2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

RobotVQA — A Scene-Graph- and Deep-Learning-based Visual Question Answering System for Robot Manipulation *

Franklin Kenghagho Kenfack[1], Feroz Ahmed Siddiky[1], Ferenc Balint-Benczedi[1] and Michael Beetz[1]

Abstract— **Visual robot perception has been challenging to successful robot manipulation in noisy, cluttered and dynamic environments. While some perception systems fail to provide an adequate semantics of the scene, others fail to present appropriate learning models and training data. Another major issue encountered in some robot perception systems is their inability to promptly respond to robot control programs whose realtimeness is crucial.**

This paper proposes an architecture to robot vision for manipulation tasks that addresses the three issues mentioned above. The architecture encompasses a generator of training datasets and a learnable scene describer, coined as RobotVQA for Robot Visual Question Answering. The architecture leverages the power of deep learning to predict and photo-realistic virtual worlds to train. RobotVQA takes as input a robot scene's RGB or RGBD image, detects all relevant objects in it, then describes in realtime each object in terms of category, color, material, shape, openability, 6*D*-pose and segmentation mask. Moreover, RobotVQA computes the qualitative spatial relations among those objects. We refer to such a scene description in this paper as scene graph or semantic graph of the scene. In RobotVQA, prediction and training take place in a unified manner. Finally, we demonstrate how RobotVQA is suitable for robot control systems that interpret perception as a question answering process.

I. INTRODUCTION

Imagine a manipulation robot (PR2) within a human-centered kitchen, standing up in front of a table, ready to serve a cup of coffee and a piece of cake to a human guest. On the table, there is a thermos full of warm coffee, a plastic cup, a ceramic cup and a plate containing a coffee spoon. To successfully perform this task, we identify at least three issues that our robot should be able to address. The first problem consists in defining an **appropriate scene ontology** for tracking the visually established semantics of the robot's scene. Our robot should not only be able to detect a cup but also estimate the material and chooses the ceramic cup. Moreover, our robot should know that before putting a piece of cake into the plate, it should take the spoon out of the plate. This problem is also referred to as the *what it means for a manipulation robot to visually understand the scene*-question [4]. We devised two criteria for evaluating this ontology's quality namely the ontology *completeness*, which is closely related to the amount of information supported by the ontology and then the criterium of *structuredness* which refers to how structured the ontology's formalism is, so that it can be accessed by primitive computational mechanisms such as robot control programs. The second issue is the **uncertainty problem**. More than being ill-posed and ill-defined, computer vision tasks require high adaptability from vision systems. That is, now that our robot knows what it should know, it should actually know it with good accuracy no matter the amount of intrinsic lack of information the robot has about the scene (i.e. noise, occlusion, depth loss, new scenes). Though deep learning techniques have been shown promising in addressing this issue, getting the appropriate training data has been the crisis:

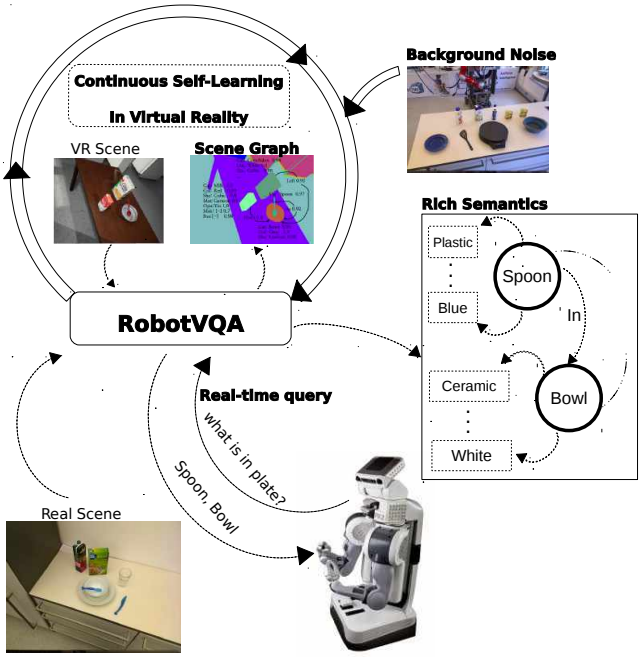

Fig. 1: Building a robot vision system (RobotVQA) for manipulation tasks with focus on **continuous and effective learning** (Virtual Reality), **rich scene semantics** (scene graph) and **realtimeness** (fast robot control).

most available datasets are diesembodied and unsituated, therefore not suitable for a particular robot. **Realtimeness** is the third issue encountered by computer vision systems and refers to the ability of a vision system to react as fast as possible to meet its environment's dynamics. Imagine that while pouring some coffee into the cup, the latter spills, then the vision system followed by the robot control should immediately be aware of it. Assuming that our vision system is realtime, then the robot control program will be too: that is, the scene semantics presented above allows the control program to only access the necessary information by querying the vision system for it. Realtimeness is relevant for vision systems supporting living applications such as manipulation robots.

Summarisingly, successful robot manipulation needs an environment-adaptable vision system that can provide in realtime a sufficiently complete and structured description of the scene with good accuracy so that a running robot control program can just query it to quickly get any information about the scene, which is actually necessary to pursue the manipulation task.

In the next section, we show that actual computer vision systems fail to properly address these issues, then expose an architecture to address these issues and finally provide empirical evidences about the practicability of the scheme. According to the architecture (figure 1), the robot environment is virtualized, then the robot

*This work was supported by the EASE Research Project and Institute for Artificial Intelligence (IAI) at the University of Bremen

[1]Researchers at the Institute for Artificial Intelligence (IAI) at the University of Bremen

is continuously trained in the virtual world to perceive its scene for manipulation and the knowledge acquired by the robot are transferred to the real world. In doing this, this paper achieves the following manifold contribution:

- Design and implementation of a **kitchen-activity-related dataset synthesizer**. It defines a **scene representation**, noted in this paper as *Scene Graph*, for autonomous manipulation robots. Then, it generates robot scene images and annotates them with respect to this representation.
- Synthesis of a **big set of kitchen-activity-related RGBD-images**, annotated with corresponding scene graphs and augmented with real images.
- Design and implementation of a **unified deep learning model**, coined as RobotVQA, that takes a scene RGB(D) image as input and outputs the corresponding scene graph. RobotVQA stands for Visual Question Answering for Robots. We demonstrate **transferability of RobotVQA's knowledge** from virtual to real worlds and **suitability** to robot control programs.

