Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Learning Unknown Objects with ARI Robot.

Martin Zderadička

Supervisor: doc. Ing. Tomáš Pajdla, Ph.D.
January 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Zderadička Martin**	Personal ID number:	**499320**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Learning Unknown Objects with ARI Robot

Bachelor’s thesis title in Czech:

Učení neznámých objektů s robotem ARI

Guidelines:

1) Review work [1] on open set recognition and relevant references there, ARI robot [2], and chatbots for robotics [4].
2) Use the method for open set recognition from [1] on ARI [2] robot to detect new objects and ask for their labels via the chatbot [4].
3) Design and execute an experiment demonstrating learning new objects with an ARI robot.

Bibliography / sources:

[1] N. Sokovnin. Recognizing Unknown Objects for Open-Set 3D Object Detection. BSc Thesis, FEL CVUT in Prague. 2021.
[2] ARI Robot. Pal Robotics. Available at https://pal-robotics.com/robots/ari/
[3] Stanford Artificial Intelligence Laboratory et al., 2018. Robotic Operating System, Available at: https://www.ros.org.
[4] N. Gunson, D. Hernández García, W. Sieińska, C. Dondrup, O. Lemon. Developing a Social Conversational Robot for the Hospital waiting room. 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), https://drive.google.com/file/d/1AJ4KnLFdBjhKQCIuFXzJrpil9BIBidVr/view

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Tomáš Pajdla, Ph.D. Applied Algebra and Geometry, CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **19.02.2023** Deadline for bachelor thesis submission: **09.01.2024**

Assignment valid until: **22.09.2024**

doc. Ing. Tomáš Pajdla, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student’s signature

Acknowledgements

I would like to thank my thesis supervisor doc. Ing. Tomáš Pajdla, Ph.D. for guiding me through this thesis and providing me with new perspectives on the problem as well as introducing me to the problem itself and providing me access to new learning opportunities. I would also like to thank CIIRC for providing me with the environment and resources to work on this thesis. Namely, I would like to thank Ph.D. Michal Polic for his support and guidance in navigating this task. As well as Ing. Rakshith Madhavan, a former CVUT student and CIIRC employee who has helped me numerous times with addressing the technical nuances of the ARI robot.

Declaration

Abstract

This bachelor's thesis investigates the task of enabling the ARI humanoid robot to learn and identify new objects using basic concepts from machine learning and computer vision. The study revolves around developing and implementing a straightforward 3D object detection and classification pipeline, with the aim of enabling the robot to recognize objects it has not previously encountered. The approach integrates fundamental aspects of open set recognition and incremental learning, focusing on the application of these techniques in a practical setting with the ARI robot.

The effectiveness of the implemented system is assessed through a series of elementary experiments, concentrating on its ability to detect and categorize new objects. These initial tests provide insights into the basic functioning of the system and its potential utility in a controlled environment.

This thesis contributes to the field of robotics at an introductory level, presenting an initial exploration into the use of machine learning and computer vision in a practical robotic context. It lays the groundwork for future research in the area of robotic object recognition.

Keywords: Humanoid Robotics, Machine Learning, Computer Vision, Object Detection, Object Classification, Open Set Recognition, Incremental Learning, ARI Robot, 3D Object Recognition, Robotic Learning

Supervisor: doc. Ing. Tomáš Pajdla, Ph.D.

Abstrakt

Tato bakalářská práce se zabývá úkolem umožnit humanoidnímu robotu ARI učit se a rozpoznávat nové objekty pomocí metod strojového učení a počítačového vidění. Práce se zabývá vývojem a implementací systému pro detekci a klasifikaci 3D objektů, s cílem umožnit robotu rozpoznat objekty, se kterými se dříve nesetkal. Přístup integruje základní aspekty rozpoznávání ve světě bez omezení a inkrementálního učení, zaměřuje se na aplikaci těchto technik v praktickém prostředí s robotem ARI.

Účinnost implementovaného systému je hodnocena prostřednictvím série základních experimentů, zaměřujících se na jeho schopnost detekovat a kategorizovat nové objekty. Tyto počáteční testy poskytují náhled do základního fungování systému a jeho potenciálního užití v kontrolovaném prostředí.

Tato práce přispívá do oblasti robotiky na úvodní úrovni, představuje počáteční průzkum využití strojového učení a počítačového vidění v praktickém robotickém kontextu. Předkládá základy pro budoucí výzkum v oblasti robotiky a rozpoznávání objektů.

Klíčová slova: Humanoidní robotika, Strojové učení, Počítačové vidění, Detekce objektů, Klasifikace objektů, Rozpoznávání ve světě bez omezení, Inkrementální učení, Robot ARI, Rozpoznávání objektů ve 3D, Učení robotů

Překlad názvu: Učení neznámých objektů s robotem ARI

Contents

Figures

Chapter 1

Introduction

In recent years, the integration of robotics into various aspects of daily life has significantly accelerated, transforming mundane tasks into automated processes. One of the most intriguing developments in this domain is the advancement of humanoid robots, exemplified by the ARI robot. Humanoid robots like ARI have begun to move beyond their traditional roles in industrial settings, making inroads into more dynamic and unpredictable environments such as healthcare, hospitality, and domestic assistance.

This evolution of robotic applications necessitates a shift from pre-programmed functionalities to adaptive learning capabilities. The ability of a robot to recognize and learn about unknown objects in its environment is paramount for efficient and intelligent interaction in real-world scenarios. This thesis modestly contributes to this evolving field by exploring the potential of the ARI humanoid robot to learn and classify unknown objects, employing machine learning techniques that bring together aspects of computer vision and natural language processing.

1.1 Motivation and Goals

This thesis represents a step forward from a preceding work [1] focused on detecting unknown objects, aiming to apply it in the real world and enhance the ARI robot's ability not just to detect, but also to learn and classify these objects. Embracing this challenge serves a dual purpose: it contributes modestly to the field of robotics and plays a crucial role in my academic and professional development. As a student, engaging in this project has been instrumental in understanding the collaborative nature of engineering and the importance of building upon existing work. This process of learning, adaptation, and contribution is essential in shaping a thoughtful and resourceful engineer.

The specific goals of this thesis are:

- To explore and implement a system for the ARI robot, enhancing its ability in open set recognition, enabling it to recognize and adapt to new object classes it encounters.

- The system will incorporate methodologies that allow the ARI robot to continuously expand its object knowledge base in a structured and effective manner.
- The system will leverage NLP techniques to meld visual data with linguistic elements, enriching the robot's comprehension and classification capabilities of objects.

Chapter 2

Related work

This section presents a review of related work, highlighting significant research and advancements that inform and contextualize our study

2.1 Open set recognition and Open set detection

Open-set detection stems from open-set recognition, which as described in [49] deals with developing models which are are able to recognize unknown classes of objects, that weren't present in their training dataset. To measure the capability of these models [50] introduced open set risk, a measure of the error of mistaking unknown objects for known ones. [45] recognizes two types of approaches to minimizing open set risk, discriminative and generative. While discriminative approaches rely on tight confinement of known classes feature representation, generative approaches augment the training dataset with artificially generated data, to improve its recognition capabilities in diverse environments. The former is the more widely adopted one.

Open-set detection is a domain of open-set recognition concerned with object detection. [47] proposed open-set detection as a means to address the brought-up issue of over-confident false positive detections by conventional closed-set object detectors. In [1], inconsistencies in the classifications of these false positives when observing the same object from different perspectives were observed and used to design an open set detector that relies on false positives for the detection of unknown objects.

2.2 Open world detection, incremental learning

Open-world detection incorporates the process of continually extending the set of known objects by learning to classify the detected unknown objects, perhaps by external data annotation by an oracle. This closely resembles the problem of incremental learning, which is all about incrementally teaching the model new capabilities, while also retaining the old ones, [51] described three types of incremental learning: task-incremental learning, domain-incremental learning, and class-incremental learning [52], class-incremental learning bear-ing the closest similarity to open world detection is defined by the ability to learn new classes. It's recognized as the most challenging of the three as learning to distinguish a new class requires putting it into the context of all already known classes.

[46] put forth the distinction between class-incremental learning and open-world detection, that is, the ability to recognize unknown classes and distinguish them, which steps up the difficulty as it requires with each learned class, the ability to put in the context of it something never seen before. Among methods utilized in [46] to tackle this issue, the most relevant to our work is the use of contrastive clustering as a training objective.

2.3 Natural language processing, Multimodal models

The progression of Natural Language Processing has seen a shift from focusing solely on text to embracing multimodal approaches. Early developments in word embeddings [5, 4, 6, 7] and sequence models [8, 10] set the stage for this transition. Similarly, advancements in image processing models like VGG, ResNet, and ViT [11, 12, 13] paralleled these efforts. The culmination of these developments is seen in models like CLIP [15], which integrate language and visual data, leading to the emergence of Multimodal Large Language Models (MMLLMs)[19, 53]. These models, while still not well understood, exhibit promising capabilities in visual reasoning tasks by blending textual and visual understanding.

Chapter 3
Setup

This section is dedicated to offering a concise yet comprehensive overview of the foundational prerequisites to our work.

3.1 Ari

ARI, developed by PAL Robotics, is a humanoid robot platform designed primarily for Human-Robot Interaction (HRI) and front-desk tasks. It is equipped with a range of features for multimodal expressions and gestures, including a touchscreen, gaze control, and an LCD display for eyes, enhancing its interactive capabilities. The combination of an Intel i7 processor and NVIDIA Jetson TX2 GPU provides the necessary processing power for AI-driven tasks such as navigation, manipulation, perception, and speech recognition. ARI is intended for use in controlled environments under supervision. Its functionalities are accessible through a user-friendly web interface and an extensive ROS API, supporting both customization and application development.

3.1.1 Body

The ARI robot features a neck with two degrees of freedom, arms with four degrees of freedom, and hands with a single degree of freedom, providing a range of motion for interactive tasks. Mobility is facilitated through two drive wheels, and the unit is powered by an internal battery pack that supports its autonomous operations.

3.1.2 Cameras

While our work primarily utilizes the front torso RGB-D camera for depth perception, ARI is equipped with a range of cameras, each serving distinct purposes, ensuring comprehensive visual capabilities for complete environmental awareness.

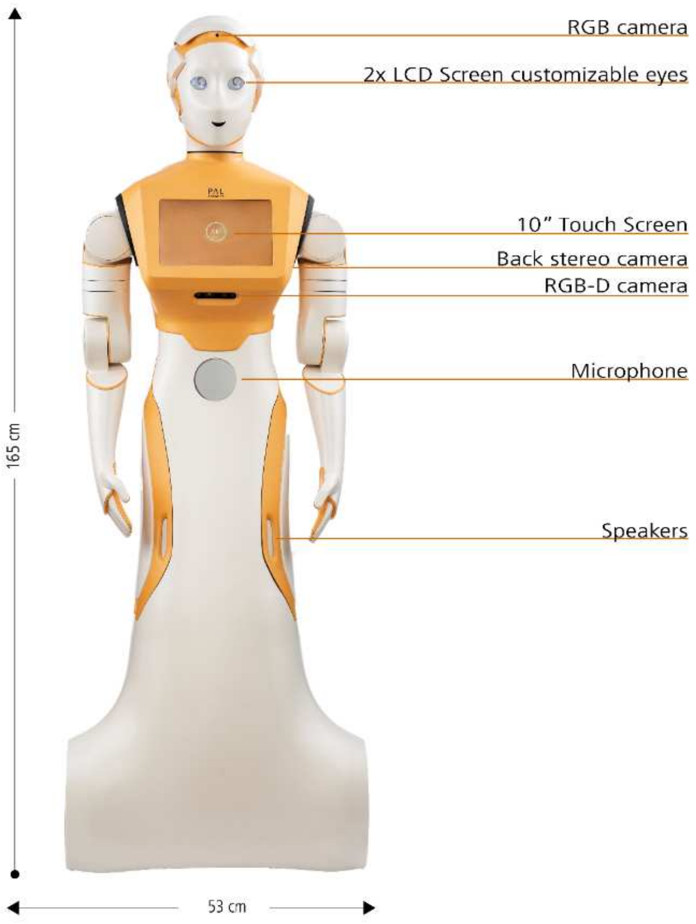


Figure 3.1: the ARI robot

Front torso stereo RGB-D camera

Mounted on the torso, below the touchscreen, ARI's front-facing Intel RealSense D435i provides RGB and depth imaging capabilities. It supports depth image resolutions up to 1280 x 720, although for our applications, we utilize a setting of 640 x 480. This camera is complemented by an integrated IMU, enabling the monitoring of inertial forces and altitude within the robot's environment.

