EdgeGrasper: Fast Polygon Segmentation-Based Grasp Detection

Williard Joshua Jose, Bertram David Matabang, Joemer Aliman, Michel Onasis S. Ogbinar, Rowel Atienza

***Abstract*— Robotic grasping is a critical ability to enable robots to manipulate generic objects in real-world environments. Grasp detection is a formulation of robotic grasping in from the two-dimensional perspective. Recent works address the grasp detection problem using deep learning by training end-to-end convolutional neural network models on grasp detection datasets. However, they are not formulated to explicitly take advantage of existing priors available from computer vision models, such as object detection and instance segmentation. Guided by this insight, we propose EdgeGrasper, a fast polygon segmentation-based grasp detection network that leverages the edges of an instance segmentation efficiently produce grasping point candidates. EdgeGrasper defines grasp detection candidates using pairs of vertices of the polygon representing the instance segmentation of the object, where the midpoint, orientation, and distance of the pair of points directly define the grasp. EdgeGrasper takes a list of vertices as input to a lightweight multilayer perceptron-based network that predicts grasp confidence between all possible pairs of points. We show the ability of our method to perform fast and robust robot grasp detection against existing methods by evaluating on two challenging grasp detection datasets. We also deploy EdgeGrasper to a real-world Franka Research 3 robot arm successfully demonstrate grasp execution of various everyday objects on a table top.**

I. INTRODUCTION

Robotic manipulation is a well-studied field in robotics, where a robot interacts with the environment through the use of an end-effector [1], [2]. One of the key abilities involved in robotic manipulation is *grasping*, which is the process to securely hold objects through the use of normal and friction forces. This is crucial for different applications requiring pick-and-place of various objects. Robot grasping is a complex task that requires precise coordination of other robotic components such as perception, planning, and actuation. Thus, robot grasping, especially in real-world environments, is still an open problem.

A subset of the robot grasping field is *grasp detection*, which is the problem of determining the optimal position and orientation to place the robot's end-effector in order to securely grasp an object. Classical solutions to the grasp detection problem involve object detection and edge/shape detection [3]–[5], whereby the possible forces and torques exerted by the end-effector's fingers are analyzed for stability using physics models [1], [6]–[9]. Learning-based methods have leveraged the performance of deep learning to train computer vision models to directly predict optimal grasp poses from single image inputs of the environment using convolutional neural networks (CNNs) [10]–[16]. These models are trained end-to-end using large-scale offline grasp detection datasets and may also utilize other neural network models trained on larger more general image datasets.

Electrical and Electronics Engineering Institute, University of the Philippines

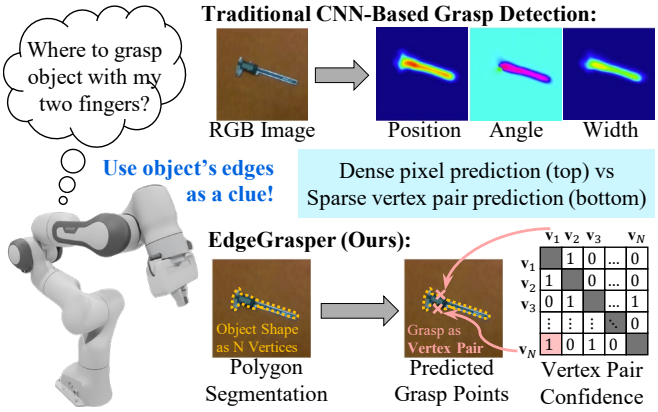


Fig. 1. Traditional end-to-end grasp detection directly predicts dense heatmaps of grasp detection candidates from input RGB images. EdgeGrasper instead utilizes the sparse polygon segmentations of the object of interest to predict feasible grasps using a fast and low-parameter network. EdgeGrasper performs on par or better than state-of-the-art grasp detection models while running as much as 120x faster (CPU) and using 42x less parameters.

Despite these advances, challenges remain regarding the accuracy and efficiency of grasp detection. Because classical CNN-based grasp detection networks are trained end-to-end on grasp detection-specific datasets, their predictions are specific to the distribution of objects in the datasets and may not translate directly to real-world or in-the-wild objects. Additionally, they are not able to leverage data or pretrained models from general large-scale image datasets which can potentially improve grasp detection performance. Although recent work [17] adapts strong instance segmentation models such as Segment Anything (SAM) [18] to implicitly use image segmentation priors, these priors act only as auxiliary objectives and are not directly used as input to the model.

To address these challenges, we propose EdgeGrasper, a fast polygon segmentation-based network to address the grasp detection problem. Instead of directly using RGB images as input, EdgeGrasper opts to use the vertices of the polygon segmentation of an object that needs grasping. Each vertex represents a potential grasping point, and pairs of vertices define candidate grasps. EdgeGrasper directly takes the list of vertices as input and predicts a confidence value for all possible pairs of vertices. This approach offers several advantages. First, operating on the space of vertices instead of dense RGB images greatly reduces the input and output spaces allowing EdgeGrasper to use a lightweight multilayer perceptron (MLP) network. Second, EdgeGrasper is able to abstract away image domain variations like texture and color and focus solely on the object's shape when predicting how to grasp it. This improves generalization and allows

EdgeGrasper to be trained on combined datasets without needing to account for changes in image distributions.

To evaluate EdgeGrasper, we benchmark against other text-conditioned grasp detection methods using available robot grasp detection datasets such as GraspAnything [19], GraspAnything++ [20], and Jacquard [21]. Our experiments show that EdgeGrasper achieves competitive grasp success rates compared to state-of-the-art while having significantly faster inference speed due to having a lighter model. Finally, we evaluate our method in real-world scenarios using the Franka Research 3 robot arm by grasping everyday objects on a table top to also showcase qualitative results.

