Semantic Grasping Via a Knowledge Graph of Robotic Manipulation: A Graph Representation Learning Approach

Ji Ho Kwak, Jaejun Lee, Joyce Jiyoung Whang, and Sungho Jo, *Senior Member, IEEE*

***Abstract*—Semantic grasping aims to make stable robotic grasps suitable for specific object manipulation tasks. While existing semantic grasping models focus only on the grasping regions of objects based on their affordances, reasoning about which gripper to use for grasping, e.g., a rigid parallel-jaw gripper or a soft gripper, and how strongly to grasp the target object allows more sophisticated robotic manipulation. In this letter, we create a knowledge graph of robotic manipulation named roboKG to represent information about objects (e.g., the material and the components of an object), tasks, and appropriate robotic manipulation such as which component of an object to grasp, which gripper to use, and how strongly to grasp. Using knowledge graph embedding, we generate semantic representations of the entities and relations in roboKG, enabling us to make predictions on robotic manipulation. Based on the predicted gripper type, grasping component, and grasping force, a real robot performs seven different real-world tasks on 42 household objects, achieving an accuracy of 95.21%.**

***Index Terms*—AI-enabled robotics, representation learning, grasping.**

I. Introduction

RELIABLE robotic grasping is the first step of successful robotic manipulation of various objects. To allow robots to grasp objects stably, diverse grasping policies have been considered including ambidextrous robot grasping [1] where heterogeneous grippers are used, a contact-sensing-based grasping [2] where contact feedback is leveraged for robust grasping under uncertainty, and learning grasping policies from synthetic training examples [3]. These existing grasping models focus on objects' physical and geometric properties and propose feasible regions for robots to grasp regardless of the following tasks. However, grasping regions should also be determined depending on the following tasks. For example, if a target task is to open the lid of a bottle, a robot should grasp the lid of the bottle instead of the body. Differentiating the lid and the body requires robots to understand the components of an object that consist of the object. Also, robots should be able to reason about the relationships between the given tasks and the components of the objects to decide where to grasp depending on the tasks.

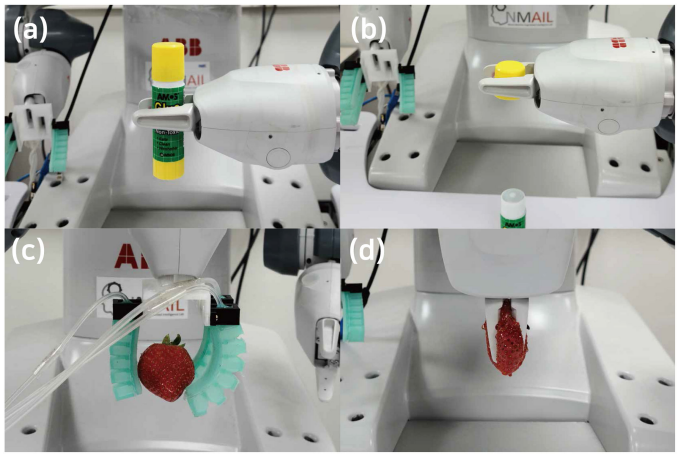


Fig. 1. Semantic grasping that considers which component to grasp, which gripper to use, and how strongly to grasp. (a) & (b): A suitable grasping component depends on the given task, i.e., lifting or opening. (c) & (d): A robot should use a soft gripper to lift a deformable object while a robot should strongly grasp the given object using a rigid gripper for squeezing.

Semantic grasping has been proposed to generate task-dependent grasps that are functionally suitable for specific object manipulation tasks [4]. For example, recent studies in semantic grasping propose to predict the affordances of objects via semantic segmentation [5] or to use a probabilistic logic framework to detect the most likely object component to be grasped [6]. While these methods focus only on the grasping regions of objects, considering other factors such as which gripper to use for grasping and how strongly to grasp benefits more sophisticated robotic manipulation.

We propose semantic grasping that considers three factors: which component a robot should grasp (e.g., lid, body, or handle), which gripper a robot should use (e.g., a rigid parallel-jaw or a soft gripper), how strongly a robot should grasp the target object (e.g., strong, medium, or weak). For example, Fig. 1 shows an example of a robot grasping different components of a

Manuscript received 24 February 2022; accepted 27 June 2022. Date of publication 15 July 2022; date of current version 26 July 2022. This letter was recommended for publication by Associate Editor Y. Bekiroglu and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported in part by IITP, Korean government (MSIT) under Grants 2017-0-00432, 2022-0-00369, and 2020-0-00153 (Penetration Security Testing of ML Model Vulnerabilities and Defense), and in part by the NRF of Korea, Korean Government (MSIT) under Grants 2016R1A5A1938472, 2018R1A5A1059921, and 2022R1A2C4001594 [DOI: 10.1109/LRA.2022.3191194]. *(Ji Ho Kwak and Jaejun Lee equally contributed to this work.) (Corresponding authors: Joyce Jiyoung Whang; Sungho Jo.)*

The authors are with the School of Computing, KAIST, Daejeon 305-701, Republic of Korea (e-mail: jim9611@kaist.ac.kr; jjlee98@kaist.ac.kr; jjwhang@kaist.ac.kr; shjo@kaist.ac.kr).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3191194, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3191194

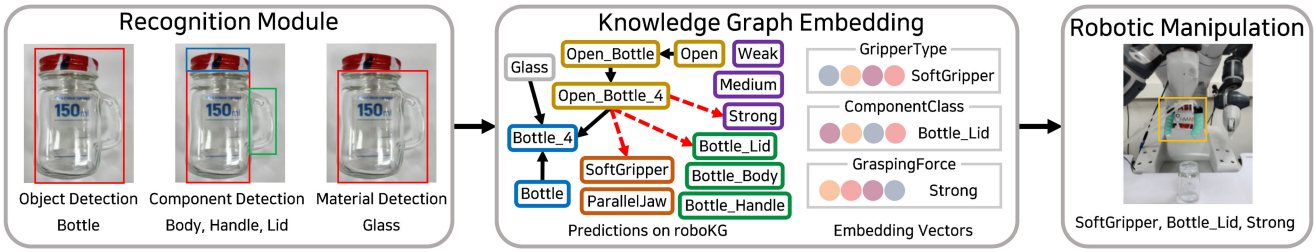


Fig. 2. Our method consists of the recognition module (including object detection, component detection, and material detection), knowledge graph embedding and predictions on *roboKG*, and robotic manipulation based on the predictions.

glue stick depending on tasks, i.e., grasping the body for lifting and grasping the lid for opening ((a) & (b)). Also, a robot should use different gripper types with appropriate force, e.g., a robot should use a soft gripper to lift a strawberry and use a rigid parallel-jaw gripper with a strong force to squeeze it ((c) & (d)).