Frontal and back stereo-fisheye cameras

Frontal and rear stereo-fisheye cameras are placed at both the front and back sides of ARI's torso. These cameras deliver stereo images at 30 frames per second, providing expansive field-of-view coverage in both fisheye and black-and-white formats. Each camera is equipped with an IMU sensor.

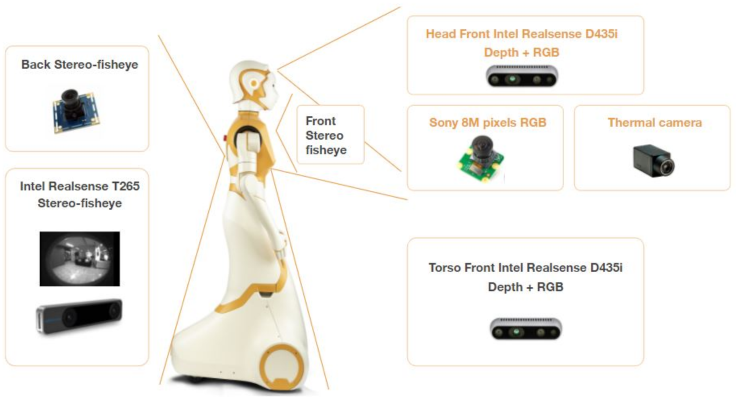

Back Stereo-fisheye
Intel Realsense T265 Stereo-fisheye
Front Stereo fisheye
Head Front Intel Realsense D435i Depth + RGB
Sony 8M pixels RGB
Thermal camera
Torso Front Intel Realsense D435i Depth + RGB

Figure 3.2: Placement of ARI's cameras

Head camera

Positioned on the head, there is an RGB camera capable of capturing high-quality color images.

3.2 ROS

The Robot Operating System (ROS)[44] serves as the foundation for managing sensory data acquisition and processing within the ARI humanoid robot. It operates through a network of nodes communicating via topics, services, and actions, which establishes a modular and scalable system architecture. This setup is essential for the effective coordination and utilization of sensory inputs in our machine-learning task.

ROS also provides a useful set of tools, including RViz, a visualization tool that is particularly beneficial for monitoring and debugging various aspects of the robot's operation, and ROS bags, a feature of ROS that enables the recording and playback of data streams into and from rosbag(.bag) files, allowing us to replay sensor and state data as if they were being transmitted live from the robot, facilitating testing and analysis.

ROS also provides a comprehensive system for managing coordinate frames and transforms, essential for accurately interpreting the spatial relationships and movements within the robot's environment. This system creates a transform tree that maintains the hierarchy and relationships between various coordinate frames. Thanks to this transform tree, determining transforms from one coordinate frame to any other within the tree is simplified; we only need a single transform between that specific frame and any other frame already in the tree.

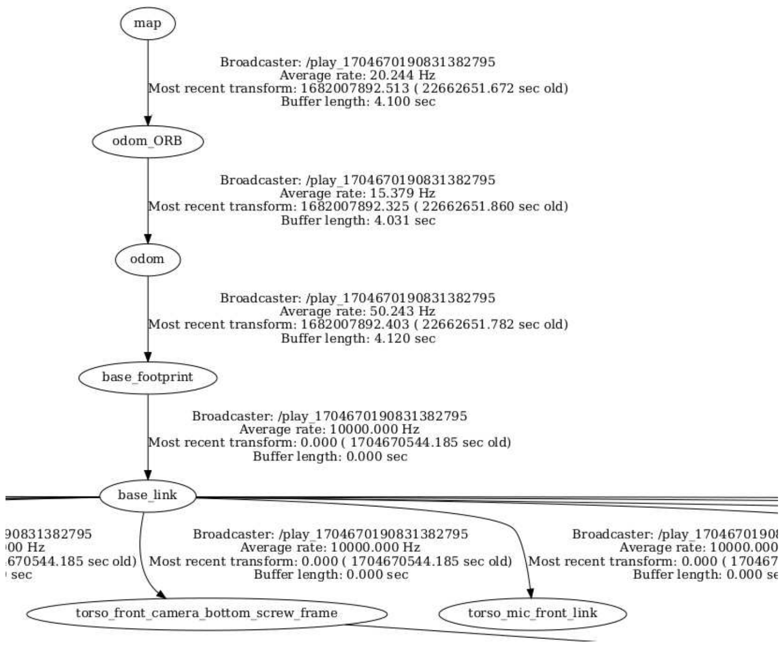

map
Broadcaster: /play_1704670190831382795
Average rate: 20.244 Hz
Most recent transform: 1682007892.513 (22662651.672 sec old)
Buffer length: 4.100 sec
odom_ORB
Broadcaster: /play_1704670190831382795
Average rate: 15.379 Hz
Most recent transform: 1682007892.325 (22662651.860 sec old)
Buffer length: 4.031 sec
odom
Broadcaster: /play_1704670190831382795
Average rate: 50.243 Hz
Most recent transform: 1682007892.403 (22662651.782 sec old)
Buffer length: 4.120 sec
base_footprint
Broadcaster: /play_1704670190831382795
Average rate: 10000.000 Hz
Most recent transform: 0.000 (1704670544.185 sec old)
Buffer length: 0.000 sec
base_link
Broadcaster: /play_1704670190831382795
Average rate: 10000.000 Hz
Most recent transform: 0.000 (1704670544.185 sec old)
Buffer length: 0.000 sec
torso_front_camera_bottom_screw_frame
torso_mic_front_link

Figure 3.3: An illustrative art of a transform tree

3.2.1 ROS on Ari

ARI offers a ROS API that facilitates control and access to its diverse sensor data. This includes, but is not limited to, camera images, IMU readings, and calibration data, all of which are transmitted to their own ROS topics. For brevity, only the relevant topics will be listed.

- /torso_front_camera/color/image_raw: This topic broadcasts 640x480 resolution RGB images from the front torso RGB-D camera.
- /torso_front_camera/color/image_raw/compressed: Here, the RGB image data from the front torso RGB-D camera is transmitted in a compressed format, optimizing for reduced file size, this is mostly useful when recording ROS bags.
- /torso_front_camera/aligned_depth_to_color/image_raw: This topic provides depth data images from the front torso RGB-D camera, aligned with the corresponding RGB images for cohesive depth and color information.
- /torso_front_camera/color/camera_info: This contains the intrinsic parameters of the front torso RGB-D camera, essential for understanding and processing the camera's visual data.

Important coordinate frames within ARI's transform tree are the following:

- odom: This frame represents ARI's estimated position change based on its odometry.

- map: This is the global coordinate frame of ARI's environment, established by the localization module.
- base_link: This frame serves as the primary reference point for ARI, linking its body to the coordinate system.
- torso_front_camera_color_optical_frame: This frame denotes the specific coordinate system of the front torso RGB-D camera.

Localization

Mapping, the process of constructing ARI's environmental map, and localization, the process of estimating ARI's position within this map, have been implemented using rtabmap[54] by Ing. Rakshith Madhavan (refer to acknowledgments). The localization ROS node contributes the map coordinate frame and its transform to the transform tree. Additionally, it broadcasts ARI's position and orientation to the /robot_pose/ topic.

3.3 The foundational 3D open set detector

Our work builds upon [1], where an open set 3D object detector was designed and implemented. This section will briefly describe the implementation details of this work, as it provides crucial context for the discussion of our work.

This detector is implemented as a multi-view meta-classifier, multi-view meaning that it uses multiple 2D images (or views) of the detected objects taken from different perspectives when classifying objects, and meta-classifier meaning that the classification is decided by reasoning about the accumulated outputs, also referred to as "feature vector", of another backbone classifier. Refer to [1] for a formal definition of a meta-classifier

The detector takes as input a series of RGB images $(x_1, ...x_M)$ and their corresponding depth data $(d_1, ...d_M)$ and a series of corresponding camera poses w.r.t. a global coordinate system $((R_1, t_1), ...(R_M, t_M))$ where $R_i \in \mathbb{R}^3$ is the rotation matrix of the camera pose and T_i is the camera's coordinate vector. The detector processes these series one by one and at each step updates its internal state, which contains 3D coordinates of objects' centroids and their respective classifications and confidence scores. Each component of this detector will be described below.

3.3.1 Yolact

Yolact[28, 29] is a real-time instance segmentation CNN[43] model which serves as the detector's backbone for object detection and meta-classification. Given an image x_t it outputs, per each object detection $o^i_{x_t}$, the following:

- $m_{o^i_{x_t}} \in \{0, 1\}^{H \times W}$: The segmentation mask of the detected object, i.e. the set of pixels which are a part of that object.

- $s_{o^i_{x_t}} \in \mathbb{R}^N$: The softmax distribution of classification scores.
- $l_{o^i_{x_t}}$: the classification label
- $sc_{o^i_{x_t}} \in [0,1]$: the classification confidence score

Where $H, W \in \mathbb{N}$ are the height and width of the image respectively, and $N \in \mathbb{N}$ is the number of classes that Yolact is trained to recognize. In our case $H = 480, W = 640$ and $N = 80$ Additionally, [1] includes $c_{o^i_{x_t}} \in \mathbb{N}^2$, which is the pixel coordinate of the centroid (center of mass) of $m_{o^i_{x_t}}$

3.3.2 Centroid back-projection

For each image, the object detections $(o^1_{x_t}, ...o^D_{x_t})$ are associated with 3D coordinates $(v_{o^1_{x_t}}, ...v_{o^D_{x_t}})$ w.r.t the global coordinate frame, where $v_{o^i_{x_t}}$ is obtained by back-projecting $c_{o^i_{x_t}}$ as described below.