The main contributions of our paper are:

- We propose EdgeGrasper, a novel polygon segmentation-based grasp detection method that directly predicts grasps from pairs of vertices in the segmentation polygon.
- We evaluate our model on two challenging grasp detection datasets and show that our approach performs competitively against state-of-the-art while having an order of magnitude fewer parameters.
- We deploy and evaluate EdgeGrasper on a physical Franka Research 3 robot arm and demonstrate its practical applicability when grasping household objects in the real world.

II. RELATED WORK

A. Image Segmentation Models

Image segmentation is a well-studied problem in computer vision which involves segmenting images into regions depending on defined criteria. A particular type is *instance segmentation*, where the objective is to determine the boundaries of each instance detected within an image. Given an image input, instance segmentation outputs a set of detected instances, each of which generally takes the form of either a pixel-level image mask segmentation or a polygon segmentation.

Various methods have been developed over the years, particularly using CNNs with image mask segmentation outputs. Single-stage methods include fully convolutional instance-aware semantic segmentation (FCIS) [22], SOLO [23], [24], and PolarMask [25], where the output segmentations are generated in a single pass through the model. Two-stage methods such as Mask-RCNN [26] first generate instance proposals before passing them through a mask generator and post processing to remove duplicates and incorrect segmentations.

Recent advances in vision transformers helped accelerate research on various computer vision tasks such as image segmentation. Segment Anything (SAM) [27] is a transformer-based image segmentation model trained on large amounts of data which allows it to segment previously unseen instances given a prompt. It is able to output instance masks from single images as well as videos (for Segment Anything 2 [18]). Polygon segmentation outputs, which was previously considered too complicated due to its sequential output requirement, also started becoming competitive [28], [29]. For certain shapes (and above a certain size), polygon segmentation is a more compact representation than pixel-based mask representations. When LLMs started gaining traction, attention started shifting towards adapting vision-language foundation models (VLFMs) to diverse downstream tasks, including image segmentation. Some of these VLFMs, such as GSVA [30], PSALM [31], and Unified-IO [32], are able to directly output mask segmentations using an image decoder head. Others, like Florence-2 [33], directly output polygon segmentations.

Because instance segmentation is a complex problem due to the dense prediction output needed, the aforementioned segmentation models are generally able to learn rich features from the input images which can be used for various downstream tasks. Segmentation models have been used as backbone models for other applications such as image classification, object detection, monocular depth estimation, image-to-image translation, image super-resolution, background remove, etc. Finally, the outputs of instance segmentation models provide excellent prior information on objects of interest and can be directly utilized for downstream tasks such as for robotic manipulation.

B. Grasp Detection

Robotic grasping has extensively been explored in the field of robotics. Grasp detection is the problem of determining the positions of valid grasp configurations in a scene that satisfy a set of criteria in order to successfully manipulate an object. A common simplified grasping formulation is antipodal grasping [13], [16], where there are two robotic fingers exerting opposite normal forces on an object to allow for grasping.

The classical solutions to grasp detection involve analytic solutions, where geometrical and physical models are created for the target objects for grasping. Analytic solutions to the grasp detection problem can involve shape [3], [4] and corner detection [5] to geometrically predict the optimal location of grasping points. Additionally, the possible forces and torques exerted by the end-effector's fingers can also be analyzed for stability using physics models [1], [6]–[9]. However, these methods generally make assumptions for computational tractability such as simplified contact models, Coulomb friction, and rigid body modeling, which introduce errors to the grasp predictions [2].

Recent works explore data-driven and learning-based approaches to grasp detection, with the potential for better grasp generalizability and robust performance. With the increased performance of convolutional neural networks (CNN), several works have implemented end-to-end trained CNN-based grasp detection networks, using depth images as input [10]–[12], RGB images as input [13], or both depth and RGB images [14]–[16]. Although this has led to a similar boost in grasp detection performance, these works generally predict context-free grasp predictions and lack the ability to explicitly condition predictions on other objectives. Additionally, these models directly predict dense grasp features end-to-end, and are not directly able to utilize

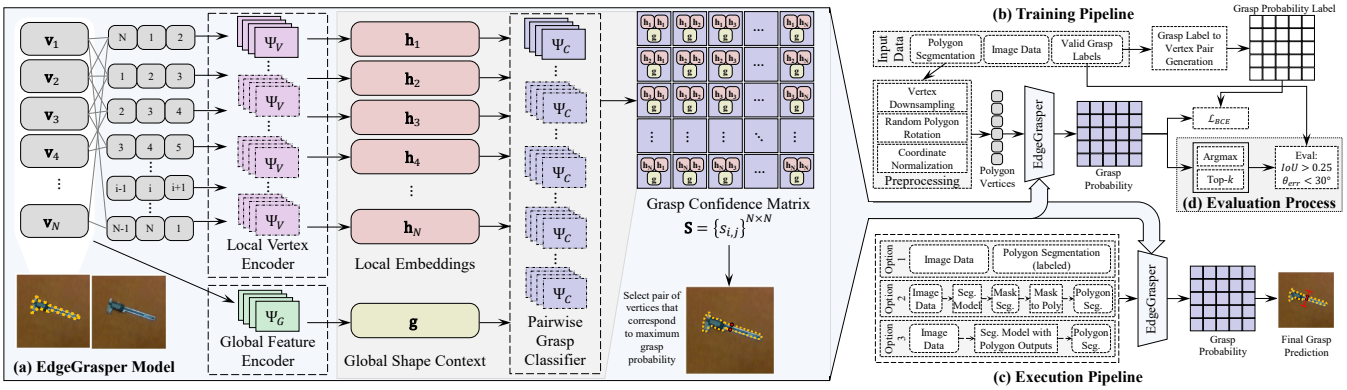


Fig. 2. An overview of our EdgeGrasper approach. (a) EdgeGrasper processes the polygon's vertices ($\mathbf{v}_i$) to produce local feature embeddings ($\mathbf{h}_i$) and a global shape context ($\mathbf{g}$). These embeddings are combined for all vertex pairs and classified to produce an $N \times N$ grasp confidence matrix ($\mathbf{S}$). (b) During training, input polygons are are preprocessed. The predicted grasp confidence matrix is learned through binary-cross entropy against labels of vertex pairs generated from the ground-truth grasp labels. (c) EdgeGrasper requires polygon vertices during execution which can either be directly provided or generated by segmentation models with either mask or polygon outputs. (d) During evaluation, the highest-scoring vertex pair is chosen as the final predicted grasp and compared against ground-truth grasp labels.

