
Open-Ended Learning Approaches for 3D Object Recognition

Hamidreza Kasaei - Cognitive Robotics Course (2021-2021)

<http://www.ai.rug.nl/hkasaei> <https://rugcognitiverobotics.github.io>

Assignment overview

Cognitive science revealed that humans learn to recognize object categories ceaselessly over time. This ability allows them to adapt to new environments, by enhancing their knowledge from the accumulation of experiences and the conceptualization of new object categories. Taking this theory as an inspiration, we seek to create an interactive object recognition system that can learn 3D object categories in an open-ended fashion. In this project, “open-ended” implies that the set of categories to be learned is not known in advance. The training instances are extracted from on-line experiences of a robot, and thus become gradually available over time, rather than being completely available at the beginning of the learning process.

Your goal for this assignment is to implement an open-ended learning approach for 3D object recognition. We break this assignment down into two parts:

1. The first part is about implementing/optimizing offline 3D object recognition systems, which take an object view as input and produces the category label as output (e.g., *apple*, *mug*, *fork*, etc).
2. The second part of this assignment is dedicated to testing your approach in an open-ended fashion. In this assignment, the number of categories is not pre-defined in advance and the knowledge of agent/robot is increasing over time by interacting with a simulated teacher using three actions: `teach`, `ask`, and `correct` (see Fig. 1).

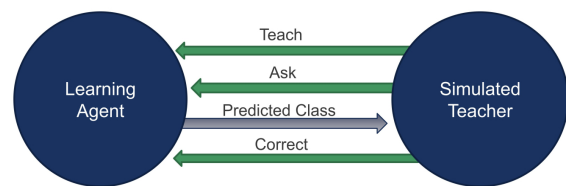


Figure 1: Abstract architecture for interaction between the simulated teacher and the learning agent.

Further details of these assignments are explained in the following sections. To make your life easier, we provide a virtual machine that has all the necessary programs, codes, dataset, libraries, and packages. We also offer template codes for each assignment.

⚠ If you are not familiar with the concept of ROS, please follow the [beginner level of ROS Tutorials](#). For all student, going over all basic beginner level tutorials is strongly recommended.

+ I recommend installing MATLAB on your machine since the output of experiments are automatically visualized in MATLAB. You can download it from [download portal](#) or use [an online version](#) provided by the university.

+ As an alternative, we also provide a python script to visualize the generated MATLAB plots automatically. You can use it as follows: `$ python3 matlab_plots_parser.py -h` to check the instruction

Policies

- Feel free to collaborate on solving the problem but please write your code/report individually. **In particular, do not copy code/text from other students or online resources.**
- **You are not allowed to publish any part of this code online** or claim that you have written it. It does not matter, even if the code is partially used. If you want to publish your results as a scientific paper or use this framework in other projects, contact [Hamidreza Kasaei](#) (hamidreza.kasaei@rug.nl) directly and discuss the case explicitly.

Part I: Offline 3D object recognition setting (50%)

In this assignment, we assume that an object has already been segmented from the scene and we want to recognize its label. We intent to use an instance-based learning (IBL) approach to form new categories. From a general perspective, IBL approaches can be viewed as a combination of an object representation approach, a similarity measure, and a classification rule. Therefore, we represent an object category by storing the representation of objects' views of the category. Furthermore, the choice of the object representation and similarity measure have impacts on the recognition performance as shown in Fig. 2.