Point back-projection

For brevity, let's denote a general case, where indices of variables will not be denoted. Given pixel coordinates $p = [x, y]^T$, its corresponding depth data z a rotation matrix R, camera position T and the camera's intrinsic matrix K (it represents the transformation from 3D points in the camera's coordinate frame to their respective pixel coordinates in the image x). Where K is defined by:

$$\begin{bmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y represent the focal length, p_x and p_y represent principal point and s represents the skew coefficient between x and y axis. Note that normally K would be a 3-dimensional square matrix, but here it's extended to accommodate the use of homogeneous coordinates, which allow for the representation of affine transformations in matrix form by using the 4th coordinate for translation. This will be important in the construction of the camera's extrinsic matrix E as it uses homogeneous coordinates and we intend on composing E with K We'll construct the camera's extrinsic matrix E (it represents the transformation from the global coordinate frame to the camera's coordinate frame) like so:

$$\begin{bmatrix} \mathbf{R}^T & \mathbf{C} \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{C} = -\mathbf{R}^T\mathbf{T}$ Finally, we compute the 3D coordinates $[x_w, y_w, z_w]$ of p like so:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{K}^{-1} \begin{bmatrix} x \cdot z \\ y \cdot z \\ z \\ 1 \end{bmatrix}$$

3.3.3 Object representation and clustering

Objects are for simplicity represented as spheres of constant radius r, which is the detector's parameter. The detector maintains a set W of candidate object coordinates, referred to as cluster centers because they are determined by clustering the points $v_{o^i_{x_t}}$ in each time step t. This is done by searching for the closest point $w \in W$, if its distance is less than r then it is clustered together and w is replaced by the average of all points that were clustered to the cluster whose center was w, if its distance is greater than r, then it becomes the center of a new cluster and is inserted into W. The following pseudocode describes this process in more detail:

Algorithm 1 Clustering method pseudocode as described in [1]

```
1:  **Result**: Clusters with assigned labels
2:  $enm \leftarrow$ init_yolact()
3:  $env \leftarrow$ init_env()
4:  $clusters \leftarrow \{\}$
5:  $episode\_over \leftarrow$ False
6:  while not $episode\_over$ do
7:      $image \leftarrow env.step()$
8:      $detections \leftarrow cnn(image)$
9:      $centroids \leftarrow$ compute_centroids($detections$)
10:     $assigned \leftarrow$ False
11:     for ($centroid, detection$) in zip($centroids, detections$) do
12:         $world\_position \leftarrow$ backprojection($centroid$)
13:         $cluster \leftarrow$ find_corresponding_cluster($world\_position, clusters$)
14:         if $cluster$ is $None$ then
15:             $cluster \leftarrow$ init_cluster($world\_position$)
16:             $clusters.append(cluster)$
17:         else
18:             $cluster.update\_center(world\_position)$
19:         end if
20:         $cluster.append(detection)$
21:     end for
22: end while
```

3.3.4 Cluster classification

[1] tested two possible methods for making a classification decision. Confidence thresholding over the accumulated detection scores (feature vector), also referred to as "maxsum" and entropy measurement over the mean softmax vector (mean over the accumulated detections). In our work, we consider only the first approach, as both methods resulted in similar results on real-world data and our chosen approach would not be compatible with the second method. The confidence thresholding approach is as follows:

- From the set of cluster's associated object detections O a feature vector $a \in \mathbb{R}^{\mathbb{N}}$ (N is the number of classes known to Yolact) is constructed such that $a_k = \sum_{o \in O, l_o = k} o_{sc}$
- A vector $v = \frac{[a_1, \dots, a_N]^T}{\sum_{i=1}^{N} a_i}$ is constructed
- Classification is decided by

$$L^*(v) = \begin{cases} K_{\arg\max(v)}, & \text{if } \max(v) > \theta_1 \\ \text{unknown}, & \text{otherwise} \end{cases}$$

  where $\theta_1 \in [0, 1]$ is the confidence threshold, which is the detector's parameter.

Chapter 4
Approach

This chapter documents our approach to implementing a solution to the assignment.

4.1 Yolact 3D on Ari

This section details the adaptation of Yolact 3D, as outlined in [1], for practical deployment on the ARI robot in real-world settings. While Nikita's initial work was a critical foundation, it primarily operated within simulated environments. Transitioning to real-world applications presented unique challenges, mainly due to the unpredictable and diverse nature of real-world data. To address these, specific modifications and optimizations were required, which are elaborated in the following subsections.

4.1.1 Caveats and adaptations

Implementing Yolact 3D in a real environment necessitated overcoming challenges chiefly arising from the inherent uncertainty in real-world measurements. This uncertainty led to a deviation from the methods outlined in [1], demanding a more robust and adaptable approach. The key adaptations are delineated in the following segments, highlighting our tailored strategies to enhance practical applicability.

4.1.2 Determining detections 3D position

[1] relied on the 3D back-projection of 2D centroids of detected object instances pixel masks. Two issues were encountered:

Firstly, the pixel associated with the 2D centroid, may not lie inside its pixelmap, given a nonconvex object, the resulting 3D thus may not correspond to the real object's position at all.

Second, the depth camera available (Intel® RealSense™ Depth Camera D435i) could not consistently provide depth data for each pixel of its output frame, even if in range compliant with the camera specifications (28-200cm).

As a result of this, depth data for an object's centroid pixel may not be available even if it lies inside its segmentation pixel map, rendering the back-projection impossible.

To address the issue, the strategy of determining the object's 3D position was changed from the back-projection of the 2D centroid of an object's pixel mask to taking the centroid of 3D backprojections of pixels from the pixel mask for which depth data was available.

This solution introduced another issue, the segmented pixelmaps, especially for low confidence detections, which [1] relies on, often, due to error also include pixels in the object's vicinity. This can have a significant impact on the position of the 3D centroid as a multitude of outliers are introduced to the back-projected point cloud.

To counter this, an outlier filtering method /citeoutlier is used in the process. Note that this approach is still subject to failure if the segmentation error is high enough.

4.1.3 Object representation

With the aforementioned change to determine the 3d position object detections a more natural approach to accumulating those detections into a single object representation comes to mind, that is, representing the object as a point cloud constructed by merging the back-projected pixel mask clouds. Clustering appropriate detections together would then instead of centroid proximity rely on a density-based point cloud clustering algorithm, in our case dbscan[41].

This approach, in theory, provides a more robust solution capable of handling objects of radically different sizes and arranged in close proximity. However, in practice, it proved impractical, as the error in the robot's localization was significant enough to cause misalignment between different detections of the same object.

Instead, we opted for a similar approach to the original one, which is more robust to inaccuracies in object positions. This approach is only different in the way in which a new object centroid is determined after merging detections. Previously, this was done by taking the average of the detections of centroids, which can shift the resulting position towards that side of the object, from which it has been observed the most. Our approach is to take the average of the detection's point clouds after merging and downsampling with a voxel grid filter, this way the downsampling step takes care of overlapping detections.

4.1.4 Yolact3d ROS node

The classifier's implementation was, with the help of Ing. Rakshith Madhavan (see acknowledgments), encapsulated within a ROS node, and designed to integrate seamlessly with ARI's existing system architecture. This node subscribes to the RGB-D camera's depth and image topics, as well as the localization module's pose topic "/robot_pose", and publishes the resulting poses and classification scores of detected objects.

4.1.5 Experiments

The performance of the detector was then evaluated in an experimental setting in a controlled environment. (see 5.1)

4.2 Formalizing the Pipeline

This section aims to abstractly delineate the system's structure, facilitating initial reasoning about its design without delving into the complexities of its physical implementation.

The system is conceptualized as a tuple (S, O, M, R, D, s, o, r, d, g, l), with each element representing a distinct component or function within the pipeline. This formalization serves as a foundational framework, guiding the implementation and ensuring clarity in the system's functional dynamics. See 4.1

- S the set of all possible data given the system's sensors
- O the set of all possible observations given a single observed unit of data
- M the set of all memory states, this is where the information about the observed world as well as oracle input is accumulated.
- R the set of all scene representations, this is where the model constructs an internal representation of its surroundings from the observed data.
- D The set of all object descriptions, the purpose of those is to have an interface through which the input and output of information about individual detected objects can occur.
- $s : \mathbb{R} \longrightarrow S$ The sensing function, it encapsulates the process of collecting sensor data from (approximately) one point in time
- $o : S \longrightarrow O$ The observation function, it encapsulates all processing done on data from just one point in time
- $r : R \times M \times O \longrightarrow R \times M$ The scene representation function, it encapsulates the process of constructing the model of the system's immediate surroundings as well as the knowledge base of the world it is situated in

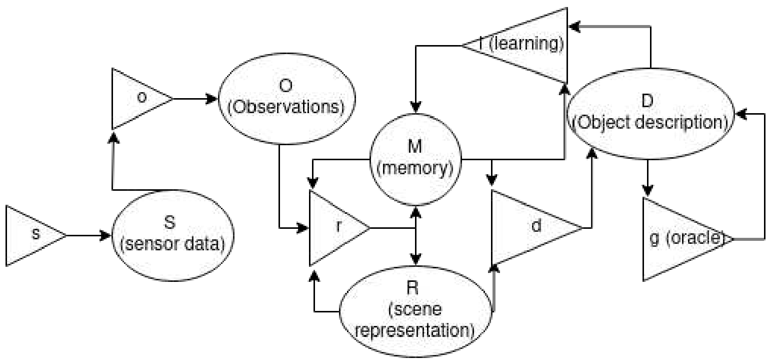


Figure 4.1: Diagram of the pipeline's formal description

- $d : R \times M \longrightarrow \cup_{n=0}^{\infty} D^n$ The object description function. Since R and M are arbitrary, it is not guaranteed that individual objects will be explicitly represented in either of them, the object description function encapsulates the process of extracting these explicit representations per each instance of an object detected.

- $g : D \longrightarrow D$ The ground truth (oracle) function. It encapsulates the process of obtaining the ground truth from the oracle.

- $l : M \times D \longrightarrow M$ The learning function. It encapsulates the process of applying the results from the oracle such that future predictions will be in better alignment with the ground truth provided by the oracle, what it means to be in better alignment is implementation-specific.

4.2.1 Similarity to the PAC framework

The Probably Approximately Correct (PAC) learning framework, as formalized in [2], is widely recognized in machine learning for conceptualizing the learning process as an optimization task. It emphasizes minimizing the error between a model's output (hypothesis mapping) and the ground truth (concept mapping).

This framework evaluates the conditions under which a concept can be learned with a specified accuracy, based on a finite set of examples. Similar to our approach, the PAC framework uses ground truth annotations to guide the learning process and focuses on accurate classifications. While our model emphasizes an incremental approach, adapting continuously to new information, it shares with the PAC framework the core objective of refining classification accuracy through learning.

While our focus isn't on the accuracy guarantees of the PAC framework, we draw inspiration from its methodology of deconstructing complex problems into clear, manageable components.

4.3 Implementing the pipeline

In this section, we delve into the practical aspects of designing and implementing each element of the formal pipeline, as defined in the previous section. Central to this undertaking is the challenge of data scarcity, which is inherent to the system, since the robot has to collect the data it is supposed to learn from and any realistic environment where the robot will operate will be limited in the number of present instances of objects.

There are a couple of ways to approach this, one is through few-shot deep learning, this approach relies on a classifier architecture, which once pre-trained on a large dataset, can utilize its intrinsic knowledge to extend its label set given just a few samples of data per new class, these methods may be prone to what is known as "catastrophic forgetting", which is the inability to maintain original performance on old classes after introducing enough new ones.

There are some models in the incremental learning literature that are designed to combat this issue /citefew-shot-incremental, however, it is not known, whether such model architectures would exhibit the same properties as were observed in [1] and maintain them throughout the process of extending the set of known classes. For that reason, we opted for another approach, which will be described in the following section

4.3.1 Language-aligned embeddings as the basis for semantic understanding

Instrumental to our approach to tackling the scarcity of learning data is comparing unknown objects to known ones in terms of semantic similarity. This is an approach natural to humans, who instinctively associate unknown objects with known ones that exhibit similar features [3].

vector space embeddings

A widely used approach to encode the features descriptive of the semantics of a concept is to use vector space embeddings, which is a mapping from given data to an Euclidian space $\mathbb{R}^n$ (also often referred to as a latent space). These embeddings became prominent in the field of natural language processing, where they were first used to represent the semantic meaning of individual words [5, 4, 6, 7] these methods were among the first to introduce the mapping of concepts to the latent space, such that the semantics structure was preserved in the latent space in a meaningful manner. For example, the vector connecting the embeddings for "queen" and "king" would be very close to the vector connecting the word embeddings for "man" and "woman" (4.2).