possibly helpful image priors. In particular, standard computer vision tasks object detection, edge detection, or instance segmentation provide rich information about an object's location as well as possible grasping points from the object's edges. GraspSAM [17] is a recent work that indirectly takes advantage of instance segmentation priors by utilizing a pre-trained SegmentAnything (SAM). It finetunes on the grasp detection task by adding and training adapters, learnable token embeddings, and a grasp detection prediction head alongside the original instance segmentation head. Although it has strong performance in grasp detection, it has significantly more parameters compared to the end-to-end CNN networks due to the SAM model. Additionally, it only implicitly takes advantage of image segmentation priors from the pre-trained SAM weights and by finetuning segmentation as an auxiliary objective; however, the segmentation information itself is not directly used for grasp detection.

C. Vision-Language Foundation Models for Robotic Manipulation

Recent advances in large language models (LLMs) and transformers have renewed interest in unifying different modalities into a shared embedding space. To this end, various multi-modal vision-and-language models have been developed. Initially, these models were trained and evaluated on the same fixed task (such as image captioning, visual question answering, image-text retrieval, and image-text generation). Of particular note is the ability of VLFMs to reason with natural language, which naturally allows them to serve as an interface for conditioning and contextualizing other tasks. Later on, large scale datasets have enabled scaling multi-modal transformer-based models and increasing their performance. These so-called *vision-language foundation models* (VLFMs) can be jointly trained across multiple tasks, and are able to be directly adapted to other tasks such as image classification, object detection, and image segmentation [33]–[40].

The rapid pace of development of natural language processing and multi-modal learning together with computer vision deep learning models has led to the emergence of various text-conditioned grasp detection models. A common strategy employed by several works [20], [41]–[43], [43] is to use CLIP [44], a model that aligns images and text in a shared embedding space, as a text encoder to use text features as a conditioning variable or as part of the input to the grasp detection method. However, relying solely on CLIP and related models may prove counterproductive, as these models may not have been trained on tasks with fine-grained details, which may be critical for accurate grasp detection. Additionally, CLIP does not have the ability to reason with text and can only predict text-conditioned grasping based on proximity in the embedding space. To address this challenge, RT-Grasp [45] improves on text-conditioned grasp detection by fine-tuning a LLaVa [34] model to directly output grasp candidates framed as VLFM responses. This allows RT-Grasp to have strong text reasoning inherited from the LLaVa model. However, since LLaVa is not explicitly trained on fine-grained instances, there is likely a performance gap when predicting dense grasp outputs.

III. APPROACH

A. Polygon-Based Grasp Detection

We address the problem of antipodal robotic grasping where a robot gripper makes contact with an object at exactly two points. Formally, given an input image, the objective is to predict a grasp configuration $\mathbf{g} = (\mathbf{c}_g, \phi_g, w_g, q_g)$, where $\mathbf{c}_g = (u, v)$ is the grasp center in image coordinates, ϕ_g is the grasp orientation, w_g is the required gripper width, and q_g is a quality score indicating the likelihood of success.

Instead of directly predicting these grasp parameters from dense pixel-level features, we instead reformulate the grasp detection problem by operating on a polygonal approximation of the target object's segmentation mask, represented as an ordered set of N vertices, $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^{N}$, where each vertex

$\mathbf{v}_i \in \mathbb{R}^2$. We aim to identify a pair of vertices from this set, $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{V} \times \mathcal{V}$, representing an antipodal grasp. This pair of vertices defines the grasp configuration as follows:

- Grasp center ($\mathbf{c}_g$): The midpoint of the line segment connecting the two vertices, calculated as $\mathbf{c}_g = \frac{\mathbf{v}_i + \mathbf{v}_j}{2}$.
- Grasp angle (ϕ_g): The orientation of the grasp, defined by the angle of the vector pointing from $\mathbf{v}_i$ to $\mathbf{v}_j$. Mathematically, $\phi_g = \text{atan2}((\mathbf{v}_j - \mathbf{v}_i)_y, (\mathbf{v}_j - \mathbf{v}_i)_x)$.
- Grasp width (w_g): The separation between the two grasp points, computed as the Euclidean distance $w_g = \alpha \|\mathbf{v}_j - \mathbf{v}_i\|_2$, where α is a grasp width scaling hyperparameter.

The problem is thus transformed into learning a scoring function $f(\mathbf{v}_i, \mathbf{v}_j)$ that predicts the quality of a potential grasp given a set of vertices $\mathcal{V}$.

B. EdgeGrasper Architecture

We introduce EdgeGrasper, a novel grasp detection approach that utilizes polygon segmentations of objects as grasping edges. This network is designed to be computationally efficient by directly processing geometric vertex data rather than high-dimensional image data. In particular, given a list of vertices from the polygon segmentations, EdgeGrasper learns to predict grasp confidence for each pair of vertices which are interpreted as candidate grasping points. The EdgeGrasper architecture is composed of three synergistic modules: a Vertex Feature Extractor for encoding local geometry, a Global Feature Encoder for capturing the holistic shape context, and a Pairwise Classifier for final grasp confidence prediction. Our proposed network architecture is designed to compute a grasp confidence matrix $\mathbf{S} \in [0, 1]^{N \times N}$, where each element $s_{i,j}$ represents the model's confidence in the grasp formed by the vertex pair $(\mathbf{v}_i, \mathbf{v}_j)$.