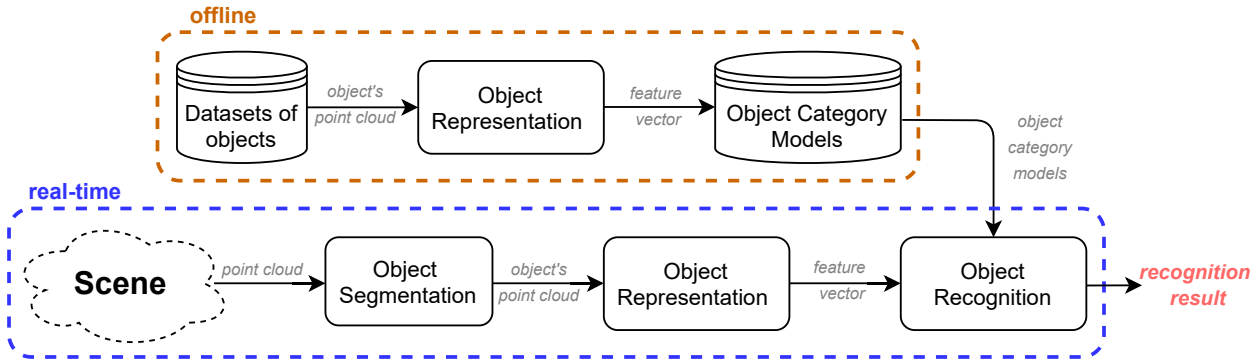


Figure 2: The components used in a 3D object recognition system.

In the case of the similarity measure, since the object representation module represents an object as a normalized histogram, the dissimilarity between two histograms can be computed by different distance functions. In this work, you need to select five out of 14 distance functions that are dissimilar from each other. This policy will increase the chance that different functions lead to different results. The following 14 functions have been implemented and exist in the RACE framework:

Euclidean, Manhattan, χ^2 , Pearson, Neyman, Canberra, KL divergence, symmetric KL divergence, Motyka, Cosine, Dice, Bhattacharyya, Gower, and Sorensen.

✚ For the mathematical equations of these functions, we refer the reader to a comprehensive survey on distance/similarity measures provided by S. Cha (1).

The main intuition behind using instance-based learning in this study is that, IBL serves as a baseline approach for evaluating the object representations used in object recognition. More advance approaches, e.g., SVM-based and Bayesian learning, can be easily adapted.

To examine the performance of your object recognition, we provide a K-fold cross-validation procedure. **K-fold cross-validation** is one of the most widely used methods for estimating the generalization performance of a learning algorithm. In this evaluation protocol, K folds are randomly created by dividing the dataset into K equal-sized subsets, where each subset contains examples from all the categories. In each iteration, a single fold is used for testing, and the remaining nine folds are used as training data. For K-fold cross-validation, we set K to 10, as is generally recommended in the literature. This type of evaluation is useful not only for **parameter tuning** but also for comparing the performance of your method with other approaches described in the literature.

✍ What we offer for this part

- A detail instruction about how to run each of the experiments
- A ROS-based cpp code for 10 fold-cross validation: we have implemented a set of object representation approaches and different distance functions for object recognition purpose. You need to study each approach in depth and optimize its parameters.
- A ROS-based cpp code for 10 fold-cross validation with various deep learning architectures as object representation and a set of distance functions for object recognition purpose. You need to study each approach in depth and optimize its parameters.

- Sample bash scripts for running a bunch of experiments based on GOOD descriptor (hand-crafted), and MobileNetV2 architecture (deep transfer learning), find them out in `rug_kfold_cross_validation/result`).
- A python script to visualize the confusion matrix as the output. Run `python3 matlab_plots_parser.py -p PATH_TO_EXP_DIR/ --offline` to visualize the confusion matrix. You can use `[-h]` to see the instruction.

Your tasks for this part

For this assignment, students will work partly individual and partly in groups of two. Each student needs to optimize one hand-crafted and one deep learning based 3D object recognition algorithm. Therefore, each group will have four set of results. The students will need to write up the report together by discussing the selected approaches and comparing the obtained results in terms of **instance accuracy** ($acc_{micro} = \frac{\# \text{true predictions}}{\# \text{predictions}}$), **average class accuracy** ($acc_{macro} = \frac{1}{K} \sum_{i=1}^K acc_i$), and **computation time**. Note that we need to report average class accuracy to address class imbalance, since instance accuracy is sensitive to class imbalance.