Methods for embedding the semantic meaning of entire sequences of words,

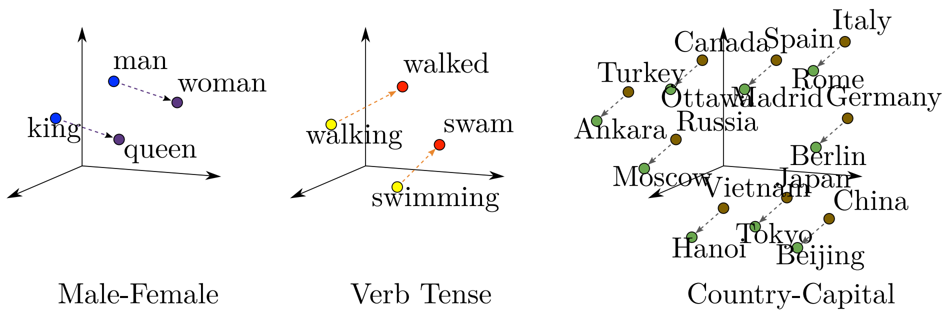


Figure 4.2: Visualization of the semantic structure captured in word embedding latent space (https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space)

beyond individual words, were also developed [8, 10] most image classification models also rely on mapping the image data into lower-dimensional arrays [11, 12, 13], this could also be seen as a latent space mapping (up to an isomorphism), these methods (both for images and sequences of words) do not produce a structure in the latent space, which would produce examples as intuitive as the previous one, however, they still preserve the semantic structure in a meaningful way, such that semantically similar sentences, or images also produce similar embeddings. (4.3)

Such methods, when trained on sufficiently sized datasets, exhibited great generalization capabilities, that is, the produced embeddings corresponded very well with the underlying semantic meaning behind given data and weren't very much affected by nuanced shifts in the data.

These capabilities are well suited for our application, since being able to tell, which of the objects of unknown class share the underlying semantic meaning, which is closely associated with the classification of that object, seems promising as the mechanism for connecting the information gained from the oracle function g to already observed data in M.

The process of generalizing concepts learned from training data to novel, but similar concepts is called zero-shot learning. Many models, which were pre-trained on large datasets with a wide range of concepts [15, 14, 16] have demonstrated zero-shot generalization capabilities. CLIP [15] is among the more widely known ones

CLIP

CLIP [15] is a famous multimodal embedding model, which maps both sequences of words as well as images into the same latent vector space.

This was achieved by pretraining two encodings (embedding) models (a Transformer [17] for encoding text sequences and a Visuion Transformer (ViT) [13] for encoding images) on a large dataset of image-description pairs scraped from the internet, optimizing a contrastive learning task, where the

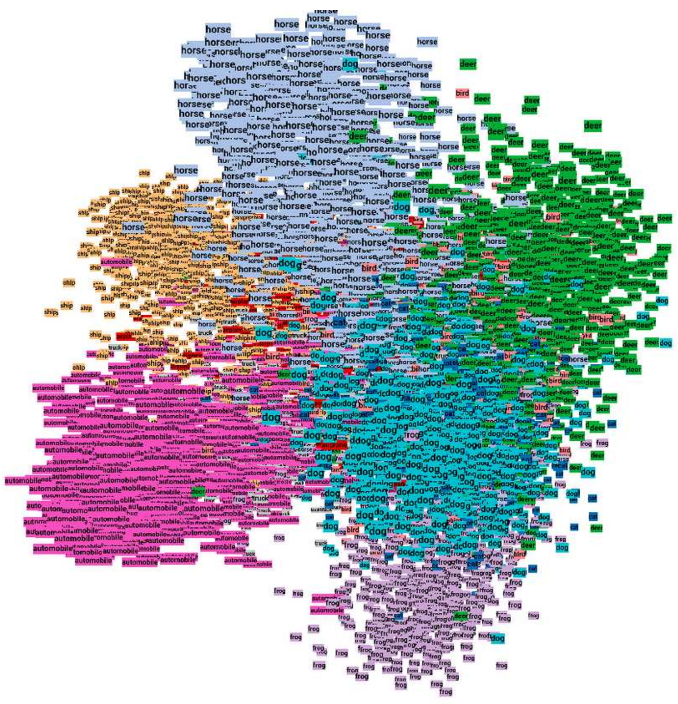


Figure 4.3: Visualization of the semantic structure captured in image embedding latent space it is the low dimensional projection of embeddings of images of animals (grouped by color) (https://github.com/smivv/tensorflow-cifar-10)

objective is to, given batches of N pairs of images and text descriptions and their encodings, maximize the cosine similarity between the encodings of the N correct image-text pairings and minimize the cosine similarity between the encodings of $N^2 - N$ image-text pairings.

As mentioned before, CLIP demonstrated generalized zero-shot learning capability, or as referred to in the paper: zero-shot transfer, where by comparing cosine similarity between the encodings of an arbitrary set of labels and the encoding of an image, CLIP could be without any modification or fine-tuning used for the task of image classification, despite being trained for a different task. It achieved 76% accuracy on the Imagenet classification task. These results mark CLIP as a promising solution to the problem of deriving some sort of semantic understanding from the observed data and comparing them. (Since all images of a class of object are similar to the class label, they must therefore be close to one another)

In the literature, there are multiple instances of CLIP being used to derive semantic meaning from images for a wide array of applications, such as scene understanding, semantic segmentation, text-to-image generation, and even visual commonsense reasoning. [18, 19, 20, 21, 22]

However, it is not clear how well these results will transfer to the data

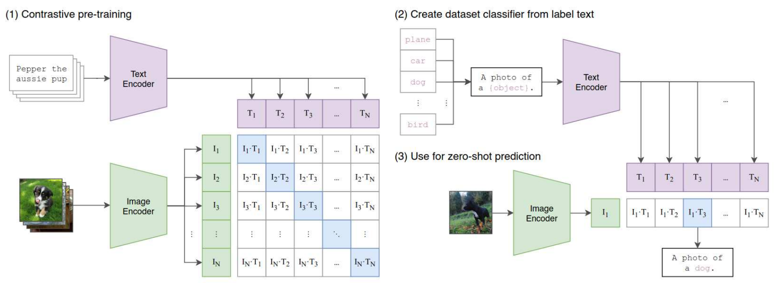

(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
(3) Use for zero-shot prediction
A photo of a dog.

Figure 4.4: Visual overview of CLIP approach (from [15])

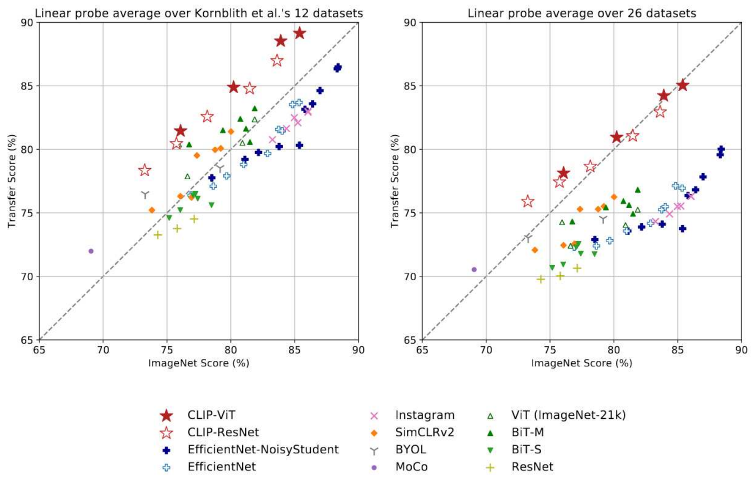


Figure 4.5: Clip robustness vs (former) sota models on datasets designed around the shift in distribution. (from [15])

captured by the robot, since the data distribution, i.e. the type of images present in the CLIP training dataset, differs from the image data captured by the camera used. CLIP was trained on images salvaged from the internet, those are mostly hand-captured photographs and the objects from the corresponding descriptions are usually at the center of the image composition. The rigid position of the camera on the robot imposes an awkward view of captured objects from an unusual angle (see 3.1.2), which is not likely to be well represented in the CLIP training dataset.

This concern is alleviated by the fact that [15] observed remarkable robustness to data distribution shifts when using CLIP for classification tasks (see 4.5). [23, 24] further cement these results and reveal that it is the careful selection of training examples, not the size of the dataset, that contributes the most towards the observed robustness. There are also different alternatives to CLIP [26, 27], and an open source implementation [25], which are worth considering in future work.

4.3.2 Sensing

Capturing and synchronizing sensor data is already implemented in ROS, see 3.2.1. The Robot localization node provides seamless access to pose data as if it were sensor data, see 3.2.1. S is in our case comprised of RGB images and depth maps from the front torso RGB-D camera images and pose of the camera with respect to the robot's map coordinate frame. Thus

$$S = \{0, 1 \dots 255\}^{480 \times 640 \times 3} \times \mathbb{R}^{480 \times 640} \times \mathbb{R}^3 \times \mathbb{H}$$

($\mathbb{H}$ denotes the set of all quaternions, they are commonly used in robotics and computer vision to represent orientation/rotation in 3D space)

4.3.3 Adapting existing code

The codebase of the classifier described in 3.3 was refactored into encapsulated modules such that the encapsulation of the code is in accord with the formal encapsulation from 4.2. This allows for modularity in its components, most importantly, the YOLACT classifier [28, 29], which since its release has been outperformed in both speed and accuracy simultaneously [30, 31, **?**]. Whether the current state-of-the-art approaches exhibit the same properties when it comes to the distribution of classification confidence for known and unknown classes as observed in [1] remains to be explored in future work, as YOLACT[28, 29] has proved to be sufficiently performant for the purposes of our work.

4.3.4 Observing

All logic from the aforementioned metaclassifier regarding its underlying model inference and the processing of its outputs was compartmentalized into two encapsulated modules. One for processing the image data with 2D models and one for using the spatial data from the camera's depth map and pose data. O is therefore made up of the union of all possible outputs of these two modules. (Some outputs of the 2D module may be omitted as they are used only by the second module) The output of these two modules is as follows:

2D

- $(m_i)_{i=1}^n : m \in \{0, 1\}^{480 \times 640}$ Segmentation pixelmaps of detected instances of objects
- $(l_i)_{i=1}^n : l \in L$ Class labels of detected instances of objects
- $(s_i)_{i=1}^n : s \in \mathbb{R}$ Confidence scores of detected classifications
- $(x_i)_{i=1}^n$ where

- $(e_i)_{i=1}^n : e \in \mathbb{R}^k$ Vector space embeddings of the image cropped to the bounding box of the detections. These are used to store the semantic information about the surrounding world

Where n is the number of detected instances of objects, k is the dimension of the embedding space of the embedding model used and L is the set of labels of classes known to the instance segmentation model.

3D

- $(p_i)_{i=1}^n : p \in \{X : |X| < \infty, X \subset \mathbb{R}^3\}$ back-projected point clouds of detected pixel masks, as described in 4.1.2
- $(c_i)_{i=1}^n : c \in \mathbb{R}^3$ Centroids of back-projected point clouds

Where n is the number of detected instances of objects

4.3.5 Scene reperesentation

Just like in the previous section, all logic from 3.3 regarding the aggregation of individual observations of objects in 3D was separated into an encapsulated module, the 3D positions of objects and their observed data are what was chosen as our scene representation R. While according to the formal description of the pipeline, the process of obtaining individual instances of objects and their descriptions (classification labels in this case) from the scene representation should be encapsulated in its own module, it has in this case proven to be more practical to include it in the same module as the used scene representation is by design composed of object instances.

4.3.6 Memory

Memory is a crucial component of our system, it acts as a repository for two distinct types of information: the semantics of detected objects and the descriptions (labels) provided by the oracle. This section elaborates on the structure and functionality of the memory component, detailing the database technologies employed and their integration into the overall system.