We create a knowledge graph of robotic manipulation named *roboKG* that represents object-related information (e.g., components of an object and the material of an object), task-related information (e.g., the task hierarchy), and manipulation-related information (e.g., a gripper type and a grasping force). Based on *roboKG* representing desirable grasping strategies for diverse objects and tasks, we compute the embedding vectors of the entities and relations in *roboKG* by knowledge graph embedding [7]. Using the computed embedding vectors, we can predict the appropriate robotic manipulation factors. We use hierarchical knowledge graph embedding methods [8]–[10] that generate embedding vectors by reflecting not only the structure of *roboKG* but also the hierarchies of the entities. Fig. 2 shows an overview of our method consisting of the recognition module, predictions on *roboKG* using knowledge graph embedding, and robotic manipulation based on the predictions. Given an object, a recognition model identifies what the object is, which components the object has, and which material the object is made of. Each of these factors affect the suitable grasping strategies. Given a specific object and a specific task, we predict which component to grasp, which gripper to use, and how strongly to grasp based on the embedding vectors of our *roboKG*. Based on these predictions, a real robot performs the given task on the target object. Using the dual-arm robot IRB 14000 Yumi from ABB, our method achieves an accuracy of 95.21% on 188 grasps consisting of seven different real-world tasks on 42 household objects. Our contributions include a novel representation of robotic manipulation using *roboKG* and knowledge representations using knowledge graph embedding to predict the desired robotic manipulation factors.

II. Related Work

Different semantic grasping strategies have been proposed for task-oriented grasping [5], [6], [11]. For example, CAGE [12] is a context-aware grasping engine that considers affordance, material, object status, and task constraints. A grasp planner considers stability, tactile contacts, and hand kinematics by using a semantic affordance map [4]. Even though some recent grasping models use a knowledge graph [13], [14], it is quite different from our *roboKG*, which is firstly introduced in this letter. More importantly, these existing semantic grasping methods only focus on detecting grasping regions of objects and do not consider other factors such as gripper types and grasping forces as we do.

In robotics, the knowledge representation and reasoning mechanisms have been studied in various contexts with different types of knowledge bases. For example, KnowRob 2.0 [15] has been proposed to represent a general ontology covering a broad range of human manipulation knowledge about motor cognition and robot control. In some scenarios such as making pancakes, robots can retrieve instructions from the World Wide Web and exploit the predefined ontology to semi-automatically translate it into a robot plan [16]. Recently, ontology-based knowledge representation has shown great successes in purposive learning [17], e.g., inferring human activities from complex observations by defining the semantic rules using a knowledge base [18].

A knowledge graph is a knowledge base in the form of a graph consisting of entities and relations. While a graph is a discrete data structure, representing nodes and edges in a graph in continuous feature space is beneficial for solving diverse machine learning and data mining problems such as link prediction and node classification [19]. For this, many different graph representation learning methods have been proposed [20]. Knowledge graph embedding aims to represent entities and relations in a knowledge graph as low-dimensional feature vectors by preserving the structure of the given knowledge graph. Among a number of knowledge graph embedding methods [7], some methods take into account a hierarchical structure between entities [8]–[10], [21]. We use HAKE [8], MuRP [9], and ConE [10] in our experiments because these methods can be applied to our *roboKG* with no extra constraints.

Knowledge graph embedding has been considered in robotics very recently with diverse applications. RoboCSE [22] generalizes semantic knowledge about object affordances, locations, and materials even though it is not tested on object manipulation of a real robot. In the context of one-shot task execution, knowledge graph embedding has been explored to integrate task plans with domain knowledge for task generalization [23]. Also, the incremental knowledge graph embedding problem has been studied for continual learning [24]. Different from these approaches, we define a new knowledge graph, *roboKG*, which is

used to predict the factors required for the robotic manipulation of a real robot using knowledge graph embedding.

III. Recognition Module

Given an object, a robot should recognize what the object is, what components consist of the object, and what material the object is made of. We apply the existing object detection, part detection, and material recognition methods for this recognition module. Our robot is equipped with an RGB-D camera on its body to inspect objects from a diagonal view; we use images and point clouds of objects for the recognition module. We consider 94 different household objects, including food items, kitchen items, tool items, shape items [25], etc. In order to get a list of common household objects, we consider the YCB dataset [25], GMU Kitchen dataset [26] and RGB-D object dataset [27]. Since the same object can be differently named on different datasets, we convert the object names based on ConceptNet [28]. For object detection, we use the YOLOv5 model [29], [30] pre-trained with the MS-COCO dataset [31]. Given an image, the YOLOv5 model returns the object label. Once an object label is determined, we label each component of the object (e.g., body, lid, or handle) using PartNet [32] which provides labels of each component by semantic part detection using point clouds. The material types are chosen from the MINC dataset [33], one of the standard datasets for material recognition. For material recognition, we use the DEP model [34] pre-trained with the MINC2500 [33] dataset.

IV. Knowledge Graph Embedding of Robotic Manipulation

We create *roboKG*, a knowledge graph of robotic manipulation to represent human knowledge about desirable robotic grasping for various tasks. Using knowledge graph embedding methodologies, we convert the factors of robotic manipulation into semantic representation vectors. We predict the appropriate factors for successful robotic manipulation of different tasks using the feature space representations.