II. RELATED WORKS

Careful investigation of the literature on computer and robot vision systems revealed two major insights regarding the three issues just cited above in the introduction.

On the one hand, deep-learning-based approaches, mainly relying on supervised Convolution Neural Networks (CNNs) and supervised Recurrent Neural Networks (RNNs), demonstrate ability to handle uncertainty: the technique allows end-to-end learning for self-construction of models (i.e. low biasedness), transfer and multi-task learning for fast adaptation in evolutive (i.e. dynamic) environments and data-scalability for generalization [11]. However, this technique is not exploited enough in robotics due foremost to the lack of rich annotated data required for training: this issue is commonly referred to as the big data's crisis. One could argue for training deep-learning-based robot vision systems with publicly online available datasets, unfortunately this would lead to the two well-known problems of **disembodiedness** (the robot body's actual architecture is not considered) and **unsituatedness** (the robot's actual operating environment is not considered). Secondly, it has been noticed that most of these approaches are not real-time on standard computers such as most robot computers and usually only output ambiguous information (i.e. unstructured scene description) which are only difficultly accessible to primitive agents such as robots (i.e. translation of image to natural language text). Another common flaw in today's visual perception systems is their focus on solving only a single elementary task such as recognition, segmentation, detection or pose estimation of scene objects. Though they tend to successfully accomplish these tasks, it only results in a poor scene description, limiting therefore the robot's ability to physically interact with its environment. As pointed by [4], a cognitive vision system should go beyond just focusing on a single task and deal with all these atomic perceptual operations as well as considering relational aspects among scene entities such as introduced in [2]. Unfortunately, while most of these systems are not suitable for robot manipulation (i.e. inappropriate concepts), few of them appropriate to it such as [3], even when deeply relying on huge abstract ontologies of manipulation robots' worlds, present weak grounders of symbols in sensor data: these grounders are based on traditional computer vision techniques.

On the other hand, it has been noticed that more than providing limited scene semantics to robots, traditional computer vision systems such as [9] based on technologies such as traditional Artificial Neural Networks (ANNs), K-Nearest Neighboor Classifiers (KNNs) and Decision Trees (DTs) are inherently limited in modelling the environment relaxation (e.g. noise, cluttter) and unable to scale with the expansion of the robot's operating domain (i.e. weak generalization). These approaches usually rely on naive assumptions (e.g shape- and color-based segmentation, color-histogram-based color estimation, CAD models) that completely break down when slightly complex domains (e.g. multi-color and irregularly shaped objects, visual constancy) are considered.

III. ARCHITECTURE

The figure 2 depicts the architecture proposed in this paper to build robot vision systems for manipulation tasks. (1) The process starts with the *dataset generator* which is currently a semi-automatic module. In the dataset generator, a submodule called the *scene generator* defines a scene ontology to represent the robot scene's semantics: we refers to such representation in this paper as *scene graph*. (2) Then, it prepares virtual reality assets (3*D*-CAD models, textures, materials) in order to virtualize the robot world; these vr assets are stored in a knowledge base. The virtual reality engine is depicted in the figure as a combination of an interface (vr scene) for interactions with the scene generator and a hidden part called vr semantics which is stored in the knowledge base. The vr semantics is a model of all possible knowledge that one can get from the vr world such as object poses, masks, geometry and physics. Before starting a simulation, the scene generator initializes the scene using configuration files from the knowledge base. These configuration files hold information about the virtual robot's trajectory, which objects are parts of the scene, how they are spatially configured and how the scene should be updated. (3) As the scene generator starts the simulation and updates the scene over time, the second submodule *data collector* collects the scene images and annotates them with scene graphs. (4) The collected dataset is then passed to our second module *scene describer* (RobotVQA) for deep learning and the knowledge acquired are transferred to the real world. To reinforce knowledge transferability, the dataset collector augments vr data with some real data in order to incorporate background noise. (5) During real robot manipulation, RobotVQA keeps outputting the scene semantics as scene graphs so that a robot control program can just query it for the specific information needed to pursue the manipulation. (6) Notice that the process can go back to (1) for a lifelong learning. Our implementations can be found online at [7].

IV. DATASET GENERATOR

The novel approach proposed in this paper to address the problem of visual scene understanding for manipulation robots starts with a careful definition of the scene's semantics. This semantics does not only inform about the outcomes of the perception system but also about which data are needed to train it.

A. Scene Graph: Towards Complete & Structured Scene Semantics

The concept of scene graph as representation of visual interpretations of scenes is not new. However, the term has been only informally defined so far [6]. In this paper, a scene graph is essentially a directed graph in which each node is a relevant scene object's formal description and each directed edge is a formal relationship between two objects. Each relevant object is defined by its affordance, shape, color, material, openability, 6D-pose, instance mask and timestamp, allowing the robot to effectively manipulate the object. As far as relations among objects are concerned, we essentially focus on spatial relations in this paper, which mainly

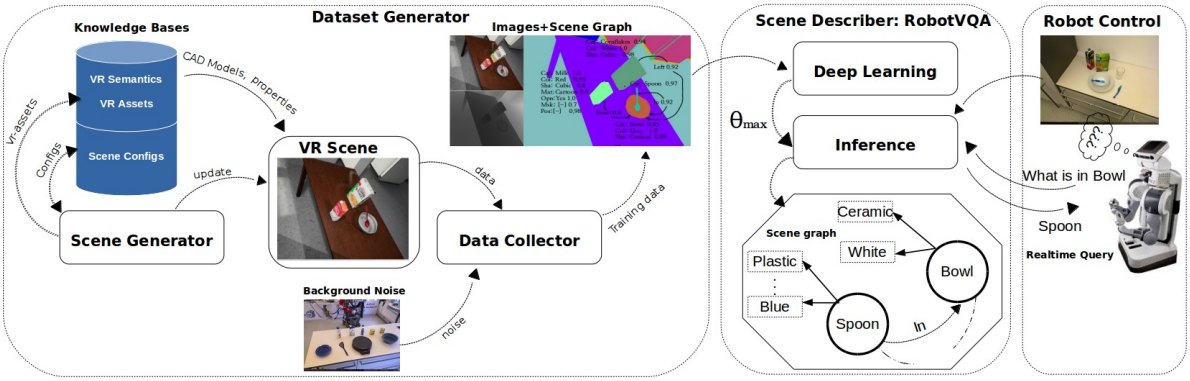

Fig. 2: **Our Architecture.** Continuous self-learning of the robot vision system (RobotVQA) in virtual worlds (vr) with appropriate training data. Getting background noise (e.g. chairs, cables, pictures) from real worlds and injecting them into the learning process. Transferring knowledge acquired from virtual to real world. RobotVQA provides a sufficiently complete and structured semantics of the scene (scene graph). Robot control program accesses necessary information about the scene through real-time query of RobotVQA.

provide the robot with insights into how scene objects do and can interact with each other.