Vector databases

As the name implies, vector databases are designed around storing vectors, usually vector space embeddings of data, this allows for querying the database based on semantic similarity to the queried data. Vector databases have seen a surge in popularity with the rise of artificial intelligence and large language models, serving as a memory for storing long-term contexts, usually in the form of embeddings of documents or text. They are also being used in image-based multimodal applications, for example for relevant image retrieval and image search tasks. These are very similar to our application.

$$S_C(AB) = \frac{A \cdot B}{\|A\|_2 \|B\|_2} \cos\theta$$

Figure 4.6: The cosine similarity of vectors A and B. θ is the angle between A and B

ChromaDb

ChromaDb [33] was chosen as a back end for storing the semantics of detected objects, it is a lightweight open-source vector database. Chroma was chosen because it offers native support for multimodal embeddings and for its developer-friendly design philosophy. For the purposes of our application, it stores the image embeddings e and metadata containing the classification score and label assigned by the open-set meta-classifier(3.3) as well as a boolean flag to signal whether the classification is made by the oracle, or by the classifier, it is possible that the meta-classifier does not associate the observation with an object and does not classify it, this can happen if there aren't enough observations paired with it. In that case, the label is assigned a sentinel value of "undecided". These entries are deleted every time a new scene representation R is being constructed.

TinyDb

Complementing ChromaDb, TinyDb[34] is employed as a lightweight, JSON-based database to store descriptions provided by the oracle. Its minimalistic design makes it ideal for managing the less complex but equally crucial data related to Oracle inputs. TinyDb's primary role is to store the labels assigned to objects by the oracle, ensuring they are readily accessible for future reference

4.3.7 Description

In our case, the description of an object is simply its classification label, as the purpose of this work is to build upon an existing classifier, which uses just that. The label is also paired with the associated cropped images in order to provide sufficient information to the oracle. The implementation of the object description function g is split into two steps (both of these implemented in the scene representation module as explained in 4.3.5):

Classification from scene representation

This process is a direct application of the open-set classification method described in 3.3

Classification from memory

If an unknown object is detected, the memory is searched for similar objects that have either been classified as a known class or assigned a label by the oracle.

The search is done by querying the vector database for k(is a parameter) nearest neighbors (determined by the cosine similarity α of image embeddings). The vector, which is used to query the database is obtained by taking the mean of the observation embeddings e that were associated with the object.

This is justified by empirical results of methods [21, 22] which fused CLIP embeddings of images taken from different perspectives by taking the mean with good results, indicating that the semantic meaning of these embeddings is well preserved by taking their mean.

After that, a score is calculated for each of the neighbors, given by $\frac{max(\alpha-\lambda,0)}{\alpha^*-\lambda}$ where $\lambda \in \mathbb{R}$ is a parameter, α is the cosine similarity of the neighbor and α^* is the cosine similarity of the nearest neighbor.

Finally, a meta-score is computed, and a classification is made, using the confidence thresholding method (maxsum) from 3.3. The approach of classifying an unknown object by looking at the closest matches from a set of known objects is well-known in the domain of machine learning, it's referred to as a k-nearest-neighbors (knn) classifier.

There is a similar approach employing a knn classifier on top of CLIP embeddings explored in the literature [35] it maintained competitive performance within the domain of classifiers capable of incremental learning. While this approach makes its classification decision based on the majority vote of its neighbor's classifications, our approach extends upon it by accounting for the difference in similarity with different neighbors and extends the classifier itself from a closed set to an open-set by employing the methods of meta-classification as explored in [1].

4.3.8 Oracle

The 'Oracle' in our system is conceptualized as the primary source of ground truth, providing labels for objects belonging to unknown classes. We explored two avenues for this role: direct user input and we also conducted preliminary exploration into the use of a multimodal large language model (MMLLM) as an automated alternative. The process of applying the knowledge obtained from the oracle, i.e. implementing l is as simple as inserting the right label into the appropriate vector database entries metadata.

User as oracle

While some software for managing user interaction with ARI has been developed, [36] provides a back end for the process of approaching people and starting an interaction, on top of which various user interactions may be implemented, it is not well-documented, nor accessible.

We have for those reasons chosen to only provide an API for interfacing with the pipeline that could be simply implemented into an application for interacting with people, which we leave as a subject for future work. Altho the best interface for real-life applications is to ask users directly via face-to-face, speech-to-speech interactions with the ARI robot. It is not important for the purposes of this work, as it does not provide any functional benefit for the task of learning new objects.

Interaction API

The API provides functions for fetching the appropriate data, i.e. images and labels objects of unknown classes or uncertain classification, and for saving Oracle's labels back to memory. We have, for the purposes of our work implemented a simple interface for displaying appropriate images and querying for labels on top of it.

MMLLM as oracle?

In considering alternative approaches for the oracle component in the system, the potential use of a multimodal large language model (MMLLM) emerges as an intriguing substitute for direct user input. This theoretical consideration is driven by the extensive dataset exposure and sophisticated capabilities inherent in MMLLMs, which could potentially provide a broader and more nuanced knowledge base compared to an average user.

Can LLMs adapt to known label sets?

In future research, it's essential to explore whether Large Language Models (LLMs) can adapt to the specific linguistic domain of known labels within our system. This adaptability is crucial for providing contextually relevant labels and for aligning the LLM's outputs with the system's evolving knowledge base. Such an investigation would involve assessing the LLM's performance across various contexts and its ability to generate specific labels, ultimately determining if LLMs can serve as automated, context-aware oracles within dynamic environments. This area of research promises to significantly enhance the system's adaptability and responsiveness to new information.

The uncertainty of obtaining unknown labels

As [1] acknowledged the open set recognition literature [**?**, 48] recognizes classes in terms of associated labels. This in our case poses a challenge, since it

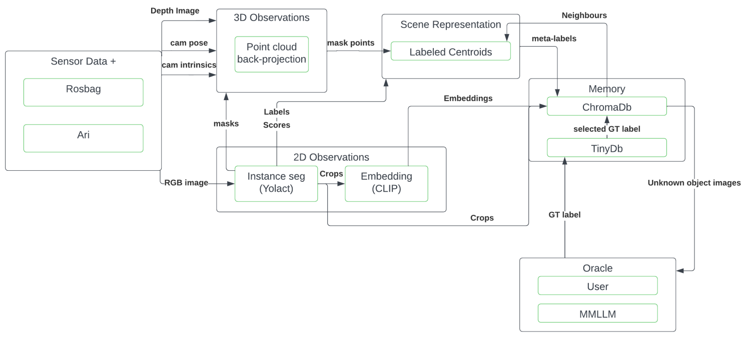


Figure 4.7: Visual summary of the pipeline's implementation and its data flow

implicitly posits that each detected object of the unknown class is associated with exactly one class label. However, in practice, it is not guaranteed that two different instances of an object of a certain class will be assigned the same label by the oracle, it is not even guaranteed that the oracle will provide the same label twice given two separate queries.

One way to approach this issue would be to expand the capabilities of our from-memory classifier to support multi-label classification, this would however introduce a wide range of nuanced issues, such as the effect of the number of labels on the decision of whether a meta-classification should be known or unknown, what even is an unknown class when multiple labels are considered, and dealing with label hierarchies, which would require complicated solutions, especially since labels are introduced incrementally.

For those reasons, we have opted for a simpler approach, which relies on keeping all labels assigned to objects by the oracle stored in memory and from these selecting the most relevant one based on the text-to-image embedding similarity. This approach introduces an issue regarding synonymous labels, that is, two instances of objects of the same class may be assigned different synonymous labels (i.e. cup and mug) and would thus be seen as objects of different classes. This would negatively affect the final classification results. This issue could be solved by clustering objects from memory based on image and/or label similarity, introducing another interaction with the oracle, which ensures that the clustered objects are in fact in the same class, and finally selecting one class label for each cluster. Since this is just a theoretical edge case, we leave addressing it as a subject for further work.

4.3.9 Possible scene representation alternatives

For future work, it may be interesting to consider different scene representations R, that provide more complete information about the surrounding world. We have conducted some preliminary exploration of this possibility with dense point cloud reconstructions with corresponding semantic language-aligned embeddings per each point. They will be referred to as embedded point clouds.

Other methods worth considering include other geometric data structures commonly used in computer vision, such as voxel grids or mesh grids, these would, much like point clouds allow for a more fine-grained spatial description of the detected objects and reasoning about its 3D geometry.

There are also relatively new radiance field-based methods for representing 3D scenes [37, 38], that are capable of capturing visual information in unprecedented detail. Their capability has recently been extended to capture semantic information as well [22, 39, 40]. Such methods could thus provide a powerful backbone for understanding the world, however, efficiently using the information stored in these world models, without converting them to more classical ones such as point clouds or mesh grids, is still an active area of research.

4.4 Visualization

To demonstrate, observe, and understand the workings of a system, it is beneficial to have a way of viewing the exchanged and accumulated data in a human-interpretable format. For this purpose, a real-time visualization utility, which provides an intuitive and interactive way to explore the observed data O, the scene representation R, and memory M, was developed using the RViz utility.

4.4.1 Rviz

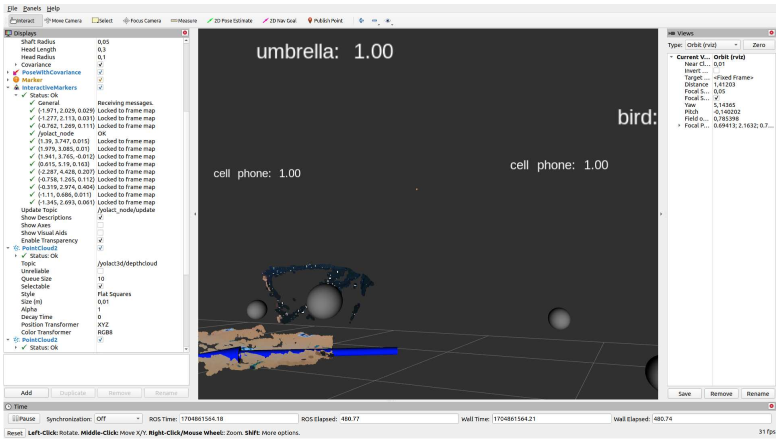
Rviz, short for ROS visualization, is a utility distributed along most ROS distributions. It's a powerful tool used to visualize the data from most of the standard ROS messages. It also provides a library for building basic interactive robotics applications and visualization software, where the workflow can be controlled by interacting with the displayed visualization elements.

4.4.2 Visualization server

The ROS visualization library allows for displaying various types of geometries and an interface to interact with them, either by clicking or dragging them with the mouse cursor, or via a pop-up context menu.

To provide interactive visualization capabilities to our application, we implemented a visualization server, which communicates with a running RViz instance over the ROS infrastructure. This server runs as a component of the classifier ROS node and is provided access to its relevant internal data. This server manages the visualization and interaction with data from R and M via an interface described in the next section.