A. roboKG: *Knowledge Graph of Robotic Manipulation*

Given an object and a task a robot should perform (e.g., lifting a bottle or pushing a cup forward, etc.), a robot should decide where to grasp, which gripper to use, and how strongly to grasp the given object. These factors can be decided based on what the object is, which components the object has, which material the object is made of, and which task a robot should perform using the object. For humans, given a specific object and a specific task, we can designate the desirable factors to accomplish the task.

A knowledge graph represents human knowledge, where each known fact is represented by a triplet consisting of a head entity, a relation, and a tail entity. Formally, a knowledge graph is defined as $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ where $\mathcal{E} = \{e_1, e_2, \cdots, e_{|\mathcal{E}|}\}$ is a set of entities, $\mathcal{R} = \{r_1, r_2, \cdots, r_{|\mathcal{R}|}\}$ is a set of relations, and $\mathcal{T} = \{(e_i, r_k, e_j) | e_i \in \mathcal{E}, r_k \in \mathcal{R}, e_j \in \mathcal{E}\}$ is a set of triplets. Once human knowledge is organized using a knowledge graph, we can apply a graph representation learning technique to convert the entities and relations in a knowledge graph into representations in continuous feature space; this conversion allows us to make predictions on missing links in the knowledge graph.

TABLE I
Entity Types in *roboKG*

Entity Type	Example	Freq.
ObjectClass	`Bottle`, `Apple`, ⋯	94
ObjectInstance	`Bottle_1`, ⋯	153
MaterialClass	`Ceramic`, `Fabric`, `Food`, ⋯	11
ComponentCategory	`Body`, `Handle`, `Lid`	3
ComponentClass	`Bottle_Body`, `Bottle_Handle`, ⋯	126
TaskCategory	`Grasp`, `Lift`, `Open`, ⋯	7
TaskClass	`Grasp_Bottle`, `Open_Bottle`, ⋯	422
TaskInstance	`Open_Bottle_1`, ⋯	703
GripperType	`ParallelJaw`, `SoftGripper`	2
GraspingForce	`Weak`, `Medium`, `Strong`	3

We create *roboKG*, a novel knowledge graph representing human knowledge about desirable robotic manipulation for various tasks on various objects. We define ten different entity types, ten different relations, and ten different triplet types in *roboKG* as shown in Table I and Table II where Freq. indicates the frequency. Our *roboKG* contains 1,524 entities, 10 relations, and 4,588 triplets. To represent objects, we define an entity type called ObjectClass (e.g., `Bottle`, `Apple`, `Bag`, `Jar`, `Ball`, `Screwdriver`, etc.).

Robotic grasping strategies can be changed depending on which material an object is made of and which components an object has. Therefore, we specify objects based on their materials and their components. For example, if there are two bottles where the first bottle is made of plastic and has a lid and the second bottle is made of glass and does not have a lid, then we differently name these bottles, e.g., we call the former `Bottle_1` and the latter `Bottle_2`. The ObjectInstance represents an instance of an object with a specific material and specific components. For each object, the number of entities of ObjectInstance is related to the combinations of materials and components of the objects. The relation `InstantiateObject` connects ObjectClass and ObjectInstance. For example, to indicate `Bottle_4` is a bottle, we create a triplet such as (`Bottle`, `InstantiateObject`, `Bottle_4`). We consider 11 different materials: `Ceramic`, `Fabric`, `Food`, `Glass`, `Leather`, `Metal`, `Paper`, `Plastic`, `Rubber`, `Stone`, and `Wood`. The MaterialClass represents these materials. To map a material to a specific object, we define the relation `Make` and triplets such as (`Glass`, `Make`, `Bottle_4`). We consider three possible components of an object: `Body`, `Handle`, and `Lid` represented by ComponentCategory. While all objects have the body by default, the handle or lid can be missing depending on the objects. To represent a component of a specific object, we define ComponentClass, e.g., `Bottle_Body` and `Bottle_Lid`. To connect ComponentCategory and ComponentClass, we define the relation `SpecifyComponent` and triplets such as (`Lid`, `SpecifyComponent`, `Bottle_Lid`). To represent which object instance has which components, we define the relation `IsComponentOf` and triplets such as (`Bottle_Lid`, `IsComponentOf`, `Bottle_4`).

TABLE II
RELATIONS AND TRIPLET TYPES IN *roboKG*

Relation	Triplet Type	Example	Freq.
InstantiateObject	(ObjectClass, InstantiateObject, ObjectInstance)	(Bottle, InstantiateObject, Bottle_4)	153
Make	(MaterialClass, Make, ObjectInstance)	(Glass, Make, Bottle_4)	153
SpecifyComponent	(ComponentCategory, SpecifyComponent, ComponentClass)	(Lid, SpecifyComponent, Bottle_Lid)	126
IsComponentOf	(ComponentClass, IsComponentOf, ObjectInstance)	(Bottle_Lid, IsComponentOf, Bottle_4)	219
SpecifyTask	(TaskCategory, SpecifyTask, TaskClass)	(Open, SpecifyTask, Open_Bottle)	422
InstantiateTask	(TaskClass, InstantiateTask, TaskInstance)	(Open_Bottle, InstantiateTask, Open_Bottle_4)	703
Include	(TaskInstance, Include, ObjectInstance)	(Open_Bottle_4, Include, Bottle_4)	703
WhichGripper	(TaskInstance, WhichGripper, GripperType)	(Open_Bottle_4, WhichGripper, SoftGripper)	703
WhichForce	(TaskInstance, WhichForce, GraspingForce)	(Open_Bottle_4, WhichForce, Strong)	703
WhichComponent	(TaskInstance, WhichComponent, ComponentClass)	(Open_Bottle_4, WhichComponent, Bottle_Lid)	703

We consider seven different tasks from [35]: Grasp, Lift, Open, Pour, Push, Rotate, and Squeeze represented by TaskCategory. To differentiate each of these tasks based on a target object, we define TaskClass that represents which task is performed on which object, e.g., Push_Bottle, Squeeze_Sponge, etc. To narrow down TaskCategory to TaskClass, we define the relation SpecifyTask and triplets such as (Open, SpecifyTask, Open_Bottle). Depending on a specific object, possible tasks are differently determined. For example, we can open a bottle's lid only if the bottle has a lid. Thus, we define TaskInstance to map each task to a specific object, e.g., Open_Bottle_4 and Open_Jar_3. To narrow down TaskClass to TaskInstance, we define the relation InstantiateTask and triplets such as (Open_Bottle, InstantiateTask, Open_Bottle_4). Also, each TaskInstance should be connected to ObjectInstance to represent which task includes which object. To represent this, we define the relation Include and triplets such as (Open_Bottle_4, Include, Bottle_4).