1) Vertex Feature Extractor: The initial stage of our network involves extracting a descriptive feature embedding for each vertex that encapsulates its local geometric context. For a given vertex $\mathbf{v}_i$, we define its local context, $\mathbf{x}_i \in \mathbb{R}^6$, by concatenating its coordinates with those of its immediate neighbors in the polygon sequence as $\mathbf{x}_i = \mathbf{v}_{i-1} \oplus \mathbf{v}_i \oplus \mathbf{v}_{i+1}$, where $\oplus$ denotes the concatenation operator and indices are taken modulo N. This contextual vector $\mathbf{x}_i$ is then processed by a multi-layer perceptron (MLP), the Vertex Feature Extractor $\Psi_V(\cdot)$, to generate a high-dimensional local embedding $\mathbf{h}_i$:

$$\mathbf{h}_i = \Psi_V(\mathbf{x}_i; \theta_V) \in \mathbb{R}^d \tag{1}$$

where θ_V are the learnable parameters of the MLP and d is the feature dimension. This operation is applied independently to all vertices to produce a set of local feature embeddings $\mathcal{H} = \{\mathbf{h}_i\}_{i=1}^N$.

2) Global Feature Encoder: Understanding the overall shape of the object is vital for distinguishing between viable grasps. To do so, we introduce a Global Feature Encoder, $\Psi_G(\cdot)$, which distills the entire set of vertex coordinates into a single global shape context, $\mathbf{g}$. This is achieved using an MLP transformation:

$$\mathbf{g} = \Psi_G(\mathbf{v}_1, \ldots, \mathbf{v}_N; \theta_G) \in \mathbb{R}^d \tag{2}$$

where θ_G are the parameters of the global encoder. This global context $\mathbf{g}$ serves as a shape context that informs the final pairwise classification.

3) Pairwise Grasp Classifier: We use both the local vertex embeddings and global shape context to predict the grasp confidence of every possible vertex pair $(\mathbf{v}_i, \mathbf{v}_j)$. The Pairwise Grasp Classifier, $\Psi_C(\cdot)$, is an MLP head designed for this purpose. For each pair of vertices, a composite feature vector $\mathbf{f}_{i,j}$ is constructed by concatenating their respective local embeddings and the shared global context $\mathbf{f}_{i,j} = \mathbf{h}_i \oplus \mathbf{h}_j \oplus \mathbf{g} \in \mathbb{R}^{3d}$. This concatenated feature vector provides the classifier with information about the specific local geometry around each candidate grasp point and the global context of the object's shape. The classifier then processes this vector to produce a grasp confidence score:

$$s_{i,j} = \Psi_C(\mathbf{f}_{i,j}; \theta_C) \tag{3}$$

Ψ_C contains a sigmoid activation function to constrain $s_{i,j}$ to the range 0 to 1. This is performed for all $N \times N$ pairs to form the final confidence matrix $\mathbf{S} = \{s_{i,j}\}^{N \times N}$. To prevent a vertex from being paired with itself, a diagonal mask is applied to set $s_{i,i} = 0$ for all i.

C. EdgeGrasper Training and Inference

1) Training: The network is trained end-to-end by minimizing the error between the predicted grasp confidence matrix $\mathbf{S}$ and a ground-truth label matrix $\mathbf{Y} \in \{0, 1\}^{N \times N}$. An entry $s_{i,j}$ should be 1 if the grasp formed by $(\mathbf{v}_i, \mathbf{v}_j)$ is considered valid, and 0 otherwise. The grasp rectangle is defined such that its longer axis coincides and is concurrent with the grasp axis; its length is the grasp width and its width is half the grasp width. The model parameters $(\theta_V, \theta_G, \theta_C)$ are optimized by minimizing the binary cross-entropy (BCE) loss between the predicted and ground-truth matrices. The objective function $\mathcal{L}$ is formulated as:

$$\mathcal{L} = -\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{y_{i,j} \log(s_{i,j}) + (1 - y_{i,j}) \log(1 - s_{i,j})}{N^2}. \tag{4}$$

This loss function encourages the network to output high confidence scores for valid grasp pairs and low scores for invalid ones.

2) Inference: During inference, an input image is input to an instance segmentation model. Optionally, a text prompt describing the target object can be included to condition the segmentation; these can be achieved through a vision-language model (VLM). Additionally, EdgeGrasper can operate as part of a larger perception pipeline. If the segmentation is formatted as a binary mask, it is converted into a polygonal representation using contour detection in OpenCV. The polygon segmentation is subsequently resampled to produce a fixed-size set of N equidistant vertices, $\mathcal{V}$. This vertex set $\mathcal{V}$ is normalized by subtracting the centroid and scaling based on the size of the image. $\mathcal{V}$ then serves as the direct input to our network. EdgeGrasper computes the grasp confidence matrix $\mathbf{S}$, and the final grasp is selected by identifying the vertex pair $(\mathbf{v}_i, \mathbf{v}_j)$ that corresponds to the highest confidence score,

TABLE I

GRASP DETECTION PERFORMANCE ON GRASP-ANYTHING AND JACQUARD DATASETS. H REFERS TO THE HARMONIC MEAN BETWEEN BASE AND NEW.