A separate non-interactive visualization server was implemented to broadcast in visualizable ROS messages the observation data from O, as well as relevant

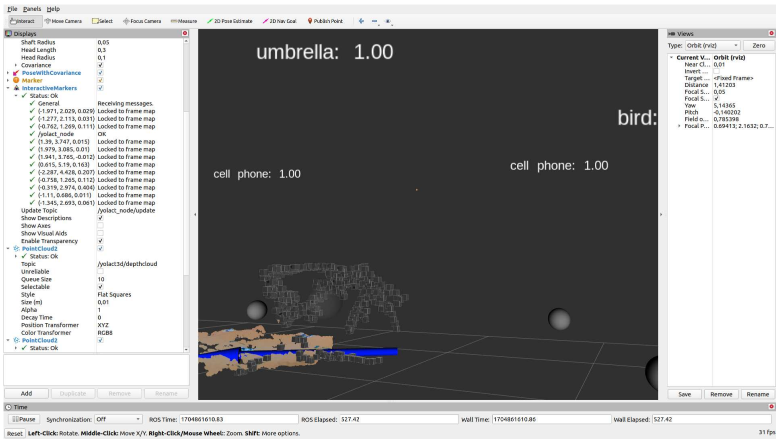


```
Shaft Radius                 0,05
Head Length                  0,3
Head Radius                  0,1
Covariance                   ✓
PoseWithCovariance           ✓
Marker                       ✓
InteractiveMarkers           ✓
  ✓ Status: Ok
    ✓ General                      Receiving messages.
    ✓ (-1.971, 2.029, 0.029)       Locked to frame map
    ✓ (-1.277, 2.113, 0.031)       Locked to frame map
    ✓ (-0.762, 1.269, 0.111)       Locked to frame map
    ✓ /yolact_node                 OK
    ✓ (1.39, 3.747, 0.015)         Locked to frame map
    ✓ (1.979, 3.085, 0.01)         Locked to frame map
    ✓ (1.941, 3.765, -0.012)       Locked to frame map
    ✓ (0.615, 5.19, 0.163)         Locked to frame map
    ✓ (-2.287, 4.428, 0.207)       Locked to frame map
    ✓ (-0.758, 1.265, 0.112)       Locked to frame map
    ✓ (-0.319, 2.974, 0.404)       Locked to frame map
    ✓ (-1.11, 0.686, 0.011)        Locked to frame map
    ✓ (-1.345, 2.693, 0.061)       Locked to frame map
  Update Topic               /yolact_node/update
  Show Descriptions          ✓
  Show Axes
  Show Visual Aids
  Enable Transparency        ✓
PointCloud2                  ✓
  ✓ Status: Ok
  Topic                      /yolact3d/depthcloud
  Unreliable
  Queue Size                 10
  Selectable                 ✓
  Style                      Flat Squares
  Size (m)                   0,01
  Alpha                      1
  Decay Time                 0
  Position Transformer       XYZ
  Color Transformer          RGB8
PointCloud2                  ✓
  ✓ Status: Ok

Type: Orbit (rviz)   Zero
Current V...   Orbit (rviz)
  Near Cl...   0,01
  Invert ...
  Target ...   <Fixed Frame>
  Distance     1,41203
  Focal S...   0,05
  Focal S...   ✓
  Yaw          5,14365
  Pitch        -0,140202
  Field o...   0,785398
  Focal P...   0.69413; 2.1632; 0.7...

Synchronization: Off   ROS Time: 1704861564.18   ROS Elapsed: 480.77   Wall Time: 1704861564.21   Wall Elapsed: 480.74
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.   31 fps
```

Figure 4.8: Visualization of detected objects and depth cloud images processed by the detector



```
Shaft Radius                 0,05
Head Length                  0,3
Head Radius                  0,1
Covariance                   ✓
PoseWithCovariance           ✓
Marker                       ✓
InteractiveMarkers           ✓
  ✓ Status: Ok
    ✓ General                      Receiving messages.
    ✓ (-1.971, 2.029, 0.029)       Locked to frame map
    ✓ (-1.277, 2.113, 0.031)       Locked to frame map
    ✓ (-0.762, 1.269, 0.111)       Locked to frame map
    ✓ /yolact_node                 OK
    ✓ (1.39, 3.747, 0.015)         Locked to frame map
    ✓ (1.979, 3.085, 0.01)         Locked to frame map
    ✓ (1.941, 3.765, -0.012)       Locked to frame map
    ✓ (0.615, 5.19, 0.163)         Locked to frame map
    ✓ (-2.287, 4.428, 0.207)       Locked to frame map
    ✓ (-0.758, 1.265, 0.112)       Locked to frame map
    ✓ (-0.319, 2.974, 0.404)       Locked to frame map
    ✓ (-1.11, 0.686, 0.011)        Locked to frame map
    ✓ (-1.345, 2.693, 0.061)       Locked to frame map
  Update Topic               /yolact_node/update
  Show Descriptions          ✓
  Show Axes
  Show Visual Aids
  Enable Transparency        ✓
PointCloud2                  ✓
  ✓ Status: Ok
  Topic                      /yolact3d/depthcloud
  Unreliable
  Queue Size                 10
  Selectable                 ✓
  Style                      Flat Squares
  Size (m)                   0,01
  Alpha                      1
  Decay Time                 0
  Position Transformer       XYZ
  Color Transformer          RGB8
PointCloud2                  ✓
  ✓ Status: Ok

Type: Orbit (rviz)   Zero
Current V...   Orbit (rviz)
  Near Cl...   0,01
  Invert ...
  Target ...   <Fixed Frame>
  Distance     1,41203
  Focal S...   0,05
  Focal S...   ✓
  Yaw          5,14365
  Pitch        -0,140202
  Field o...   0,785398
  Focal P...   0.69413; 2.1632; 0.7...

Synchronization: Off   ROS Time: 1704861610.83   ROS Elapsed: 527.42   Wall Time: 1704861610.86   Wall Elapsed: 527.42
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.   31 fps
```

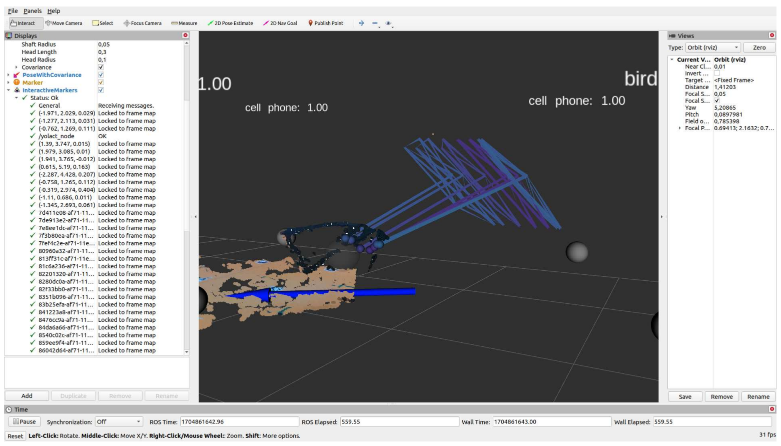
Figure 4.9: Visualization of points associated with an object cluster

sensor data from S, namely the RGB-D depth image, detection point clouds, and detection labels with scores.

Note that the aforementioned broadcast of RGB-D via the implemented server is only needed when using a rosbag recording, as only compressed camera data, which is incompatible with RViz is available there. The serialization of the RGB-D data into a point cloud message, which was our approach to work around the issue presents a major performance bottleneck due to inefficiencies in rospy, a ROS programming library for Python.

4.4.3 Interface

Each object detection is represented geometrically by a sphere (4.8). Clicking on this sphere brings up a context menu that allows for the display of camera positions from which it was observed4.10, plotting relevant data and visualizing the back-projected mask points (4.9).



```
Displays
    Shaft Radius          0,05
    Head Length           0,3
    Head Radius           0,1
  ▸ Covariance            ✓
▸ PoseWithCovariance      ✓
▸ Marker                  ✓
▾ InteractiveMarkers      ✓
  ▾ ✓ Status: Ok
      ✓ General               Receiving messages.
      ✓ (-1.971, 2.029, 0.029) Locked to frame map
      ✓ (-1.277, 2.113, 0.031) Locked to frame map
      ✓ (-0.762, 1.269, 0.111) Locked to frame map
      ✓ /yolact_node          OK
      ✓ (1.39, 3.747, 0.015)  Locked to frame map
      ✓ (1.979, 3.085, 0.01)  Locked to frame map
      ✓ (1.941, 3.765, -0.012) Locked to frame map
      ✓ (0.615, 5.19, 0.163)  Locked to frame map
      ✓ (-2.287, 4.428, 0.207) Locked to frame map
      ✓ (-0.758, 1.265, 0.112) Locked to frame map
      ✓ (-0.319, 2.974, 0.404) Locked to frame map
      ✓ (-1.11, 0.686, 0.011) Locked to frame map
      ✓ (-1.345, 2.693, 0.061) Locked to frame map
      ✓ 7d411e08-af71-11...   Locked to frame map
      ✓ 7de913e2-af71-11...   Locked to frame map
      ✓ 7e8ee1dc-af71-11...   Locked to frame map
      ✓ 7f3b80ea-af71-11...   Locked to frame map
      ✓ 7fef4c2e-af71-11e...  Locked to frame map
      ✓ 80960a32-af71-11...   Locked to frame map
      ✓ 813ff31c-af71-11e...  Locked to frame map
      ✓ 81c6a236-af71-11...   Locked to frame map
      ✓ 82201320-af71-11...   Locked to frame map
      ✓ 8280dc0a-af71-11...   Locked to frame map
      ✓ 82f33bb0-af71-11...   Locked to frame map
      ✓ 8351b096-af71-11...   Locked to frame map
      ✓ 83b25efa-af71-11...   Locked to frame map
      ✓ 841223a8-af71-11...   Locked to frame map
      ✓ 8476cc9a-af71-11...   Locked to frame map
      ✓ 84da6a66-af71-11...   Locked to frame map
      ✓ 8540c02c-af71-11...   Locked to frame map
      ✓ 859ee9f4-af71-11...   Locked to frame map
      ✓ 86042d64-af71-11...   Locked to frame map

Views
Type: Orbit (rviz)   Zero
▾ Current V...   Orbit (rviz)
    Near Cl...   0,01
    Invert ...   ☐
    Target ...   <Fixed Frame>
    Distance     1,41203
    Focal S...   0,05
    Focal S...   ✓
    Yaw          5,20865
    Pitch        0,0897981
    Field o...   0,785398
  ▸ Focal P...   0.69413; 2.1632; 0.7...

Time
Synchronization: Off   ROS Time: 1704861642.96   ROS Elapsed: 559.55   Wall Time: 1704861643.00   Wall Elapsed: 559.55
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.   31 fps
```

Figure 4.10: Visualization of detections, their respective camera poses, and centroids. Clicking on the individual detections opens a context menu that allows for the plotting of data relevant to the detection.

Chapter 5

Experiments

Carrying out a full-scale qualitative evaluation in the real world requires a lot of resources, so we'll have to make do with a few proof-of-concept experiments.

5.1 Initial classifier experiments

This section describes the initial evaluation of 3.3 on ARI that was done in collaboration with Rakshith Madhavan (see acknowledgments). This section is a paraphrase of the Deliverable D2.6 report for the SPRING research project

5.1.1 Procedure

The experiment went like so:

- Objects of varying sizes were placed around the floor in a small radius around a center point. See fig 5.3
- ARI navigated autonomously in a trajectory of concentric circles, looking toward the center, of multiple radii around the objects. See fig 5.1

The circular trajectory ensures that ARI captures the objects from all orientations, and multiple radii make ARI view the objects from varying distances. ARI moves to points along the circle, facing the center of the circle (where the objects are) at specified intervals.

Recorded topics

The following topics were recorded in rosbags:

- Robot pose in the map: /robot_pose
- Tf: /tf and /tf_static
- Torso front camera image: /torso_front_camera/aligned_depth_to_color/image_raw
- Torso front camera parameters: /torso_front_camera/color/camera_info

5.1.2 Results

Overall, three sets of data were recorded (5.5), each observed in an interactive data visualization interface (example in figure 5.3) and evaluated, due to the small scale of the test, manually.