Finally, we define three special relations that are directly related to robotic manipulation: WhichGripper, WhichForce, and WhichComponent. Given a specific object and a specific task, we represent which gripper a robot should use, how strongly a robot should grasp the object, and which component a robot should grasp. In our experiments, we have two different types of grippers: a rigid parallel-jaw gripper (denoted by ParallelJaw) and a soft gripper (denoted by SoftGripper). Also, we define three different forces for GraspingForce in a discrete manner: Weak, Medium, and Strong. For each TaskInstance, we designate a desirable gripper type, grasping force, and a grasping component. For example, we create triplets such as (Open_Bottle_4, WhichGripper, SoftGripper), (Open_Bottle_4, WhichForce, Strong), and (Open_Bottle_4, WhichComponent, Bottle_Lid).

B. Semantic Representation of Robotic Manipulation

Given a knowledge graph, knowledge graph embedding aims to represent the entities and relations in continuous feature space while preserving the structure of a knowledge graph [7]. Once the entities and relations are represented as feature vectors, the problem of predicting a missing link in a knowledge graph becomes more tractable. Let $\mathbf{e}_i \in \mathbb{R}^{d_e}$ denote an embedding vector of an entity e_i and $\mathbf{r}_k \in \mathbb{R}^{d_r}$ denote an embedding vector of a relation r_k where d_e and d_r are dimensions of an entity embedding vector and a relation embedding vector, respectively. A knowledge graph embedding method defines a scoring function of a triplet (e_i, r_k, e_j), denoted by $f(\mathbf{e}_i, \mathbf{r}_k, \mathbf{e}_j)$, using the embedding vectors of the corresponding entities and relations such that a more plausible triplet receives a higher score. For example, TransE [36], defines $f(\mathbf{e}_i, \mathbf{r}_k, \mathbf{e}_j) := -||\mathbf{e}_i + \mathbf{r}_k - \mathbf{e}_j||$. Then, TransE minimizes a margin-based loss which is defined by

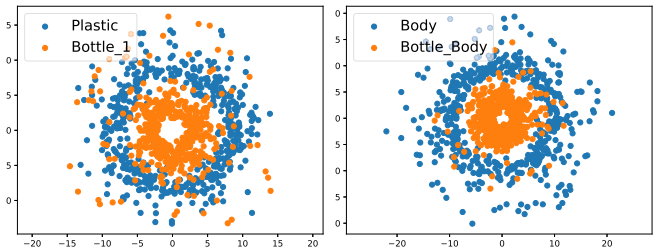
$$\sum_{(e_i,r_k,e_j)\in\mathcal{T}_{tr}} \sum_{(e'_i,r_k,e'_j)\in\mathcal{T}_{tr}{}'} [\gamma - f(\mathbf{e}_i, \mathbf{r}_k, \mathbf{e}_j) + f(\mathbf{e}'_i, \mathbf{r}_k, \mathbf{e}'_j)]_+$$

where γ is a margin, $[z]_+ := \max(0, z)$, $\mathcal{T}_{tr}$ is a set of triplets in the training set, $\mathcal{T}_{tr}{}'$ is a set of corrupted triplets where (e'_i, r_k, e'_j) indicates a corrupted triplet. Given $(e_i, r_k, e_j) \in \mathcal{T}_{tr}$, we randomly replace e_i or e_j with an entity e'_i or e'_j in $\mathcal{E}$ to make a corrupted triplet. By minimizing the above loss function, we learn embedding vectors to encourage the patterns of the triplets observed in the training set and discourage the patterns of the corrupted triplets. The resulting entity and relation embedding vectors naturally reflect the structure of the given knowledge graph, encoding the interactions between the entities and relations.

When we consider the structure of *roboKG*, we see that there exist some hierarchies between entity types: MaterialClass $\rightarrow$ ObjectInstance, ComponentCategory $\rightarrow$ ComponentClass, ObjectClass $\rightarrow$ ObjectInstance, and TaskCategory $\rightarrow$ TaskClass $\rightarrow$ TaskInstance. To appropriately incorporate these entity hierarchies into knowledge graph embedding, we use three different recently proposed knowledge graph embedding methods: HAKE [8], MuRP [9], and ConE [10]. For example, HAKE maps entities into the polar coordinate system such that the radial coordinate represents entities at different levels of the hierarchy and the angular coordinate distinguishes entities at the same level of the hierarchy. In HAKE, the embedding vectors are decomposed into two different parts: the modulus part and the phase part. The scoring function of HAKE is defined by

$$f(\mathbf{e}_i, \mathbf{r}_k, \mathbf{e}_j) := -||\mathbf{e}_i^m \circ \mathbf{r}_k^m - \mathbf{e}_j^m||_2 - \lambda||\sin((\mathbf{e}_i^p + \mathbf{r}_k^p - \mathbf{e}_j^p)/2)||_1$$

where $\mathbf{x}^m$ is the modulus part of a vector $\mathbf{x}$, $\mathbf{x}^p$ is the phase part of $\mathbf{x}$, $\mathbf{x}$ can be $\mathbf{e}_i$, $\mathbf{r}_k$, or $\mathbf{e}_j$, $\circ$ is the hadamard product, and $\lambda \in \mathbb{R}$ is a model parameter. Based on this scoring function, the loss of HAKE is defined with self-adversarial training adapted from [37]. The other two methods, MuRP and ConE, consider the hyperbolic embedding space to model hierarchies. Using



(a) MaterialClass `Plastic`, ObjectInstance `Bottle_1` (b) ComponentCategory `Body`, ComponentClass `Bottle_Body`

Fig. 3. Entity Embedding Vectors of HAKE. Embedding vectors of a higher hierarchy entity type are represented as outer rings, whereas embedding vectors of a lower hierarchy entity type are represented as inner rings.