You can think about the following groups:

- (a) **Hand-crafted object representation + IBL approach + K-NN:**
 - list of available descriptors: [GOOD, ESF, VFH, GRSD]
 - distance functions as mentioned above
 - $K \in [1, 3, 5, 7, 9]$
- (b) **Deep transfer learning based object representation + IBL + K-NN**
 - list of available network architectures: [mobileNet, mobileNetV2, vgg16_fc1, vgg16_fc2, vgg19_fc1, vgg19_fc2, xception, resnet50, denseNet121, denseNet169, densenet201, nasnetLarge, nasnetMobile, inception, inceptionResnet]
 - list of available element-wise pooling: [AVG, MAX, APP (append)]
 - distance functions as mentioned above,
 - $K \in [1, 3, 5, 7, 9]$


In this assignment, we use a small-scaled RGB-D dataset to evaluate the performance of different parameter configurations of each approach. In particular, we use **Restaurant RGB-D Object Dataset**, which has a small number of classes with significant intra-class variation. Therefore, it is a suitable dataset for performing extensive sets of experiments to tuning the parameters of each approach.


How to run the experiments

We created a launch file for each of the mentioned object recognition Algorithms. A Launch file provides a convenient way to start up the roscore, and multiple nodes and set the parameters' value (read more about launch file [here](#)).

Before running an experiment, check the following:

- You have to update the value of different parameters of the system in the launch file (e.g., `rug_kfold_cross_validation/launch/kfold_cross_validation_xxxx.launch`)

 You can also set the value of a parameter when you launch an experiment using the following command: `$ roslaunch package_name launch_file.launch parameter:=value` This option is useful for running a bunch of experiments using a bash/python script

 The system configuration is reported at the beginning of the report file of the experiment. Therefore, you can use it as a way to debug/double-check the system's parameters.

For the hand-crafted based object recognition approaches:

After adjusting all necessary parameters in the launch file, you can run an experiment using the following command:

```
$ roslaunch rug_kfold_cross_validation kfold_cross_validation_hand_crafted_descriptor.launch
```

⊕ For the deep transfer learning based object representation approaches:

After adjusting all necessary parameters in the launch file, you need to open three terminals and use the following commands to run a deep transfer learning based object recognition experiment:

≡ MobileNetV2 Architecture

```
$ roscore
$ rosrug_deep_feature_extraction multi_view_object_representation.py mobileNetV2
$ roslaunch rug_kfold_cross_validation kfold_cross_validation_deep_learning_descriptor.launch
orthographic_image_resolution:=150 base_network:=mobileNetV2 name_of_approach:=TEST
```

≡ VGG16 Architecture

```
$ roscore
$ rosrug_deep_feature_extraction multi_view_object_representation.py vgg16_fc1
$ roslaunch rug_kfold_cross_validation kfold_cross_validation_deep_learning_descriptor.launch
orthographic_image_resolution:=150 base_network:=vgg16_fc1 name_of_approach:=TEST
```

✍ What are the outputs of each experiment

- Results of an experiment, including a detail summary, and a confusion matrix (see Fig. 3 and 4), will be saved in: `$HOME/student_ws/rug_kfold_cross_validation/result/experiment_1/`

⚠ After each experiment, you need to either rename the **experiment_1** folder or move it to another folder, otherwise its contents will be replaced by the results of a new experiment.

- We also report a summary of a bunch of experiments in a txt file in the following path (see Fig. 5): `rug_kfold_cross_validation/result/results_of_name_of_approach_experiments.txt`

System configuration:

```
-experiment_name = TEST
-name_of_dataset = Restaurant RGB-D Object Dataset
-number_of_category = 10
-orthographic_image_resolution = 150
-name_of_network = /orthographicNet_service
-base_network = vgg16_fc1
-pooling function [MAX, AVG, APP] = MAX
-distance_function = chiSquared
-pdb_loaded = FALSE
-K param for KNN [1, 3, 5, ...] = 1
-running_a_bunch_of_experiments = FALSE
-other parameters = if you need to define more params, please add them here.
```