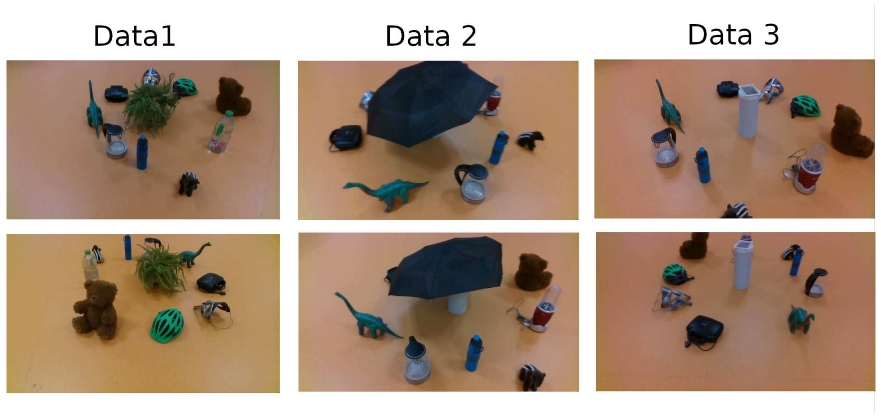


Figure 5.2: The three observed data recordings

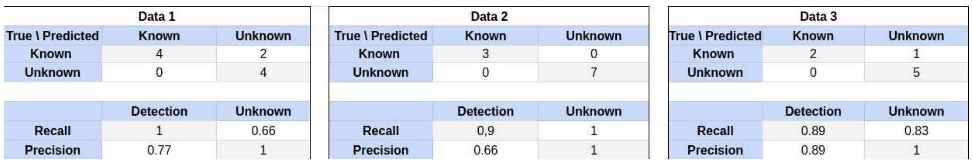
Used metrics for measuring the performance of the model consisted of recall ((**??**)) and precision ((**??**)) at both differentiating unknown from known and detecting that an object is present. As every known class was detected and classified correctly, no metric was required for known class classification.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{5.1}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{5.2}$$

Results are presented in figure 5.4.

Probable cause of failure cases in object detection is the model's (3.3) representation of objects as spheres of constant radius, which as a consequence may result in one object having multiple clusters representing it (this is treated as false positive detection), or, having YOLACT detections of another object clustered to the cluster that represents it. The first case was especially notable in Data 2 as it contained an open umbrella, which spans a relatively large area, the second case was rare enough to not cause significant interference in most classification decisions, however, there are two cases, where an object was not detected as its YOLACT detections were clustered to clusters representing other objects. Besides the addressable errors, the results have shown that the detector is somewhat applicable for real-world experimental applications.



Data 1		
True \ Predicted	Known	Unknown
Known	4	2
Unknown	0	4
	Detection	Unknown
Recall	1	0.66
Precision	0.77	1

Data 2		
True \ Predicted	Known	Unknown
Known	3	0
Unknown	0	7
	Detection	Unknown
Recall	0,9	1
Precision	0.66	1

Data 3		
True \ Predicted	Known	Unknown
Known	2	1
Unknown	0	5
	Detection	Unknown
Recall	0.89	0.83
Precision	0.89	1

Figure 5.4: Results of the experiment. Above is the confusion matrix for the Known vs Unknown classification. Below are the precision and recall scores for object detection (not classification) and recognition of Unknown over Known (Unknown is regarded as a positive).

5.2 Method application experiments

5.2.1 Data collection

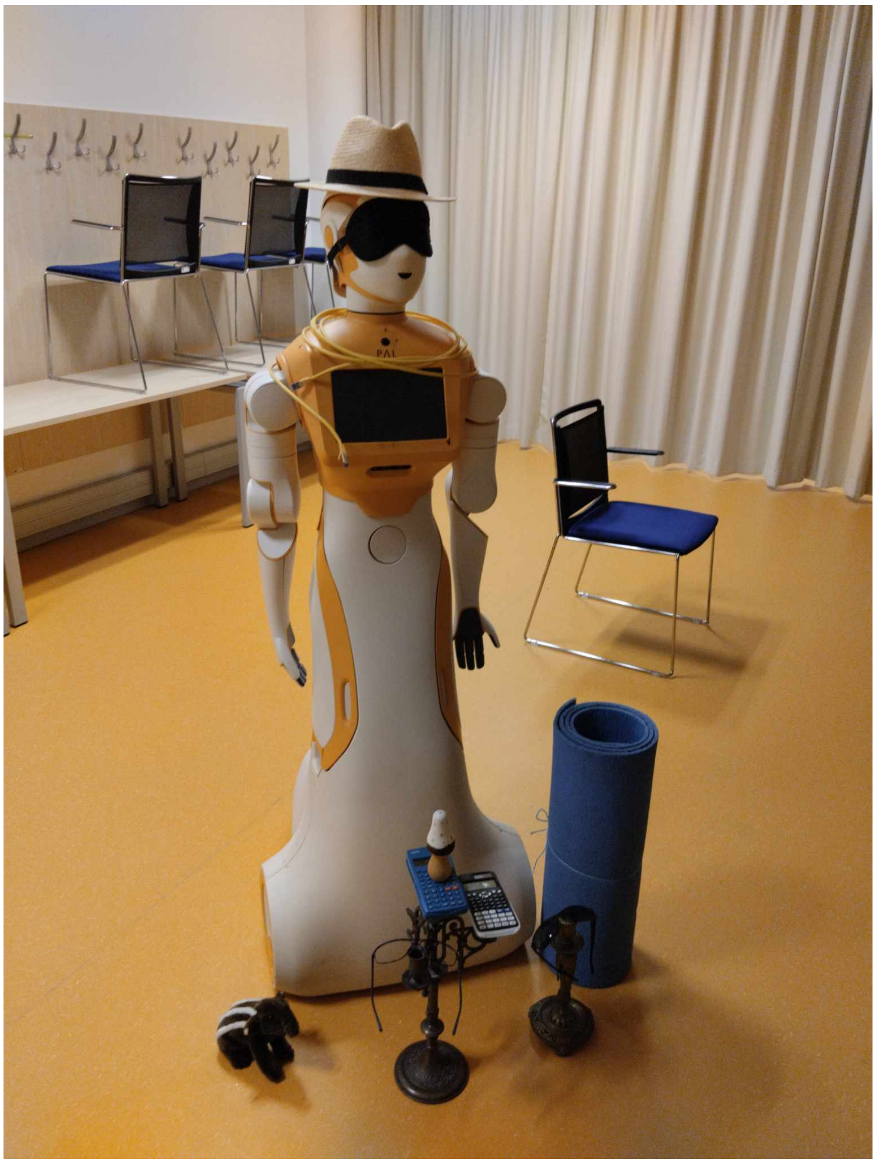


Figure 5.5: The testing environment

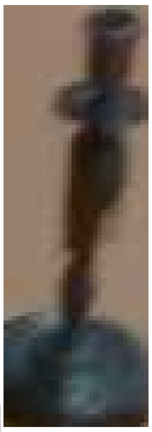
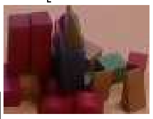
The approach to collecting data was similar to that of the initial experiment. Objects were placed on the ground and ARI navigated around them while observing them, however, having learned from previous mistakes, the objects were arranged with more spatial separation in a straight line instead of a circle, this however, as will be later pointed out again, introduced a drawback, as due to ARI's limited maneuverability fewer views of tested objects were captured. Due to spatial constraints, the data was recorded 3-4 objects at a time

Used objects

A data on 29 objects, 25 of which unknown, of 14 classes, 10 of which unknown, was recorded.

- 3 different candelabrums.
- 5 different structures constructed from colored wooden blocks.
- 5 pairs of different glasses
- 2 pairs of different chess boards, including pieces
- 2 different calculators
- 2 different figurines of a tapir
- 3 different hats
- 4 more unknown objects each of a different class and 4 known objects of different classes. These were included to increase the number of classes and increase the difficulty of the learning task.

[Candelabrum1]

[Candelabrum3]

[blocks]

[glasses2]

[glasses3]

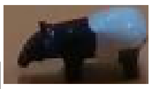
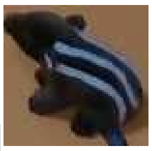
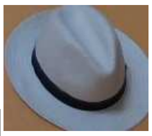
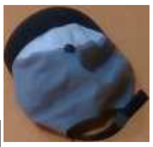
[tapir1]

[tapir2]

[hat1]

[hat2]


Figure 5.6: Some of the used objects

5.2.2 Method

All relevant data from the individual recordings resulting in scene representations was accumulated into one merged representation (disregarding spatial information) and evaluated together.

5.2.3 Yolact vs zero-shot CLIP

A possibility for relying on CLIP's representational power also for the initial open-set classification decision was also explored. For this purpose, the softmax distribution over the cosine similarity to labels known to yolact was compared with yolact's softmax distribution, over multiple views of aforementioned objects, to see if similar regulariaties emerge.

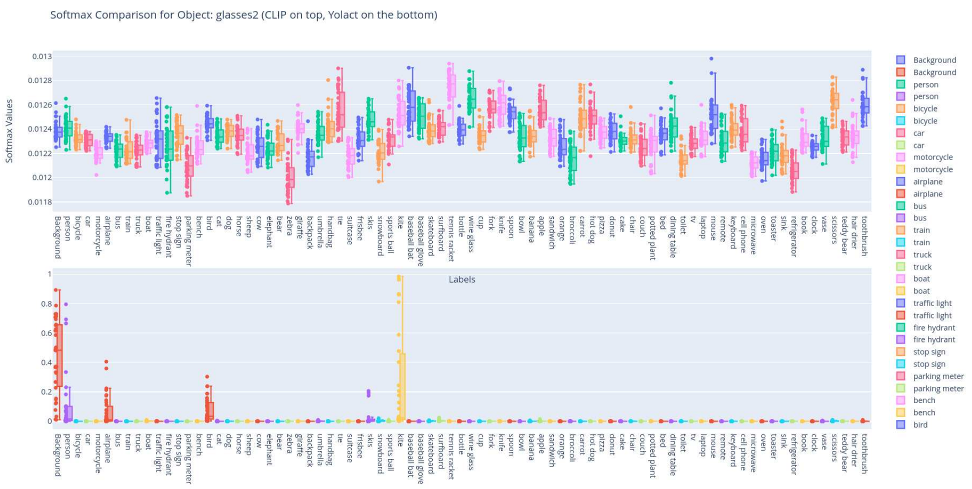


Figure 5.7: Chaotic distribution on objects of unknown class seems like a desirable behaviour, however, the distribution is equally as chaotic on objects of known classes

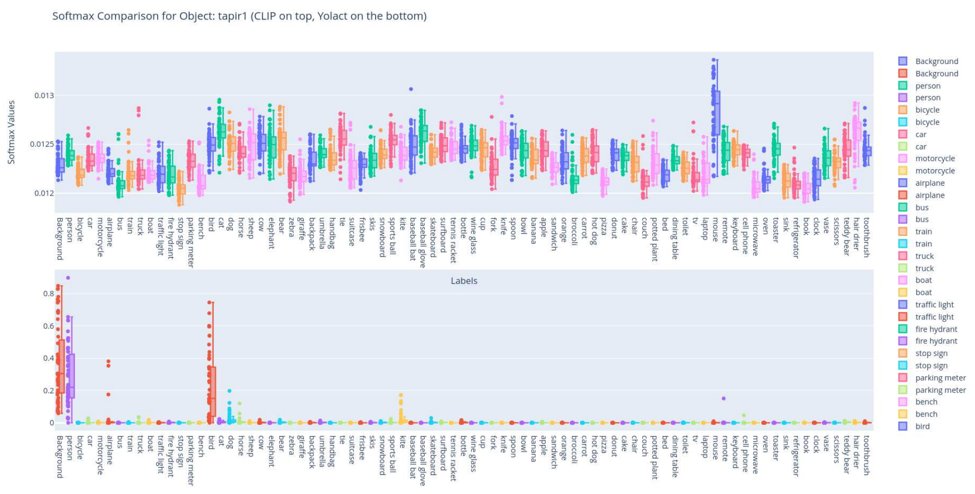


Figure 5.8: Spike on mouse label, likely due to ambiguity of the label resulting in correlation with the animal.