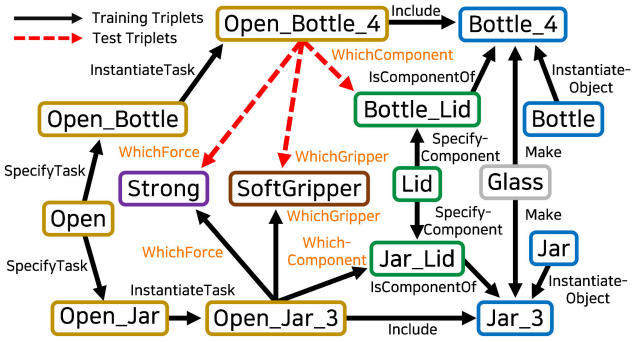


Fig. 4. Training Triplets and Test Triplets of *roboKG*.

HAKE, MuRP, and ConE, we learn embedding vectors of entities and relations in *roboKG*. For example, Fig. 3 shows entity embedding vectors generated by HAKE. In HAKE, entities at different hierarchy levels are separated by the radial coordinate. We visualize two different hierarchical entity structures: (a) MaterialClass: `Plastic` $\rightarrow$ ObjectInstance: `Bottle_1` and (b) ComponentCategory: `Body` $\rightarrow$ ComponentClass: `Bottle_Body`. Using HAKE, the entity embedding vectors of a higher hierarchy entity type are represented as outer rings, whereas the entity embedding vectors of a lower hierarchy entity type are represented as inner rings. Once we represent the entities and relations as feature vectors, we can predict missing links in a knowledge graph, which allows automated reasoning about appropriate actions to accomplish specific tasks.

C. Prediction Using Knowledge Graph Embedding

Given *roboKG*, we learn embedding vectors of the entities and relations using a training set of triplets. At test time, we predict an appropriate gripper, force, and grasping component for a task instance that is not observed during training. Let $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ denote *roboKG*. Recall that we have three special relations: `WhichGripper`, `WhichForce`, and `WhichComponent`. Let us call these three special relations querying relations. Given a specific object and a specific task, denoted by the TaskInstance entity type, e.g., `Open_Bottle_4`, our problem can be thought of as the problem of predicting tail entities of (`Open_Bottle_4`, `WhichGripper`, `?`), (`Open_Bottle_4`, `WhichForce`, `?`), and (`Open_Bottle_4`, `WhichComponent`, `?`). For an entity t of TaskInstance, let $\mathcal{S}_t$ denote the triplets containing the three querying relations for t, i.e., $\mathcal{S}_t := \{(t, r_k, e_j) | r_k \in \{\texttt{WhichGripper}, \texttt{WhichForce}, \texttt{WhichComponent}\}, e_j \in \mathcal{E}\}$. Also, let $\mathcal{S}$ denote all triplets containing the querying relations, i.e., $\mathcal{S} := \cup_t \mathcal{S}_t, t \in \mathcal{E}$. To appropriately train a knowledge graph embedding method, we randomly split $\mathcal{S}$ into the training, validation, and test sets with the ratio of 8:1:1 such that $\mathcal{S}_t$ belongs to the same set for each t. The validation set is used to tune the hyperparameters of the methods. Note that $\mathcal{S} \subset \mathcal{T}$, and the triplets in $\mathcal{T} \backslash \mathcal{S}$ represent information about the properties of objects, e.g., (`Glass`, `Make`, `Bottle_4`) or the task hierarchy, e.g., (`Open`, `SpecifyTask`, `Open_Bottle`). We assign the triplets in $\mathcal{T} \backslash \mathcal{S}$ to the training set.

After training a knowledge graph embedding method using the training set, we test whether we can correctly predict a gripper type, a grasping force, and a grasping component for each TaskInstance at the test set. Let $\hat{\mathbf{t}}$ denote an embedding vector of an entity $\hat{t}$ in the test set, and $\hat{\mathbf{r}}$ denote an embedding vector of `WhichGripper`. Also, let $\mathbf{x}_1$ and $\mathbf{x}_2$ denote embedding vectors of `ParallelJaw` and `SoftGripper`, respectively. Then, to solve ($\hat{t}$, `WhichGripper`, `?`), we compare the scores of ($\hat{t}$, `WhichGripper`, `ParallelJaw`) and ($\hat{t}$, `WhichGripper`, `SoftGripper`) using the scoring function described in Section IV-B. If $f(\hat{\mathbf{t}}, \hat{\mathbf{r}}, \mathbf{x}_1) < f(\hat{\mathbf{t}}, \hat{\mathbf{r}}, \mathbf{x}_2)$, we conclude that a robot should use a soft gripper for the task $\hat{t}$. Similarly, we can also make predictions on `WhichForce` and `WhichComponent`.

Fig. 4 shows a subgraph of *roboKG* where the solid arrows indicate the triplets in the training set and the dotted arrows indicate the triplets in the test set. In this example, our goal is to predict appropriate factors for `Open_Bottle_4`. We see that both `Bottle_4` and `Jar_3` are made of `Glass`, and have a `Lid`. Due to these shared properties, a knowledge graph embedding method attempts to place the embedding vectors of `Bottle_4` and `Jar_3` closely. Also, since `Open_Bottle_4` and `Open_Jar_3` share `Open` in their entity hierarchies, one can infer that a robot can open `Bottle_4` just as the robot opens `Jar_3`. A knowledge graph embedding method learns embedding vectors by reflecting the structure of a given knowledge graph, implying all these complicated connections between entities. Therefore, we can predict an appropriate gripper type, grasping force, and grasping component for a specific task at test time using the learned embedding vectors.