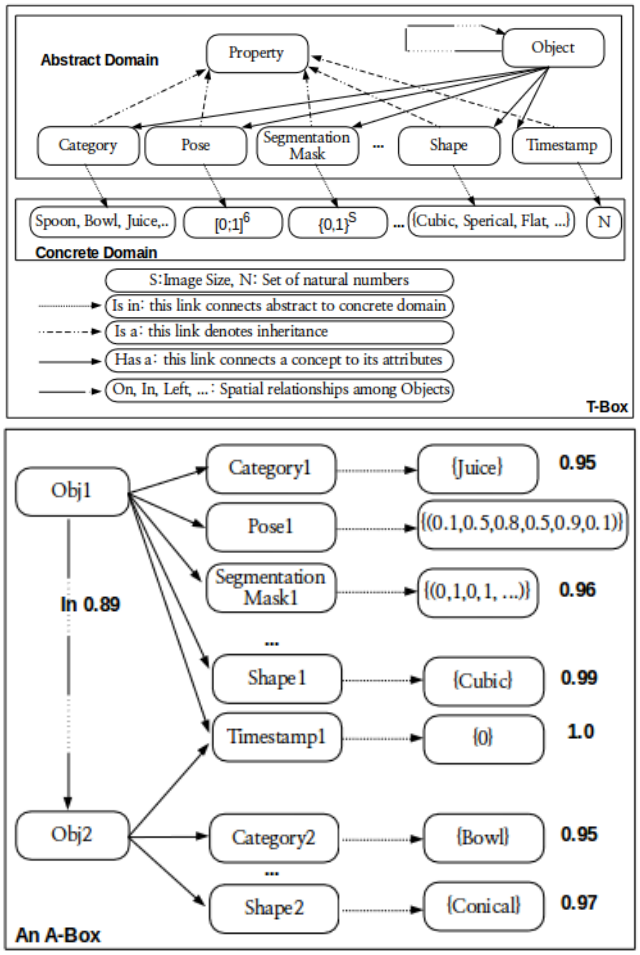

Fig. 3: On top is a T-Box of the ontology of the manipulation robot's scene. At the bottom is a corresponding A-Box. Notice that the numbers besides the rules are merely truth probabilities

For instance, by knowing that there is a spoon inside the bowl, the robot will first take the spoon out of the bowl before using the latter. To represent scene graphs, the formal probabilistic decidable general description logic with concrete domains ***P-ALCNHR+(D)*** is adopted [8]. In this logic, knowledge representation consists of T-Boxes and A-Boxes. The T-Box a.k.a. terminological box is the set of general facts or rules about the world. It contains definitions of concepts as well as definitions of relations among concepts. Each concept is then linked to a well-defined set of concrete values. Given that the logic is probabilistic, each rule or fact is assigned a truth probability. In contrast to T-Boxes, the A-Box a.k.a. assertional box holds specific facts about the world or facts about specific worlds, where each assertion is also assigned a truth probability. In this paper, the general semantics of the robot's scene a.k.a abstract scene graph is represented as a T-Box and specific facts about the robot's scene a.k.a. concrete scene graph are represented as an A-Box. The figure 3 illustrates the probabilistic terminological box (i.e. T-BOX) of our scene ontology (i.e. abstract scene graph) and a corresponding assertional box (i.e. concrete scene graph). At the beginning of the robot's exploration, the A-Box is empty and gets populated over time. This incremental population of the A-Box is due on the one hand to the fact that the robot has only a partial view of the scene at a given time. On the other hand, the robot'scene is dynamic.

B. Robot World Virtualization

Virtualization basically allows to build realistic computer models of real worlds where external real agents usually drive virtual agents through I/O-mechanisms such as joysticks. Virtualization's main advantage is the fact that it allows to carry out experiments at very low cost compared to reality. In this paper, we present the virtualization of the EASE CRC's kitchen (see acknowledgement) and robots: major outputs of this step are virtual textures, materials and 3D-models. Then, we simulate the virtual kitchen, use external control programs to move the virtual robots in the virtual kitchen and make them performing manipulation tasks. Finally, we progressively collect the virtual kitchen's images and the associated ground truths for learning. This work relies particularly on Unreal Engine 4 (UE4) and UnrealCV as frameworks respectively for the world virtualization and for enabling interactions between external control programs and virtual entities. Figure 4 demonstrates an overview of the virtualization process.

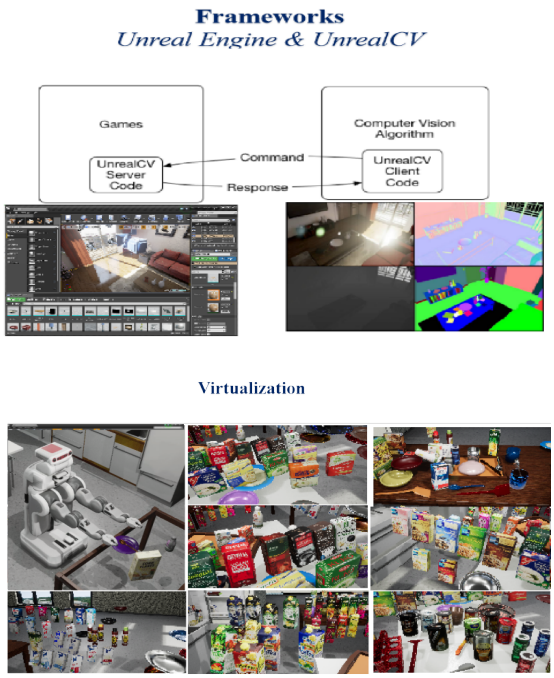

Fig. 4: An Overview of the robot world virtualization

C. Spatial Relation Compression

The robot world virtualization is followed by the simulation and then the dataset collection. Notice however that spatial relations among objects, in contrast to any other information found in the scene ontology (e.g. color, 6D-pose), are observer- and context-relative information that need to be explicitly tracked as a whole.

