate.

VII. CONCLUSION

We introduced a robust and capable pick-and-place pipeline that can be used in unconstrained environments, and can be deployed in the real-world even if using computation devices with low computational capabilities, with high picking success rate. We also introduced the Hybrid-QaT technique that still uses the GPU for most of the training, which results in a large training time improvement when compared to normal QaT, and also improved accuracy results, while maintaining the small size model footprint and fast inference time that comes with QaT approaches.

REFERENCES

[1] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.

[2] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, p. 3431–3440.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[11] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-dof object pose from semantic keypoints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2011–2018.

[12] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[13] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Conference on Robot Learning*, 2018, pp. 306–316.

[14] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.

[15] N. Pereira and L. A. Alexandre, "MaskedFusion: Mask-based 6D object pose estimation," in *19th IEEE International Conference on Machine Learning and Applications (ICMLA 2020)*, December 2020.

[16] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3343–3352.

[17] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[18] N. Pereira and L. A. Alexandre, "MPF6D: Masked pyramid fusion 6D pose estimation," 2021. [Online]. Available: https://arxiv.org/abs/2111.09378

[19] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1957–1964.

[20] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning," *arXiv preprint arXiv:1709.06670*, 2017.

[21] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[22] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[23] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 international conference on computer vision*. IEEE, 2011, pp. 858–865.

[24] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.