No.	object_name	ground_truth	prediction	TP	FP	FN	distance
1	Bottle_Object0.pcd	Bottle	Bottle	1	0	0	3.86
2	Bottle_Object1.pcd	Bottle	Bottle	1	0	0	3.492
3	Bowl_Object0.pcd	Bowl	Bowl	1	0	0	3.405
4	Bowl_Object1.pcd	Bowl	Bowl	1	0	0	3.036
5	Bowl_Object2.pcd	Bowl	Bowl	1	0	0	4.057
6	Mug_Object0.pcd	Mug	Mug	1	0	0	2.927
7	Mug_Object1.pcd	Mug	Mug	1	0	0	3.573
8	Mug_Object2.pcd	Mug	Mug	1	0	0	3.097
9	Mug_Object3.pcd	Mug	Teapot	0	1	1	3.416
10	Flask_Object0.pcd	Flask	Flask	1	0	0	6.648

Figure 3: A detailed summary of an experiment: the system configuration is specified at the beginning of the file. A summary of the experiment is subsequently reported. Objects that are incorrectly classified are highlighted by double dash-line, e.g., No. 9.

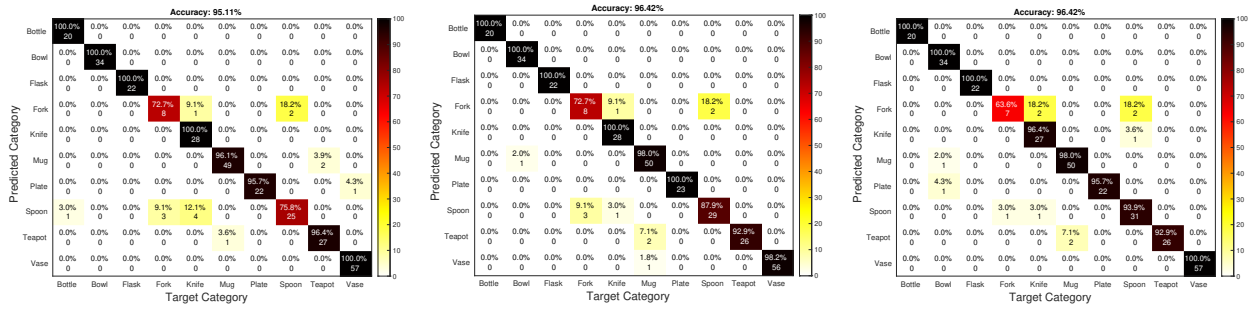


Figure 4: Confusion matrices showing how well each model performed in object recognition task on restaurant object dataset. In each cell of a confusion matrix, we present the percentage and the absolute number of predictions. The darker diagonal cell shows the better prediction by the models.

EXP	Obj.Disc.	#bins	K	Dist.Func.	Ins-Acc	Avg-Class-Acc	Time(s)
1	GOOD	25	1	chiSquared	0.9544	0.9422	2.152
2	GOOD	25	3	chiSquared	0.9381	0.9164	0.9084
3	GOOD	25	5	chiSquared	0.9479	0.9255	1.221
4	GOOD	25	7	chiSquared	0.9381	0.9105	1.273
5	GOOD	25	9	chiSquared	0.9381	0.9105	1.221
6	GOOD	25	1	KLDivergence	0.1303	0.1316	3.035
7	GOOD	25	3	KLDivergence	0.1987	0.2281	1.624
8	GOOD	25	5	KLDivergence	0.1954	0.2246	1.513
9	GOOD	25	7	KLDivergence	0.2117	0.2444	1.619
10	GOOD	25	9	KLDivergence	0.202	0.2273	1.601

Figure 5: A summary of a bunch of experiments for the GOOD descriptor with diffident K and various distance functions: in these experiments, we trained all data first. We then saved the perceptual memory to be used in other experiments.

References

- [1] S.-H. Cha, “Comprehensive survey on distance/similarity measures between probability density functions,” *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307, 2007.