Method	Grasp-Anything (↑)			Jacquard (↑)		
	Base	New	H	Base	New	H
GR-ConvNet*	0.68	0.55	0.61	0.82	0.61	0.70
Det-Seg-Refine*	0.58	0.53	0.55	0.79	0.55	0.65
GG-CNN*	0.65	0.53	0.58	0.73	0.52	0.61
LGD*	0.69	0.57	0.62	0.83	0.64	0.72
GraspSAM-tiny*	0.78	0.75	0.77	0.90	**0.81**	**0.85**
GraspSAM-t*	**0.83**	**0.81**	**0.82**	0.87	0.75	0.81
EdgeGrasper	0.77	0.78	0.77	**0.91**	0.74	0.82

$\max(s_{i,j})$. Our approach also allows for sampling from the top-k predicted pairs. This enables the generation of diverse grasp candidates for a single object, increasing the robustness of the system in real-world scenarios.

IV. EXPERIMENTS

To train our polygon segmentation-based grasp detection method, we use training data from two existing large-scale datasets: Grasp-Anything and Jacquard. Grasp-Anything [19] is a large-scale grasp dataset generated synthetically using latent diffusion models for high-resolution scenes. The Jacquard dataset [21] is a large-scale synthetic dataset for grasp detection whose objects are adapted from ShapeNet [46]. During the evaluation of quantitative metrics, we use the Base and New subsets as defined in previous work [17], [19] in order to compare against baseline methods. For language-guided grasp detection, we additionally evaluate using the Grasp-Anything++ dataset [20], an extension of Grasp-Anything that includes grasp instructions to refer where in the image a robot should grasp.

We compare our approach against learning-based grasp detection methods GG-CNN [11], GR-CNN [16], Det-Seg-Refine [47], LGD [19], and GraspSAM [17]. Our primary metrics of quantitative evaluation are grasp success rate and inference speed. We follow the definition of grasp success rate / accuracy in previous works [11], [16], [17], [19], [20], where a successful grasp occurs if the grasp intersection over union (IoU) with the ground truth grasp labels is at least 25%, and the orientation error is less than 30°. Meanwhile, inference speed is measured based on the total runtime (in seconds) from an RGB image and/or segmentation polygon input until the complete grasp detection output is returned. All experiments are run on a 8-core Intel CPU machine with 32GB of RAM and an NVIDIA GTX 1080Ti GPU.

TABLE II

CROSS-DATASET GRASP DETECTION PERFORMANCE.

Train Set	Grasp-Anything (↑)		Jacquard (↑)	
Test Set	Grasp-Anything	Jacquard	Jacquard	Grasp-Anything
GR-ConvNet	0.68	0.37	0.82	0.16
GraspSAM	**0.83**	**0.62**	0.87	**0.27**
EdgeGrasper	0.77	0.60	**0.91**	0.24

TABLE III

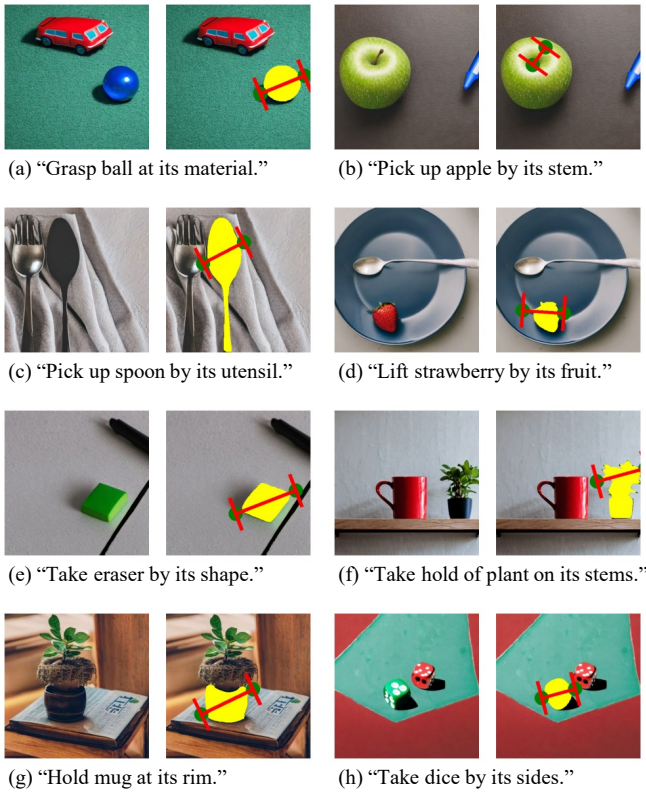
COMPARISON OF NUMBER OF PARAMETERS, FLOPS, AND MODEL INFERENCE TIME AMONG THE GRASP DETECTION METHODS.

Method	Number of Params (↓)	FLOPs (↓)	CPU Inference Time (s) (↓)	GPU Inference Time (s) (↓)
GR-ConvNet	1.90M	11.1G	0.206	0.009
GG-CNN	**0.06M**	**0.3G**	0.013	0.003
GraspSAM-tiny	11.25M	47.9G	1.022	0.094
GraspSAM-t	11.93M	115.0G	3.029	0.139
EdgeGrasper	0.27M	0.5G	**0.008**	**0.002**

For this work, we set the number of vertices $N = 64$; i.e., all polygon segmentations that are input to the model are resampled to exactly 64 vertices. $\Psi_V \cdot$ is implemented as a three-layer/two-layer MLP with dimensions . $\Psi_V(\cdot)$ is implemented as a two-layer MLP with dimension $d = 256$. $\Psi_C(\cdot)$ is implemented as a three-layer MLP with dimensions The neural networks $\Psi_V(\cdot)$, $\Psi_G(\cdot)$, and $\Psi_C(\cdot)$ are implemented as two/three-layer MLPs in PyTorch with dimensions, $d = [128, 128, 256]$, $d = [256, 256]$, and $d = [256, 256, 128]$, respectively. Training is performed for 500 epochs across each dataset with batch size of 512 and we use the Adam optimizer with learning rate of $1e-3$ and weight decay of $1e-4$.