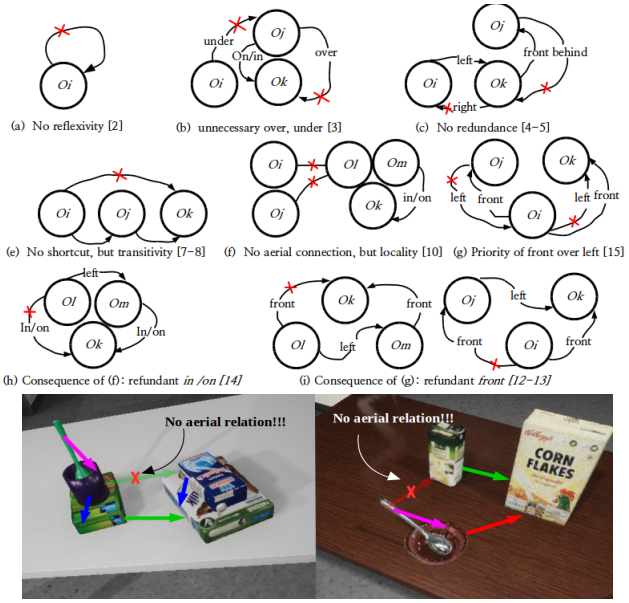

Fig. 5: Spatial relation compression (rules a-i), allowing to record only 4 and 3 spatial relations (arrows) in the above scenes rather than 25 and 16. Red=front, green=left, pink=in, blue=on.

For 8 different spatial relation types namely $\{on,\ in,\ left,\ right,\ above,\ below,\ front,\ behind\}$ and N scene objects there are expectedly $\mathcal{O}(8\times N^2)$ spatial relations among these scene objects. Notice that the complexity of the set of spatial relations grows quadratically with the number of scene objects. This complexity becomes rapidly burdensome for the whole processing pipeline namely from the annotation and tracking during the simulation till the training and inference in RobotVQA. To address this issue, a **quasi-lossless** compression scheme $\mathbb{CS}$ was devised to lower the complexity of the set of spatial relations to a linear function of N (i.e. $\approx$ as many relations as objects). Quasi-lossless due to rule g which is necessary to guarantee a linear complexity while remaining reasonable: the idea here is that the manipulator does not care anymore whether an object O_1 is left another object O_2 as long as O_1 is front O_2 since O_2 is usually unreachable due to occlusion. Notice moreover that the compressed set is ***unique***. The above figure 5 graphically illustrates the entire (***rules a-i***) compression scheme $\mathbb{CS}$ just mentioned. This compression step is already automatically performed in the data collector below.

D. Data Collection

To collect our training dataset containing aroung $71,000$ scene images + graphs, we ran 35 simulations each of them enabling the collection of around 2000 RGBD-images of size 640×480 and containing around $0\sim20$ relevant objects.

Fig. 6: Overview of the synthetic dataset

At the beginning of each simulation, the scene background (e.g. chair, table) as well as the scene foreground (e.g. robot, food items, utensils) are set. Then, the external control program moves the robot and for each robot pose, the scene's RGBD image as well as the corresponding scene graph is saved on the disk and the scene foreground is randomly and completely reconfigured (i.e. assignment of random pose, material, color, shape to foreground objects). This scene variation scheme was inspired by [1] and only **emulates** robot manipulation. Emulated manipulation means that objects are moved directly by adequately updating their properties rather than using robot arms. Controlling the arm joints of the virtual robot in figure 4 for a real manipulation is therefore not in the scope of this paper. An overview of the dataset is presented by figure 6. Notice that the collected dataset is **situated** and **embodied**: the data are generated by the robot (***at least camera's intrinsic and extrinsic properties***) itself while performing the intended task within its intended operating domain.

E. Data Augmentation

Background noise are usually objects which are quite often parts of the robot environment but not parts of the robot manipulation tasks going on in that environment. Pictures are for instance quite often hanging on kitchen walls or radios on kitchen tables though they are actually not related to the cooking activities. Since virtualization quite often ignores these details and that discriminative learners just model the boundaries amoung classes, a learning model which was supervisedly and discriminatively trained on solely *clean* synthetic images would show a high false positive rate when tested

on real images. For this reason, we augment our big synthetic dataset with some real images incorporating background noise: this improves the discriminative power of the learner. We collected around 500 real images from the EASE CRC's kitchen and 111 real images from the internet, then annotated them with the free software LabelMe and a self-developed software known as relationAnnotator [7]. Figure 7 illustrates the augmentation of our synthetic dataset with real images for the sake of incorporating background noise.

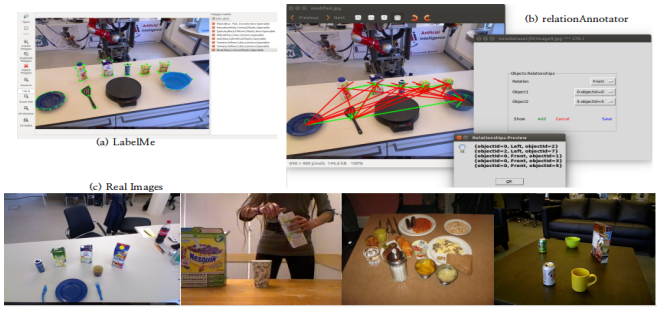

Fig. 7: Augmentation of synthetic dataset with background noise

V. SCENE DESCRIBER: ROBOTVQA

This section describes a deep-learning-based vision model, coined as RobotVQA, proposed in this paper to learn from the collected synthetic dataset in the previous section how to infer in real-time a scene graph from the scene image. Figure 8 illustrates the architecture of RobotVQA.