V. Robotic Manipulation Via Semantic Grasping

We use a dual-arm robot IRB 14000 Yumi from ABB, which has 7-DoF on each arm. To control the robot, we use the source codes developed by UC Berkeley Automation Lab [38]. Each hand is equipped with a different type of gripper: a rigid parallel-jaw gripper on the left hand and a soft gripper [39] on the right hand. An RGB-D camera is attached on the top to inspect objects in the workspace. Based on the predictions on `GripperType`, `ComponentClass`, `GraspingForce` for each `TaskInstance`, the robot manipulates the target object. For each component of an object, the exact coordinates

and orientations are predefined to the robot system. Recall that `GraspingForce` is controlled with three different levels: `Weak`, `Medium`, and `Strong`. We set the grasping force of the parallel-jaw gripper as 10 N, 15 N, 20 N, and the gauge pressure inside the soft gripper as 0.8 MPa, 1.0 MPa, 1.2 MPa, each corresponding to the three different levels, respectively.

VI. Experiments

We conduct three different types of experiments. First, we measure the accuracy of knowledge graph embedding methods to see whether *roboKG* is appropriately created, and a knowledge graph embedding method successfully predicts a missing entity at test time. Second, we operate a real robot based on the predictions made by knowledge graph embedding methods. Third, we test an end-to-end model where the objects, components, and materials are automatically identified by the recognition module.

A. Knowledge Graph Embedding for Semantic Grasping

Given a specific object and a specific task, we test how accurately a knowledge graph embedding method can predict a suitable gripper type, grasping force, and grasping component. As discussed in Section IV-B, we use three recently proposed knowledge graph embedding methods: HAKE [8], MuRP [9], and ConE [10]. We compare these methods with three baseline methods: Naïve Bayes Classifier (denoted by NBC) [40], random prediction (denoted by RD), and distribution-based random prediction (denoted by DRD). Let us briefly describe how we apply a naïve Bayes classifier [40]. To decide which gripper to use for `Open_Bottle_4`, for example, we compare the probabilities of $p(\texttt{ParallelJaw}|\texttt{Open\_Bottle})$ and $p(\texttt{SoftGripper}|\texttt{Open\_Bottle})$. Using Bayes' theorem, $p(\texttt{ParallelJaw}|\texttt{Open\_Bottle})$ can be computed by

$$\frac{p(\texttt{Open\_Bottle}|\texttt{ParallelJaw})p(\texttt{ParallelJaw})}{p(\texttt{Open\_Bottle})}$$

where $p(\texttt{Open\_Bottle}|\texttt{ParallelJaw})$ is computed by

$$\frac{N(\texttt{Open\_Bottle\&ParallelJaw}) + mp}{N(\texttt{ParallelJaw}) + m}$$

where $N(\texttt{Open\_Bottle\&ParallelJaw})$ is the number of triplets of (`Open_Bottle_?`, `WhichGripper`, `ParallelJaw`), $N(\texttt{ParallelJaw})$ is the number of triplets of (`?`, `WhichGripper`, `ParallelJaw`), p is a prior probability, and m is a hyperparameter of m-estimate. Similarly, we can also compute the probability of $p(\texttt{SoftGripper}|\texttt{Open\_Bottle})$, and predict a gripper type of a higher probability. For random prediction, we randomly predict a gripper type, and for a distribution-based random prediction, we consider the underlying distribution to predict a gripper type, e.g., if 40% of the tasks use a soft gripper, the chance we select a soft gripper is 0.4. We can predict a grasping component and grasping force similarly. Table III shows the search space of the hyperparameters of each method. The best hyperparameters are bold faced. We select the best hyperparameters on the validation set and use them at test time. We use the notations in the original source codes of the methods except d_e and d_r, which denote the dimensions of entity and relation embedding vectors, respectively. We use the Adam optimizer for HAKE and ConE, and use the Riemannian SGD for MuRP. We use the batch size of 512, 128, and 1024 for HAKE, MuRP, and ConE, respectively. Table IV shows the average accuracy of each method using 10-fold cross validation. The last row of Table IV indicates how accurately a method predicts the gripper type, grasping force, and grasping component simultaneously. We see that the knowledge graph embedding methods, HAKE, MuRP, and ConE, significantly outperform other baseline methods, showing approximately 90% accuracy.

TABLE III
Hyperparameters of Each Prediction Model

Model	Hyperparameters
HAKE	$lr \in \{\mathbf{0.00005}, 0.0001\}, \gamma \in \{6, \mathbf{9}\}, a \in \{0.5, \mathbf{1}\}$, $mw \in \{\mathbf{0.5}, 1\}, pw \in \{\mathbf{0.5}, 1\}, d_e = 1,000, d_r = 1,500$
MuRP	$lr \in \{1, 5, 10, 20, \mathbf{50}, 100\}$, $d \in \{\mathbf{40}, 200, 500\}, d_e = d_r = d$
ConE	$lr \in \{0.0001, \mathbf{0.001}\}, att \in \{50, \mathbf{100}\}$, $w \in \{0.1, \mathbf{0.3}, 0.5, 0.7\}, d_e = 1,000, d_r = 500$
NBC	$m \in \{0, 1, 2, \mathbf{3}, 4, 5, 6, 7, 8, 9, 10\}$, $p \in \{0.1, 0.2, 0.3, 0.4, \mathbf{0.5}, 0.6, 0.7, 0.8, 0.9, 1\}$

TABLE IV
Accuracy of 10-Fold Cross-Validation

Relation Type	HAKE	MuRP	ConE	NBC	RD	DRD
WhichGripper (G)	**0.959**	0.957	0.952	0.657	0.505	0.539
WhichForce (F)	**0.976**	**0.976**	0.963	0.666	0.341	0.832
WhichComponent (C)	**0.967**	0.940	0.953	0.753	0.720	0.780
Which G & F & C	**0.908**	0.888	0.875	0.371	0.118	0.320

Fig. 5. Real-world objects (ObjectInstance) used in our experiments.

B. Robotic Manipulation Without Recognition Module

We operate a real robot based on the predictions made on *roboKG*. We use 42 objects (ObjectInstance) of 34 different ObjectClass shown in Fig. 5. Seven different tasks are considered resulting in 188 TaskInstance in total. Table V shows the tasks and the criteria of successful trials[1]. Recall that our recognition

[1]Our demo video about the tasks is available on https://youtu.be/BXOmLaGSHEc.