Grasp Detection Results on Jacquard and Grasp-Anything. We present our main grasp detection results in Table I. We find that EdgeGrasper is either the highest or second highest for most of the metrics, which shows that it is competitive with state-of-the-art methods. One observation when comparing the performance of baseline methods on Jacquard and Grasp-Anything datasets is that the Base-to-New gap for all methods is significantly greater for Jacquard than it is for Grasp-Anything, which is the same for EdgeGrasper. Upon analyzing the Jacquard dataset, this might be because of the numerous "valid" grasp annotations of Jacquard which can confuse deep learning models due to the complex contact dynamics of why a grasp is valid. For Grasp-Anything, the baseline methods still have a significant Base-to-New gap, although GraspSAM's results are closer than others. Surprisingly, EdgeGrasper's score for Grasp-Anything New is slightly higher than the score for Grasp-Anything Base. This suggests that training on Grasp-Anything improves generalizability due to the dataset size, and that EdgeGrasper is much stronger for generalizability likely due to its reliance on color- and texture-agnostic shape features.

Cross-Dataset Results on Jacquard and Grasp-Anything. We further show our cross-dataset grasp detection results in Table II. Our results for the Grasp-Anything to Jacquard model is only 2 percentage points lower than GraspSAM, and is significantly higher than GR-ConvNet. Meanwhile, for Jacquard to Grasp-Anything, EdgeGrasper is 3 percentage points lower than GraspSAM but still higher than GR-ConvNet. This might be explained by the distribution shift not just of the object shapes between Grasp-Anything and Jacquard, but also the annotations. But due to EdgeGrasper's dataset-agnostic shape input format, it is plausible to retrain



(a) "Grasp ball at its material." (b) "Pick up apple by its stem." (c) "Pick up spoon by its utensil." (d) "Lift strawberry by its fruit." (e) "Take eraser by its shape." (f) "Take hold of plant on its stems." (g) "Hold mug at its rim." (h) "Take dice by its sides."

Fig. 3. Qualitative results for our language-guided polygon grasp detection. We use the Florence-2-L model to generate polygon segmentations quickly from a text prompt, which we then input to our EdgeGrasper model.

EdgeGrasper on a combined dataset that allows it to gain advantage from the larger overall dataset.

Runtime Analysis and Execution of EdgeGrasper. We discuss EdgeGrasper's runtime execution speed in Table III. The total number of floating point operations (FLOPs) using fvcore [48] while inference speed is computed using an 8-core Intel Xeon Gold CPU (for CPU inference) and an NVIDIA GTX 1080Ti (for GPU inference) by averaging 100 execution time of 100 iterations after 100 warm-up iterations. We find that EdgeGrasper outperforms all baseline methods on all metrics except GG-CNN for number of parameters and FLOPs. Although GG-CNN has lower parameter count and conversely lower FLOPs (as it was engineered for closed-loop robot control), surprisingly EdgeGrasper runs faster on both CPU and GPU likely due to execution differences between MLPs and CNNs. Additionally, EdgeGrasper's grasp detection performance is significantly higher than GG-CNN despite having faster inference. The differences in inference speed are more pronounced against GraspSAM, which has 100x more FLOPs and consequently 120x slower on CPU and 40x slower on GPU. This directly motivates the sparse input and output of EdgeGrasper from polygon vertices for fast grasp detection while still maintaining performance on par with state-of-the-art.

Language-Guided Grasp Detection. In order to use EdgeGrasper as a suitable language-guided grasp detection method, it is advantageous to select a vision-language model that can generate polygon segmentations directly. This provides a better configuration due to the model natively understanding vertices around objects, compared to using models that output mask segmentations before passing it through the polygon segmentation generator (i.e., a two-stage process). To this end, we utilize Florence-2 [33], a strong vision-language foundation model that can also output polygon segmentations with an arbitrary number of vertices. We use Florence-2-L the smaller of the two released variants for our experiments.

Our qualitative results are shown in Fig. 3. Our first insight is the high specificity of EdgeGrasper and Florence-2 for some instructions with precise positions. In Fig. 3b, it is interesting how EdgeGrasper and Florence-2 can process complex queries such as "Pick up apple by its stem" and successfully provide a plausible grasp detection candidate. Second, we see how because of Grasp-Anything++ is a synthetically generated large-scale dataset, some of the grasp instructions do not sound natural, such as Fig. 3a ("grasp ball at its material") and Fig. 3e ("take eraser by its shape"). Nevertheless, our experiments show that EdgeGrasper with Florence-2 is generally able to follow arbitrary grasp instructions and provide plausible grasps.

Fig. 4. We deploy and execute our EdgeGrasper approach on a real-world Franka Research 3 robot arm and select commonly-available household objects as grasp test cases.

TABLE IV

REAL-WORLD GRASPING EXPERIMENTS USING FRANKA ROBOT ARM.

Method	Real-World Success Rate (↑)
GraspSAM	69%
EdgeGrasper	**70%**

Real Robot Grasp Experiments. We deploy EdgeGrasper to a physical Franka Research 3 robot arm [49] controlled by an x86-based machine with an AMD Ryzen CPU and 64GB of RAM. The Franka robot arm operates on a table top with different readily available household objects (Fig. 4). Each scenario has a total of 6 graspable items consisting of 1 target object and cluttered with 5 random objects. At the

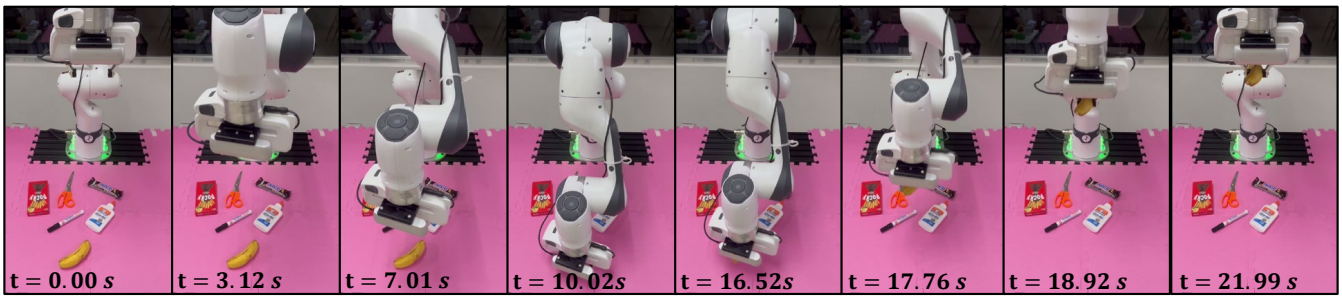


Fig. 5. Qualitative results showing the Franka Research 3 executing EdgeGrasper successfully in the real world on multiple household objects.