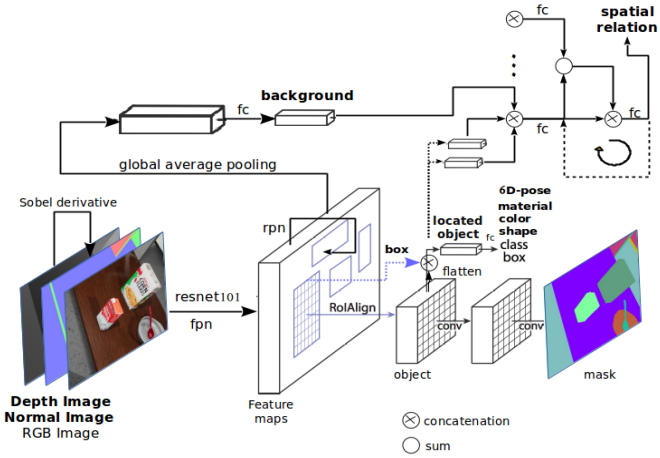

Fig. 8: An overview of RobotVQA's architecture. **Contributions to MaskRCNN are highlighted in bold**.

A. RobotVQA's Key Features

RobotVQA presents the following key features:

Transfer Learning: RobotVQA extends MaskRCNN [5], which is actually by far the state of the art on instance detection and segmentation, running at almost $5fps$ on larger images (1024×1024) containing hundreds of objects. A faster reaction is therefore expected from RobotVQA since it only operates on smaller images (i.e. 640×480) containing fewer scene objects (i.e. $0 \sim 20$).

Multimodal Inputs: RobotVQA can work either on RGB or RGBD images, making it therefore usable in many contexts.

Multitask Learning: RobotVQA is heavily multitasking, in the sense that it solves several subproblems in a unified manner. The advantages of multitask learning are manifold. Foremost, it regularizes the learning and prevents the system to overfit the training set on a single task. Secondly, it enables joint learning and therefore implicit reasoning about all the subtasks at once (inductive transfer learning): a spoon would difficultly be of circular shape for instance. Thirdly, multitasking learning encourages the specialization of sub-solvers, in the sense that a subtask can be solved very differently as from how the others are. On the other hand, explicit multitasking eases the extension and reduction of the model: for a new subtask, a solver would simply be appended to the model's head. Another positive aspect of multitasking is the reinforcement of parallelism, which in turn plays in favor of realtimeness. Finally, explicit multitask learning can operate on heterogenous datasets: the model can train on multiple datasets from multiple domains and for multiple goals.

B. RobotVQA's Machinery and Implementation

RobotVQA's Input: a $(W,H,7)$-image tensor combining a $(W,H,3)$-RGB scene image, a $(W,H,1)$-scene depth map and a $(W,H,3)$-scene normal map. This normal map acts as cues for object shape and is more camera-insensitive than the depth map. Note however that RobotVQA can disable these depth-related maps and only work with the $(W,H,3)$-RGB map.

RobotVQA's Basenet: RobotVQA leverages Mask-RCNN's feature extractor, based on **ResNet101** for a safe deeper computation and **FPN** to allow invariance to input scale and conservation of information distribution in the feature maps.

RobotVQA's Detector: The Region Proposal Network (RPN) of Mask-RCNN is exploited to explicitly localize interesting scene objects. RPN outputs a list of bounding boxes enclosing the detected objects.

RobotVQA's Object Features: Based on the image's feature maps and the list of bounding boxes, RobotVQA makes use of Mask-RCNN's ROIAlign function to extract each object's feature maps. Since each object's feature maps solely contain local information, each object's bounding box is additionally appended to its feature maps to enable spatial reasoning.

RobotVQA's Output Nodes (Head): It consists of an object describer and a spatial reasoner. The object describer is made up of classifiers and regressors (sub-solvers). Each classifier predicts one of the object's properties affordance, color, material, shape, openability and segmentation mask. Each regressor estimates either the object's $6D$-pose or bounding-box. The list of objects is sequentially processed, where each object is simultaneously passed to each sub-solver: this makes the computation invariant to the length and order of the list, then enables weight sharing as well. Except the instance segmentation which is based on a convolution-deconvolution block (MaskRCNN), all other sub-solvers are multi-layer perceptrons (MLPs) with linear/relu activations. The $6D$-pose estimator is however a multi-layer perceptron (MLPs) with leaky-relu activations and residual connections (deep non-linear regression). The working of the spatial reasoner is a bit tricky. (1) First, we compute the background B of the scene image which aims at capturing global information such as the scene's orientation. (2) Then, we compute the spatial relation between each pair $O_i \oplus O_j \oplus B$ of objects O_i and O_j. B is an additional input to this computation. (3) Thirdly, we iteratively compress the obtained set of relations G_0 according to the scheme $\mathbb{CS}$. To perform compression, each relation r in G_0 is either classified as useless or useful given the whole relational context G_0. (4) The useful relations out of G_0 are stored into a new more compressed relation set G_1 and the process is iterated from (3). Note that the number of iterations n is a hyperparameter and is set to 1 for the results presented in this

paper. This mechanism was inspired by [10, 12]. The algorithm 1 formally clarifies the above steps, where *GAP* stands for Global Average Pooling and h, g, f are leaky relu, relu and relu MLPs.

Algorithm 1 Spatial Reasoning with Compression Scheme $\mathbb{CS}$

Require: $\{O_1,..,O_m\}$, *where object* $O_i = local\ feature + 2D\ Bounding\ Box$
FM, *feature maps from basenet*
n, *number of compression steps*
Ensure: $G_n = \{r_1^{(n)},..,r_l^{(n)}\}$, *set of spatial relations among above objects compressed with respect to* $\mathbb{CS}$
$B \leftarrow h \circ GAP(FM)$, *scene background (e.g. global orientation)*
$G_0 \leftarrow \emptyset$, *initialization of the relation set without compression*
for $i \leftarrow 1:m,\ j \leftarrow 1:m$ **do**
  $r \leftarrow f(O_i \oplus O_j \oplus B)$, *context-free relation between* O_i *and* O_j *(i.e without compression or no consideration of neighboorhood relations)*
  $G_0[(i-1)*m+j] \leftarrow r$, $\oplus$ *stands for concatenation*
end for
for $p \leftarrow 0:n-1$ **do**
  $S_p \leftarrow \frac{\sum_{k=1}^{m^2} G_p[k]}{m^2}$, *context as average sum of actual spatial relations*
  $G_{p+1} \leftarrow \emptyset$, *initialization of next relation set (compressed)*
  for $i \leftarrow 1:m,\ j \leftarrow 1:m$ **do**
    $r \leftarrow g(G_p[(i-1)*m+j] \oplus S_p)$, *context-sensitive relation between* O_i *and* O_j *(i.e with compression or consideration of neighboorhood relations)*
    $G_{p+1}[(i-1)*m+j] \leftarrow r$
  end for
end for