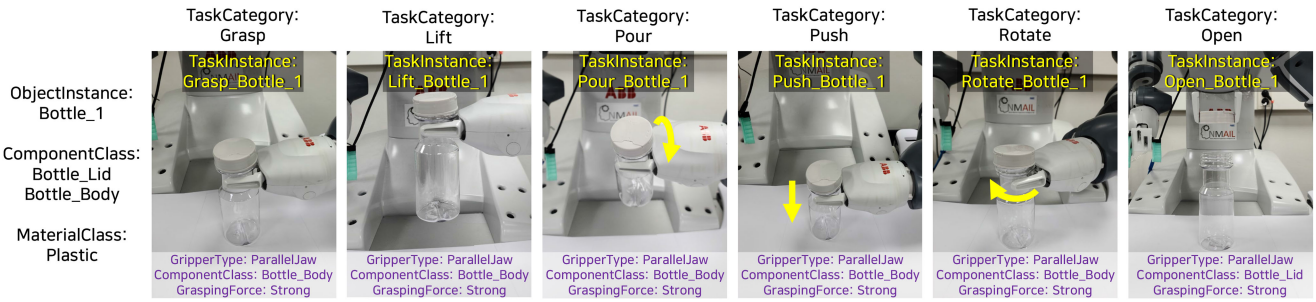


Fig. 6. Robotic Manipulation with the End-to-End Model. Examples of TaskInstance are shown.

TABLE V
CONSIDERED TASKS AND CRITERIA OF SUCCESSFUL TRIALS

Task	Criteria of Successful Trials
Open	whether a robot separates the body and the lid
Pour	whether a robot tilts 30 degrees after lifting the object
Rotate	whether a robot rotates the object for 30 degrees on the ground
Lift	whether a robot lifts the object 10cm off the ground
Push	whether a robot pushes the object forward for 10cm
Squeeze	whether a robot grasps the object and changes the shape of the object
Grasp	whether a robot grasps the designated component of the object without changing the shape of the object

TABLE VI
ACCURACY OF ROBOTIC MANIPULATION

	HAKE	MuRP	ConE	Rule	Centroid
No. of TaskInstance	188	188	188	188	188
No. of Successes	**179**	177	178	170	171
Accuracy	**95.21%**	94.15%	94.68%	90.43%	90.96%

module includes an object detection, a component detection, and a material detection model, each of which is pre-trained on the standard datasets as described in Section III. In practice, it is hard to simultaneously get the correct answers from all three detection methods for an arbitrary object using the current technologies. Therefore, in this experiment, we assume that the object labels, their components, and their materials are given. Table VI shows the accuracy of robotic manipulation with different knowledge graph embedding methods, HAKE, MuRP, and ConE and the two baseline methods: the rule-based model (denoted by Rule) and the centroid-based grasping (denoted by Centroid). For the rule-based model, we specify the following rules: (i) the robot uses `SoftGripper` for the objects made of `Ceramic`, `Food`, and `Glass`, and uses `ParallelJaw` for the other objects. (ii) the robot grasps `Lid` when the task is `Open`, grasps `Handle` if the object has a handle, and grasps `Body` for the other cases. (iii) the GraspingForce is set to be `Strong` if the task is `Squeeze` and `Medium` for the other tasks. For the centroid-based grasping, we fix the GripperType to be `ParallelJaw` and the GraspingForce to be `Medium`. Then, the robot grasps the centroid of a given object with additional consideration about the object's width and the gripper's width. In Table VI, we see that the three knowledge graph embedding methods outperform the rule-based model and the centroid-based grasping. In particular, these two baseline methods fail to lift a table tennis ball because both of the baseline methods use `ParallelJaw` instead of `SoftGripper`. Since the surface of a table tennis ball is slippery and has a curvature, the robot fails to appropriately lift it using `ParallelJaw`. On the other hand, the hierarchical knowledge graph embedding methods, HAKE, MuRP, and ConE, successfully predict the GripperType to be `SoftGripper` for this task because the embedding vectors are trained using a hierarchy that a table tennis ball is an instance of `Ball`, and the vectors are learned to encode that the desired GripperType for the ball-related tasks is `SoftGripper`. Using *roboKG* and its embedding, we can operate a real robot more sophisticatedly. We see that the accuracy of robotic manipulation is higher than the accuracy of a knowledge graph embedding method itself. For example, the accuracy of HAKE itself is 90.8% shown in Table IV whereas the accuracy of our robot operated based on the predictions made by HAKE is 95.21%. This is mainly because there are some cases where a robot succeeds in accomplishing a task using a different grasping component from a preferred component represented in *roboKG*. For example, when lifting an object with a handle, we define the handle to be the appropriate grasping component. Thus, if HAKE predicts the grasping component to be the body instead of the handle, the prediction is considered to be incorrect. But, in practice, our robot succeeds in lifting the object by grasping the body.

C. Robotic Manipulation With Recognition Module

We test the performance of our end-to-end model that includes the recognition module. Among the objects we have, 11 ObjectInstance are supposed to be correctly recognized by the recognition module, i.e., their object labels, components, and materials can be correctly identified using the methods in Section III. However, in practice, the YOLOv5 [29] method returned wrong answers for two objects, PartNet [32] returned wrong answers for one object, and DEP [34] returned wrong answers for three objects. Since our prediction procedure at least requires correct identification of an object, we conducted the robotic manipulation experiments with the nine object instances by excluding the two wrongly identified object instances by YOLOv5. For the nine ObjectInstance, we have 43 TaskInstance. Fig. 6 shows some example TaskInstance. Based on the predictions by HAKE, the robot succeeded in 34 out of 43

TaskInstance, resulting in 79.07% accuracy. The failures were mainly due to a wrong part detection by PartNet or a wrong material detection by DEP. For example, if an object instance's material is wrongly detected, the target ObjectInstance is mapped into a different one, returning possibly wrong manipulation factors. However, we also observed that sometimes the returned manipulation factors happened to be very similar to the correct ones; in this case, the robot successfully accomplished the task.