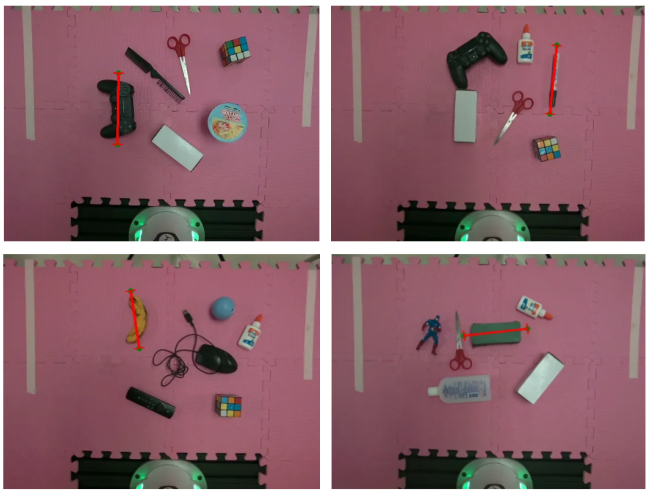


Fig. 6. Failure cases of EdgeGrasper during real-world experiments due to predicted grasp widths that are wider than the robot end-effector limits.

start of each scenario, the we randomize the position and orientation of all objects. The goal is to successfully grasp a target item using the grasp center, width, and angle inference of the model. A grasp is considered successful if the correct item is grasped, picked up, and lifted at least 20 cm from the tabletop without slippage. Five attempts were assigned per target item with 20 target items for a total of 100 attempts. Our quantitative results are shown in Table IV. Our model was able to achieve 70% success rate while the GraspSAM model achieved 69%. We also visualize a sample successful attempt in Fig. 5. Generally, EdgeGrasper's failure cases in the real-world occur due to the grasp predictions requiring grasp widths that are beyond the maximum end-effector limit of the Franka robot arm (Fig. 6) which likely occurs due to a different distribution of item sizes and shapes in the training set. A potential mitigation for future work is to add a regularization penalty on the grasp width during training and execution to encourage feasible grasps considering a specific robot arm's specifications.

V. CONCLUSION

In this paper, we presented EdgeGrasper, a fast polygon segmentation-based grasp detection model that can generate grasp proposals given a list of segmentation polygon vertices. The proposed architecture utilizes a lightweight multilayer perceptron-based network to predict grasp detection confidence across all pairs of points. By leveraging the prior information from the polygon segmentation, EdgeGrasper greatly reduces the list of possible grasp candidates. Our experimental results show that EdgeGrasper is competitive with state-of-the-art grasp detection approaches while having an order of magnitude fewer parameters. We also successfully integrate EdgeGrasper into a language-guided grasp detection pipeline, as well as deploy EdgeGrasper to a real-world Franka Research 3 robot arm. Future work can focus on directly integrating text conditioning to the grasp detection pipeline as well as improving grasp detection for objects with multiple interior edges.

REFERENCES

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, Apr. 2000, pp. 348–353 vol.1.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-Driven Grasp Synthesis—A Survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[3] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sept. 2003, pp. 1824–1829 vol.2.

[4] J. Bohg and D. Kragic, "Learning grasping points with shape context," *Robotics and Autonomous Systems*, vol. 58, no. 4, pp. 362–377, Apr. 2010.

[5] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2308–2315.

[6] D. Prattichizzo, M. Malvezzi, M. Gabiccini, and A. Bicchi, "On the manipulability ellipsoids of underactuated robotic hands with compliance," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 337–346, Mar. 2012.

[7] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, June 2012.

[8] J. Seo, S. Kim, and V. Kumar, "Planar, bimanual, whole-arm grasping," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3271–3277.

[9] C. Rosales, R. Suárez, M. Gabiccini, and A. Bicchi, "On the synthesis of feasible and prehensile robotic grasps," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 550–556.

[10] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio, and K. Goldberg, "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics," in *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, July 2017.

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

[12] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 183–201, 2020.

[13] Y. Wang, Y. Zheng, B. Gao, and D. Huang, "Double-Dot Network for Antipodal Grasp Detection," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2021, pp. 4654–4661.

[14] I. Lenz, H. Lee, and A. Saxena, "Deep Learning for Detecting Robotic Grasps," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[15] F.-J. Chu, R. Xu, and P. A. Vela, "Real-World Multiobject, Multigrasp Detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, Oct. 2018.

[16] S. Kumra, S. Joshi, and F. Sahin, "Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 9626–9633.

[17] S. Noh, J. Kim, D. Nam, S. Back, R. Kang, and K. Lee, "GraspSAM: When Segment Anything Model Meets Grasp Detection," Sept. 2024.

[18] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "SAM 2: Segment Anything in Images and Videos," Oct. 2024.

[19] A. D. Vuong, M. N. Vu, H. Le, B. Huang, H. T. T. Binh, T. Vo, A. Kugi, and A. Nguyen, "Grasp-Anything: Large-scale Grasp Dataset from Foundation Models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 14 030–14 037.