Loss & Implementation: RobotVQA was written in Python and trained with the frameworks Tensorflow and Keras, well-known for the services they offer while developing deep-learning-based applications, from the dataset preparation to the visualization of the results. During training, a muti-loss $\mathbb{L}$ is used and defined as a weighted sum of individual losses. While classifiers are trained with the "average" cross-entropy loss, the regressors rely on the "average" smooth L_1-loss. The word "average" emphasizes the fact that the mean rather than the sum of losses over all object or relation instances on the image is considered. Such a multi-loss $\mathbb{L}$ acts as a learning regularizer and further handles the problem of imbalanced classes. The equations 1 further explain RobotVQA's loss.

$$\begin{aligned}
\mathbb{L} &= \mathbb{L}_1 + \mathbb{L}_2 \\
\mathbb{L}_1 &= \alpha_1\mathbb{L}_{class} + \alpha_2\mathbb{L}_{col} + \alpha_3\mathbb{L}_{mat} + \alpha_4\mathbb{L}_{shp} \\
\mathbb{L}_2 &= \alpha_5\mathbb{L}_{opn} + \alpha_6\mathbb{L}_{mask} + \alpha_7\mathbb{L}_{pose} + \alpha_8\mathbb{L}_{bbox} + \alpha_9\mathbb{L}_{rel} \\
\mathbb{L}_{rel} &= \beta_1\mathbb{L}_{rel}^{(1)} + \beta_2\mathbb{L}_{rel}^{(2)}
\end{aligned} \tag{1}$$

Notice that the loss of the spatial reasoner in 1 has been splitted into two losses namely $\mathbb{L}_{rel}^{(1)}$ and $\mathbb{L}_{rel}^{(2)}$. For N scene objects, we count $O(8 \times N^2)$ spatial relations. However, as mentioned earlier, the spatial reasoner is expected to output a compressed set of relations of size $O(N)$. That is, within a graph of 10 objects, only 10 out of 800 possible edges are sufficient to understand how the objects are spatially configured in the scene: we qualify those 10 edges as foreground edges and the 790 others as background edges. Since background edges are considerably dominant in size, a naive spatial reasoner tends to outputs empty sets of relations. On the other hand, ignoring the background edges causes the spatial reasoner to show a high false positive rate. To solve this issue, we consider the loss of the spatial reasoner without background edges ($\mathbb{L}_{rel}^{(1)}$) and its loss with background edges ($\mathbb{L}_{rel}^{(2)}$), then we set the final loss as a weighted average of both losses. In our implementation, we observe better results while fixing $\alpha_i = 1$, $\beta_1 = 0.999$ and $\beta_2 = 0.001$.

VI. ROBOTVQA & ROBOT CONTROL

A. Visual Question Answering for Robots: RobotVQA

Robot control programs are crucial in the sense that they are top implementations of living robots as task algorithms. Allowing these programs to get information about the world through queries has at least three benefits. Firstly, it allows to turn the task's algorithm directly into program. Secondly, it allows the control program to only focus on the specific information needed and therefore run faster. Finally, it reinforces the generalization of the control program with respect to the perception system. In the vision community, a general trend to address the problem of visual question anwering has been to supervisedly train systems with images and questions. Unfortunately, this solution suffers from two problems with respect to robotics. On the one hand, the language of queries and answers is informal (e.g. natural), and on the other hand the learning is ineffective: that is, there are infinitely many questions for a single image. To address those two issues, we proposed a formal language for describing scene so that any formal language of queries can be used to extract information. Then, to make our solution independent of the type and number of queries, the proposed scene representation tries to cover as much fundamental information about the scene as possible so that any query can be reduced to a lookup problem of those information. ***Notice that the ability of answering questions (i.e. VQA) is reduced to the ability of generating accurate scene graphs***. In order to demonstrate that the proposed scene representation covers a good amount of fundamental information, the table I illustrates how the representation is exploited to address vision-specific topics during manipulation.

Topics	**Crucial information**
Detection	List of objects, relations
Localization	6D-pose
Affordance	Object category
Grasping	6D-pose, Segmentation mask
Object Kinematics	6D-pose
Spatial interaction	Spatial relations, Shape, Segmentation mask
Tracking	Color, Category, Shape, 6D-Pose, Material
Physical contact	Material
Access mode	Openability
Counting	Satisfying detection

TABLE I: Leveraging scene graphs to address vision-specific topics on manipulation tasks.

B. Robosherlock's Formal Query Language

In developing the robot perception system Robosherlock [3], the authors argued on interpreting robot perception as a controlled process that takes place through visual question answering to selectively and incrementally access the world. To achieve this, the authors proposed a query language, a query interpreter and a pervasive vision system whose goal is to deliver elementary information about the scene such as shapes, color, object hypotheses. To retrieve an information from the scene, the robot control program sends a query to the query interpreter. The query interpreter outputs a procedure that requests the desired information from the pervasive system and sends the information back to the robot control program. Robosherlock's query language offers three general queries:

- (**"detect"**, $\{prop_1 : val_1,\ prop_2 : val_2,...,\ prop_n : val_n\}$): this query returns the unique identifiers and 6D-poses of all scene objects whose each property $prop_i$ takes the values val_i. For instance, (**"detect"**,{"$type$" : "$spoon$", "$color$" : "red", "$material$" : "$plastic$"}) localizes and uniquely identifies all red plastic spoons.
- (**"examine"**, **object_id**, $\{prop_1,\ prop_2,...,\ prop_n\}$): this query returns the values of each property $prop_i$ of the scene object with identifier **object_id**. For instance, (**"examine"**,