VII. Conclusion

We define *roboKG*, a knowledge graph representing the properties of objects, the relationships between tasks, and the appropriate factors of robotic manipulation. In our *roboKG*, an object is represented in terms of its label, components, and material. Using hierarchical knowledge graph embedding methods, we generate semantic representations of the entities and relations in *roboKG*, which allows us to make predictions on a gripper type, a grasping component, and a grasping force for semantic grasping. Based on the predictions, a real robot succeeds in accomplishing various tasks on diverse household objects. We plan to extend and apply our approaches to other robotics applications requiring automated reasoning and advanced knowledge representation.

References

[1] J. Mahler *et al.*, "Learning ambidextrous robot grasping policies," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau4984.

[2] H. Merzić, M. Bogdanović, D. Kappler, L. Righetti, and J. Bohg, "Leveraging contact forces for learning to grasp," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 3615–3621.

[3] V. Satish, J. Mahler, and K. Goldberg, "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1357–1364, Apr. 2019.

[4] H. Dang and P. K. Allen, "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1311–1317.

[5] F.-J. Chu, R. Xu, and P. A. Vela, "Learning affordance segmentation for real-world robotic manipulation via synthetic images," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1140–1147, Apr. 2019.

[6] L. Antanas *et al.*, "Semantic and geometric reasoning for robotic grasping: A probabilistic logic approach," *Auton. Robots*, vol. 43, no. 6, pp. 1393–1418, 2019.

[7] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw.*, vol. 33, no. 2, pp. 494–514, Feb. 2022.

[8] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 3065–3072.

[9] I. Balažević, C. Allen, and T. Hospedales, "Multi-relational poincaré graph embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 4463–4473.

[10] Y. Bai, R. Ying, H. Ren, and J. Leskovec, "Modeling heterogeneous hierarchies with relation-specific hyperbolic cones," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 12316–12327.

[11] J. Song *et al.*, "Robust task-based grasping as a service," in *Proc. IEEE Int. Conf. Automat. Sci. Eng.*, 2020, pp. 22–28.

[12] W. Liu, A. Daruna, and S. Chernova, "CAGE: Context-aware grasping engine," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2550–2556.

[13] A. Murali, W. Liu, K. Marino, S. Chernova, and A. Gupta, "Same object, different grasps: Data and semantic knowledge for task-oriented grasping," in *Proc. Conf. Robot Learn.*, 2020, vol. 155, pp. 1540–1557.

[14] P. Ardón, Èric Pairet, R. P. A. Petrick, S. Ramamoorthy, and K. S. Lohan, "Learning grasp affordance reasoning through semantic relations," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4571–4578, Oct. 2019.

[15] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoğlu, and G. Bartels, "Know rob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 512–519.

[16] M. Beetz *et al.*, "Robotic roommates making pencakes," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2011, pp. 529–536.

[17] G. Cheng, K. Ramirez-Amaro, M. Beetz, and Y. Kuniyoshi, "Purposive learning: Robot reasoning about the meanings of human activities," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaav1530.

[18] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artif. Intell.*, vol. 247, pp. 95–118, 2017.

[19] C. Chung and J. J. Whang, "Knowledge graph embedding via metagraph learning," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 2212–2216.

[20] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[21] R. Xie, Z. Liu, and M. Sun, "Representation learning of knowledge graphs with hierarchical types," in *Proc. Int. Jt Conf. Artif. Intell.*, 2016, pp. 2965–2971.

[22] A. Daruna, W. Liu, Z. Kira, and S. Chetnova, "RoboCSE: Robot common sense embedding," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 9777–9783.

[23] A. Daruna, L. Nair, W. Liu, and S. Chernova, "Towards robust one-shot task execution using knowledge graph embeddings," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11118–11124.

[24] A. Daruna, M. Gupta, M. Sridharan, and S. Chernova, "Continual learning of knowledge graph embeddings," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1128–1135, Apr. 2021.

[25] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *Proc. IEEE Int. Conf. Adv. Robot.*, 2015, pp. 510–517.

[26] G. Georgakis, M. A. Reza, A. Mousavian, P.-H. Le, and J. Košecká, "Multiview RGB-D dataset for object instance detection," in *Proc. IEEE Int. Conf. 3D Vis.*, 2016, pp. 426–434.

[27] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 1817–1824.

[28] R. Speer, J. Chin, and C. Havasi, "ConceptNet 5.5: An open multilingual graph of general knowledge," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 4444–4451.

[29] G. Jocher *et al.*, "ultralytics/yolov5: V6.0 - YOLOv5n 'nano' models, roboflow integration, TensorFlow export, OpenCV DNN support," 2021, doi: 10.5281/zenodo.5563715.

[30] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[31] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[32] K. Mo *et al.*, "PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 909–918.

[33] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3479–3487.

[34] J. Xue, H. Zhang, K. Nishino, and K. J. Dana, "Differential viewpoints for ground terrain material recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1205–1218, Mar. 2022.

[35] S. Thermos, G. T. Papadopoulos, P. Daras, and G. Potamianos, "Deep affordance-grounded sensorimotor object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 49–57.

[36] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 2787–2795.

[37] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "RotatE: Knowledge graph embedding by relational rotation in complex space," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–18.

[38] J. Liang, J. Mahler, M. Laskey, P. Li, and K. Goldberg, "Using dVRK teleoperation to facilitate deep learning of automation tasks for an industrial robot," in *Proc. IEEE Int. Conf. Automat. Sci. Eng.*, 2017, pp. 1–8.

[39] D. P. Holland, E. J. Park, P. Polygerinos, G. J. Bennett, and C. J. Walsh, "The soft robotics toolkit: Shared resources for research and design," *Soft Robot.*, vol. 1, no. 3, pp. 224–230, 2014.

[40] A. Ng and M. Jordan, "On Discriminative vs, Generative classifiers: A comparison of logistic regression and naive bayes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, vol. 14, pp. 841–848.