[20] A. D. Vuong, M. N. Vu, B. Huang, N. Nguyen, H. Le, T. Vo, and A. Nguyen, "Language-driven Grasp Detection," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2024, pp. 17 902–17 912.

[21] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A Large Scale Dataset for Robotic Grasp Detection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 3511–3516.

[22] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully Convolutional Instance-Aware Semantic Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 4438–4446.

[23] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting Objects by Locations," in *European Conveference on Computer Vision*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., vol. 12363. Cham: Springer International Publishing, 2020, pp. 649–665.

[24] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and Fast Instance Segmentation," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 17 721–17 732.

[25] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, "PolarMask: Single Shot Instance Segmentation With Polar Representation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 12 190–12 199.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2980–2988.

[27] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment Anything," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 3992–4003.

[28] J. Lazarow, W. Xu, and Z. Tu, "Instance Segmentation with Mask-supervised Polygonal Boundary Transformers," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA: IEEE, June 2022, pp. 4372–4381.

[29] J. Liu, H. Ding, Z. Cai, Y. Zhang, R. Kumar Satzoda, V. Mahadevan, and R. Manmatha, "PolyFormer: Referring Image Segmentation as Sequential Polygon Generation," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Vancouver, BC, Canada: IEEE, June 2023, pp. 18 653–18 663.

[30] Z. Xia, D. Han, Y. Han, X. Pan, S. Song, and G. Huang, "GSVA: Generalized Segmentation via Multimodal Large Language Models," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2024, pp. 3858–3869.

[31] Z. Zhang, Y. Ma, E. Zhang, and X. Bai, "PSALM: Pixelwise SegmentAtion with Large Multi-modal Model," in *European Conference on Computer Vision*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., vol. 15092. Cham: Springer Nature Switzerland, 2024, pp. 74–91.

[32] J. Lu, C. Clark, R. Zellers, R. Mottaghi, and A. Kembhavi, "Unified-IO: A Unified Model for Vision, Language, and Multi-Modal Tasks," in *International Conference on Learning Representations*, 2023.

[33] B. Xiao, H. Wu, W. Xu, X. Dai, H. Hu, Y. Lu, M. Zeng, C. Liu, and L. Yuan, "Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks," in *IEEE/CVF Computer Vision and Pattern Recognition*, 2024.

[34] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual Instruction Tuning," in *Neural Information Processing Systems*, 2023.

[35] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," in *European Conveference on Computer Vision*, 2024.

[36] L. Zhu, T. Chen, D. Ji, J. Ye, and J. Liu, "LLaFS: When Large Language Models Meet Few-Shot Segmentation," in *IEEE/CVF Computer Vision and Pattern Recognition*, 2024.

[37] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, T. Unterthiner, D. Keysers, S. Koppula, F. Liu, A. Grycner, A. Gritsenko, N. Houlsby, M. Kumar, K. Rong, J. Eisenschlos, R. Kabra, M. Bauer, M. Bošnjak, X. Chen, M. Minderer, P. Voigtlaender, I. Bica, I. Balazevic, J. Puigcerver, P. Papalampidi, O. Henaff, X. Xiong, R. Soricut, J. Harmsen, and X. Zhai, "PaliGemma: A versatile 3B VLM for transfer," Oct. 2024.

[38] A. Steiner, A. S. Pinto, M. Tschannen, D. Keysers, X. Wang, Y. Bitton, A. Gritsenko, M. Minderer, A. Sherbondy, S. Long, S. Qin, R. Ingle, E. Bugliarello, S. Kazemzadeh, T. Mesnard, I. Alabdulmohsin, L. Beyer, and X. Zhai, "PaliGemma 2: A Family of Versatile VLMs for Transfer," Dec. 2024.

[39] S. Tong, E. Brown, P. Wu, S. Woo, M. Middepogu, S. C. Akula, J. Yang, S. Yang, A. Iyer, X. Pan, A. Wang, R. Fergus, Y. LeCun, and S. Xie, "Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs," June 2024.

[40] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, C. Liu, M. Liu, Z. Liu, Y. Lu, Y. Shi, L. Wang, J. Wang, B. Xiao, Z. Xiao, J. Yang, M. Zeng, L. Zhou, and P. Zhang, "Florence: A New Foundation Model for Computer Vision," Nov. 2021.

[41] M. Shridhar, L. Manuelli, and D. Fox, "CLIPort: What and Where Pathways for Robotic Manipulation," in *Proceedings of the 5th Conference on Robot Learning*. PMLR, Jan. 2022, pp. 894–906.

[42] K. Xu, S. Zhao, Z. Zhou, Z. Li, H. Pi, Y. Zhu, Y. Wang, and R. Xiong, "A Joint Modeling of Vision-Language-Action for Target-oriented Grasping in Clutter," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 11 597–11 604.

[43] O. Shorinwa, J. Tucker, A. Smith, A. Swann, T. Chen, R. Firoozi, M. K. Iii, and M. Schwager, "Splat-MOVER: Multi-Stage, Open-Vocabulary Robotic Manipulation via Editable Gaussian Splatting," in *Conference on Robot Learning*, 2024.

[44] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, July 2021, pp. 8748–8763.

[45] J. Xu, S. Jin, Y. Lei, Y. Zhang, and L. Zhang, "Reasoning Tuning Grasp: Adapting Multi-Modal Large Language Models for Robotic Grasping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024.

[46] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Dec. 2015.

[47] S. Ainetter and F. Fraundorfer, "End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[48] M. Research, "facebookresearch/fvcore," Sept. 2025, original-date: 2019-09-25T22:27:59Z. [Online]. Available: https://github.com/facebookresearch/fvcore

[49] F. Robotics, "Franka Research 3." [Online]. Available: https://franka.de/franka-research-3