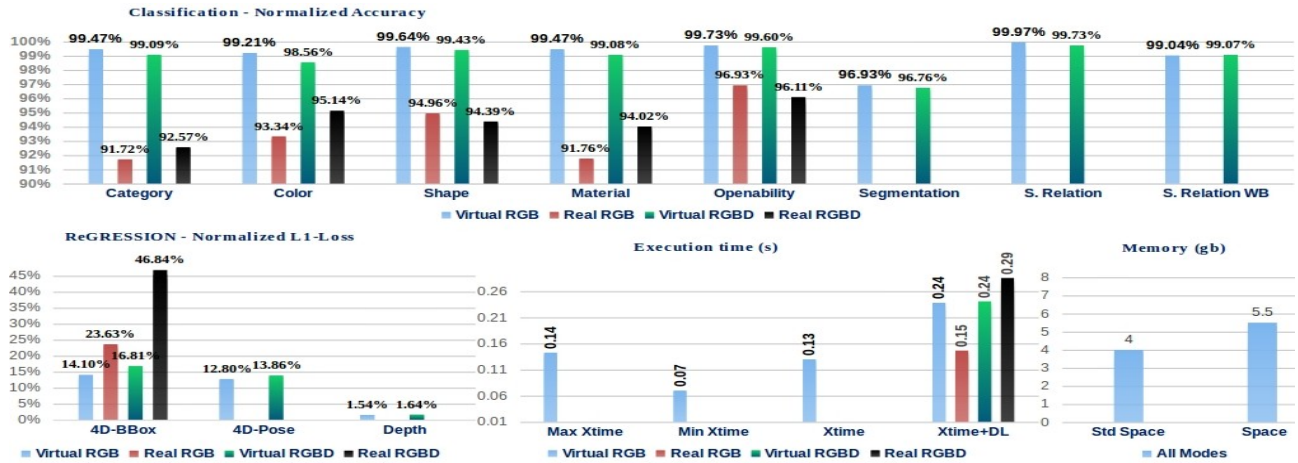

Fig. 9: RobotVQA's Evaluation. *Notice that only appearance-based properties are considered on real images:* ***reasonable and no burdensome annotation***. The first chart depicts the performance of RobotVQA's classifiers: ***Accuracy*** *is the average rate of correct detections per property*; **S. Relation** = *Accuracy on spatial reasoning with compression*; **S.Relation WB** = *Accuracy on spatial reasoning without background edges*. The second chart informs about the performance of RobotVQA's regressors: *Normalized L1-Loss*; **4D-Pose=** Depth + 3D-Orientation. The third chart informs about RobotVQA's inference time; **Max Xtime, Min Xtime and Xtime** are respectively the approximative max, min and average speed of conscious human vision and **Xtime+DL=**RobotVQA's speed. The last chart depicts RobotVQA's space complexity; **Space=** RobotVQA's space complexity and **Std Space=** memory size on very standard computer.

"spoon_1",{"*color*", "*material*"}) returns the color and material of the object with identifier spoon_1.

- (**"track"**, **object_1**, **object_2**): this query checks whether the scene objects with identifiers **object_1**, **object_2** are same. It is a query for tracking scene objects and can be achieved by comparing the visual and spatial properties of both objects.

Note that Robosherlock is the core perception framework at IAI LAB and EASE CRC.

C. Robosherlock Vs RobotVQA

To demonstrate the suitability of RobotVQA to robotics, we have successfully substituted the pervasive vision system of Robosherlock by RobotVQA. Moreover, RobotVQA provides spatial relationships among scene objects, object material and object openability that Robosherlock actually does not. RobotVQA also demonstrates stronger learning ability than actual Robosherlock's pervasive vision system which relies to a great extends on shallow machine learning techniques such as segmentation based on the object-on-plane-based assumption, color-based clustering for segmentation and color histogram for color estimation. With RobotVQA, Robosherlock could detect flat and small objects such as plates, spoons that could only be detected with difficulties before due to the object-on-plane-based assumption: that is, any scene object lies on a plane. RobotVQA also allows Robosherlock to detect objects in any spatial configuration in the scene such as objects on top of or in others: this was also difficult before due to object-on-plane-based assumption. Evidences to assertions provided in this section are shown in the figure 10 and in the attached demonstration video.

VII. EXPERIMENTATION

We evaluate RobotVQA in four different ways. Firstly, we test the ability of RobotVQA to infer scene graphs from scene RGB-images and then from scene RGBD-images. This second test would also exhibit the contribution of the scene depth. Secondly, we evaluate the ability of RobotVQA to transfer knowledge from the virtual to the real world in both modes namely RGB and then RGBD.

A. Experimental Settings

The overall dataset was randomly splitted into three sets: A training set of 50790 synthetic RGB(D)-images and 111 real RGB-images, a validation set of 10105 synthetic RGB(D)-images and a test set of 10105 synthetic RGB(D)-images and 500 real RGB-images. We then trained RobotVQA with the Stochastic Gradient Descent (SGD) over 3 weeks for 230 epochs with a batch size of 1 and a learning rate of $0.01 \sim 10^{-6}$ on a supercomputer with 32CPUs@(4Ghz Intel, 120GB) for parallel data loading and a single GPU@(NVIDIA, GeForce GTX 1080 Ti with 44GB) for training and inference. ***Notice that we do not test our system on online benchmark datasets***. This is due on the one hand to the fact that the main goal of the paper was to propose a novel architecture to robot vision for manipulation tasks rather than proposing the improvement of an existing algorithm on some datasets. On the other hand, to the best of our knowledge, no online benchmark dataset presents images of objects and annotated with scene graphs such as described in this paper.

B. Results

Remarks on the results above, depicted by figure 9, are manifold. Foremost, RobotVQA demonstrates a good ability to infer, with reasonable computational resources ($\approx 5.5GB$ @ $5fps$), scene graphs from scene images in both RGB as well as in RGBD mode. Secondly, this performance is also observed during the transfer of knowledge from the virtual world to the real world, however performing better in RGB mode than in RGBD mode. This is due to incomplete transfer learning: that is, first layers of Mask-RCNN radically changed in RGBD mode on the one hand and RobotVQA suffers from external depth-related covariate shift in RGBD mode on the other hand (high-resolution depth on training Vs low-resolution depth on testing). Finally, though RobotVQA appears to perform well on the depth and 4D-bounding box estimation, the learning of 3D-orientation of objects converges only very slowly. This is mainly due to self-symmetry of scene objects and only considering the object's Rodrigues axis would solve the problem.

Moreover, conflicts at the network head (e.g. Orientation is rotation-variant while spatial relation is not) might also lead to this issue and readjusting the setup of the multi-tasking learning in RobotVQA would prevent the issue. As actual workable solution to the problem of object orientation estimation, we make use of the object's mask returned by RobotVQA to crop the object's point cloud from the depth image. Then, we apply the principal component analysis (PCA) on this point cloud to determine the three main axes of the object. These axes have demonstrated, as shown by figure 10, to deliver enough information about the object's orientation. Figure 11 demonstrates RobotVQA in real robot scenes.

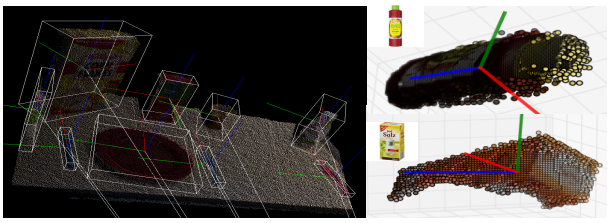

Fig. 10: Approximating object's orientation with pca's main axes.

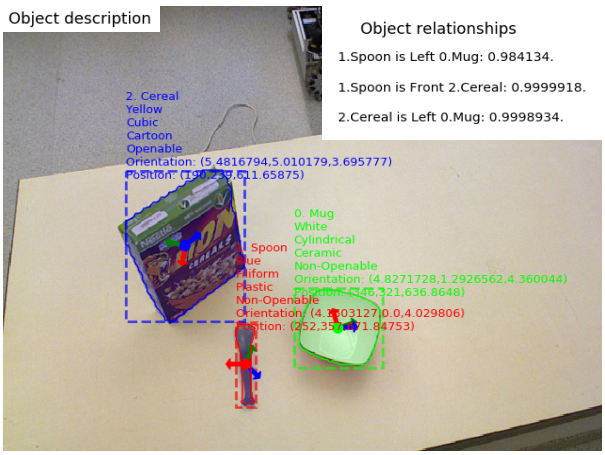

Fig. 11: Semantic graph of a real robot scene. The set of spatial relation is compressed.

For more evidences, we attached a high-resolution demonstration video to this paper.

VIII. CONCLUSION& RECOMMENTATIONS

In this paper, we showed by construction that Deep-Learning and Virtual Reality can be leveraged to build computer vision systems that effectively and efficiently support autonomous robots in the accomplishment of complex manipulation activities in very unconstrained environments while providing the robots with nearly complete structured and probabilistic description of their scene. However, there are still to improve in future works. First, deep grounding of symbols in very unstructured, noisy and dynamic perceptual data still requires fully autonomous acquisition of training data (e.g. automation of virtualization) as well as significantly faster adaptation to new domains (e.g. hierarchical learning). Furthermore, the notion of scene graph, such as presented in this paper, should be extended (e.g. not just spatial relation but deep aggregation). Robot vision is temporally continuous. Therefore, consideration of video rather than image analysis would further reinforce embodiedness. Finally, virtualization should be further advanced in order to integrate significant background noise, providing then more realisticness and enabling efficient knowledge transfer.

ACKNOWLEDGMENT

We sincerely thank the Institute for Artificial Intelligence (IAI) and the Research Project EASE at the University of Bremen for their great support in completing this work. EASE CRC is a Collaborative Research Center mainly hosted at University of Bremen in Germany and working on the research project EASE. EASE stands for Everyday Activity Science and Activity.

REFERENCES

[1] Ferenc Balint-Benczedi and Michael Beetz. "Variations on a Theme: "It's a Poor Sort of Memory that Only Works Backwards"". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*. 2018, pp. 8390–8396. DOI: 10.1109/IROS.2018.8594001. URL: https://doi.org/10.1109/IROS.2018.8594001.

[2] Niklas Barkmeyer. "Learning to recognize new objects using deep learning and contextual information". MA thesis. niklas.barkmeyer@tum.de: Technical University of Munich, Sept. 2016.

[3] Michael Beetz et al. "RoboSherlock: Unstructured information processing for robot perception". In: May 2015, pp. 1549–1556. DOI: 10.1109/ICRA.2015.7139395.

[4] Donald Geman et al. "Visual Turing test for computer vision systems." In: *Proceedings of the National Academy of Sciences of the United States of America* 112 12 (2015), pp. 3618–23.

[5] Kaiming He et al. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[6] Justin Johnson, Agrim Gupta, and Li Fei-Fei. "Image Generation from Scene Graphs". In: *CoRR* abs/1804.01622 (2018). arXiv: 1804.01622. URL: http://arxiv.org/abs/1804.01622.

[7] Franklin Kenghagho Kenfack. *RobotVQA: Scene-graph-oriented Visual Scene Understanding for Complex Robot Manipulation Tasks based on Deep Learning Architectures and Virtual Reality*. Accessed: 2019-02-23. URL: https://github.com/fkenghagho/RobotVQA.

[8] C. Lutz. "Description Logics with Concrete Domains—A Survey". In: *Advances in Modal Logic 2002 (AiML 2002)*. Final version appeared in Advanced in Modal Logic Volume 4, 2003. Toulouse, France, 2002.

[9] Luis A. Morgado-Ramirez et al. "Visual Data Combination for Object Detection and Localization for Autonomous Robot Manipulation Tasks". In: *Research in Computing Science* (2011).

[10] Adam Santoro et al. "A simple neural network module for relational reasoning". In: *CoRR* abs/1706.01427 (2017). arXiv: 1706.01427. URL: http://arxiv.org/abs/1706.01427.

[11] Athanasios Voulodimos et al. "Deep Learning for Computer Vision: A Brief Review". In: *Hindawi Computational Intelligence and Neuroscience* 2018 Article ID 7068349 (2018). URL: https://doi.org/10.1155/2018/7068349.

[12] Danfei Xu et al. "Scene Graph Generation by Iterative Message Passing". In: *CoRR* abs/1701.02426 (2017). arXiv: 1701.02426. URL: http://arxiv.org/abs/1701.02426.