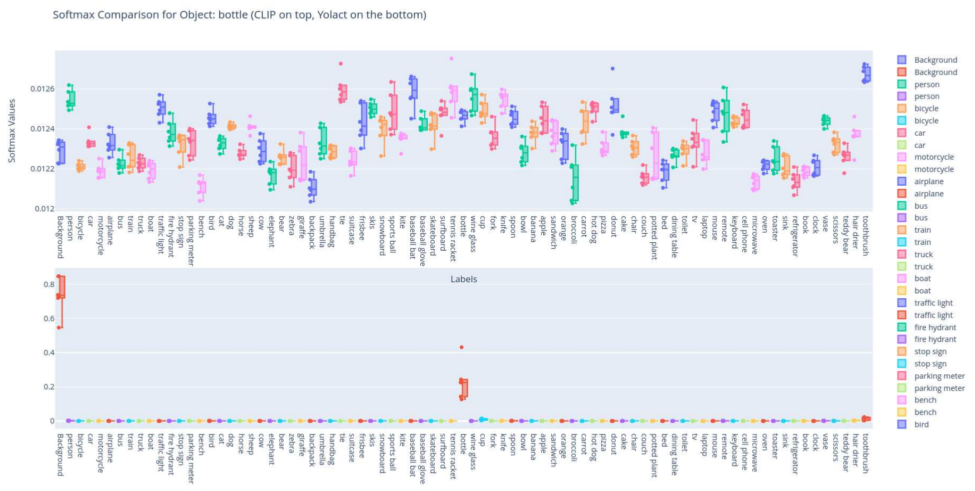
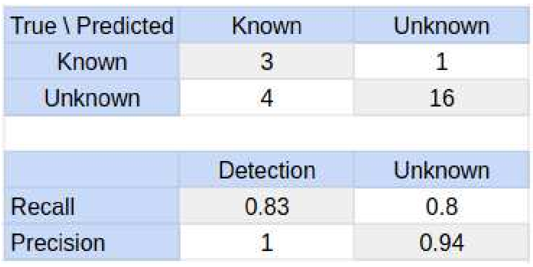


Figure 5.9: No increase around the known class observed over the usual fluctuations.

As can be seen, CLIP's softmax distribution is much more entropic than Yolact's, even on known classes (5.9), and does not appear usable. Studying these distributions in greater detail, or evaluating whether it is due to the relatively low resolution of detected cut-outs that's causing a misalignment between language and image latent space is left for future work.

5.2.4 Detection of unknown objects

We have conducted the same evaluation of the detector's performance as during the initial experiments, this time free of errors caused by the environment being incompatible with the detector's scene representation. However, overall 5 objects were not detected since there weren't as many viewing perspectives of the observed objects as before and Yolact simply didn't make enough detections for the meta-classifier to make a valid classification decision.



True \ Predicted	Known	Unknown
Known	3	1
Unknown	4	16

	Detection	Unknown
Recall	0.83	0.8
Precision	1	0.94

Figure 5.10: Confusion matrix of recognizing unknown from known. One known object was misclassified as unknown, but the remaining remains correctly classified. Thus same evaluation metrics as in the previous experiments can be employed.

5.2.5 Learning of unknown objects

The data from the unclassified/undetected object clusters (candelabrum2 and 3, charger, glasses5, and blocks2) was still used as if it were a detected object for the purposes of this section.

Inspecting the mean of embeddings

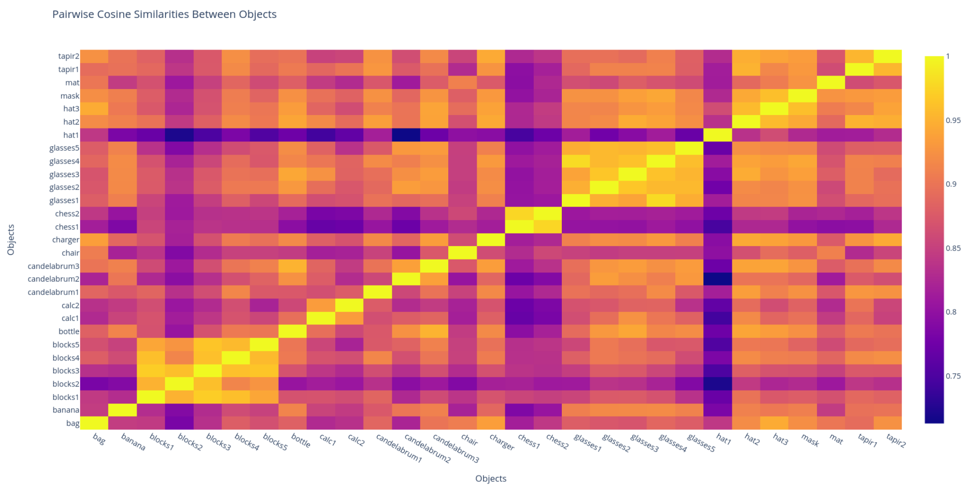


Figure 5.11: Heat map of pairwise cosine similarities between the mean embeddings of each object cluster.

The pairwise cosine similarity between the mean embeddings of observed objects embeddings As can be seen in 5.14 objects of the same class bear the closest similarity. One exception is the candelabrums class since two of the three objects in that class were among the non-detected objects, whose clusters did not accumulate enough appropriate data. Also, many detections of the detected candelabrum were not of the whole object, but of parts, which when cropped, bear very little similarity to the original object, thus polluting its mean embedding. The surprising result however, is the dissimilarity of hat1 to the other hats, since all hats had enough quality detections, and while it is a different type of hat made of different material, one would still expect that to have minimal impact on the general semantics of the concept of a hat. An intuitive observation however is that hats are similar to glasses and that glasses are similar to a blindfold(mask) since all of them appear in similar contexts (are worn on a head).

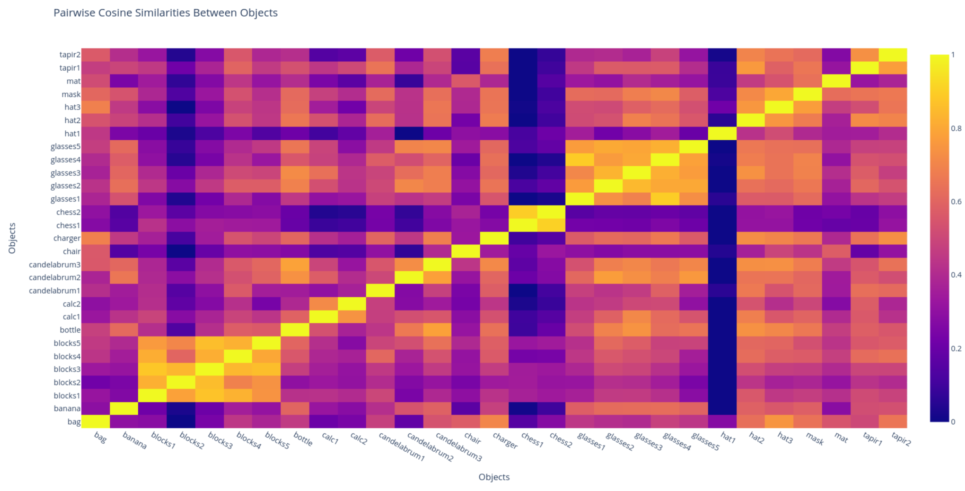


Figure 5.12: The same heat map as 5.14, except with normalized columns

Another interesting observation is that while the cosine similarity is usually highest amongst the groups of objects of the same class, the actual value of the cosine similarity varies significantly between different classes, i.e. tapir1 is of all the objects closest to tapir2, however, a banana is just as close to a candelabrum, even when it itself would be much closer to another banana. This observation is what motivated our proposed neighbor scoring function, which scales the cosine similarity by the closest neighbor.

Evaluating classification from learned memory

In this experiment, every object but the evaluated one was annotated with ground truth labels and saved to memory. The evaluated object was then assumed to be detected as unknown in the first stage of its classification and classified by the second memory-based stage, if it was among the 8 objects that were the only ones in their class, then the correct classification is unknown. If there are other representatives of the evaluated object's class, then that class is the correct classification.

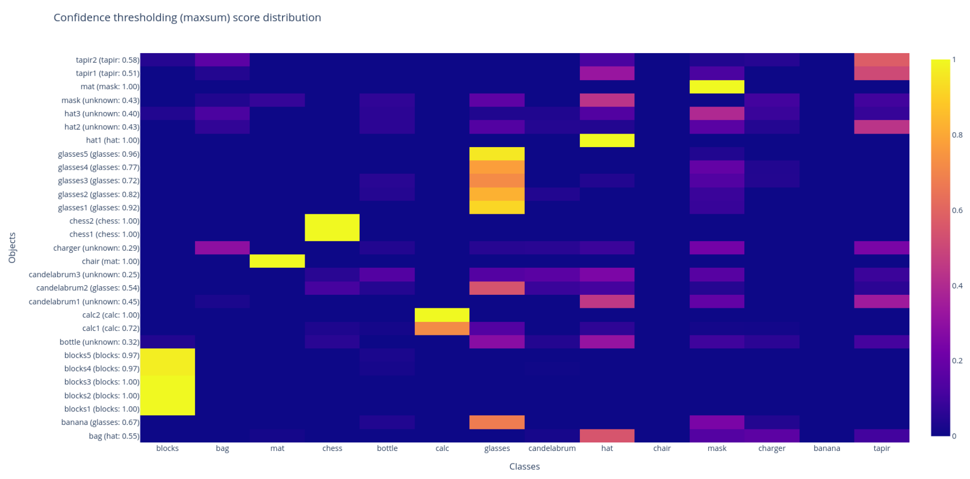


Figure 5.13: The initial results of the experiment. Candelabrum and Hat failed to learn properly, which was foreshadowed in the previous section. Overall 9 classification errors were made, resulting in a classification accuracy of 0.68

The initial results (**??**) showed over-confident detections (mat and chair), upon further inspection it was revealed that this was, in fact, the result of all neighbors but one being too far from those objects, the parameter λ implicitly defines a threshold of a distance a neighbor needs to be within to be considered relevant if he crosses that distance, his score goes to zero and he is no longer considered in the classification process. This result necessitates the introduction of another parameter, which controls the minimal number of valid neighbors for a classification to be made, otherwise, the classification will remain as "unknown".

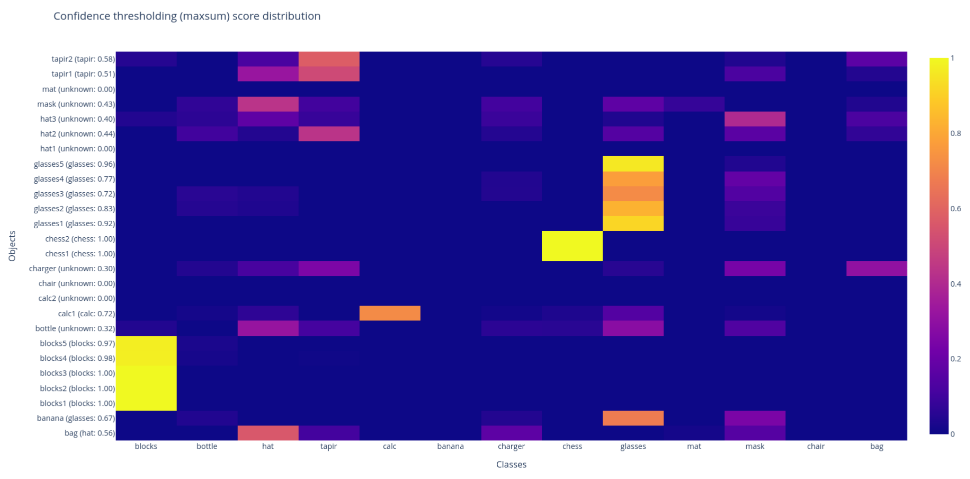
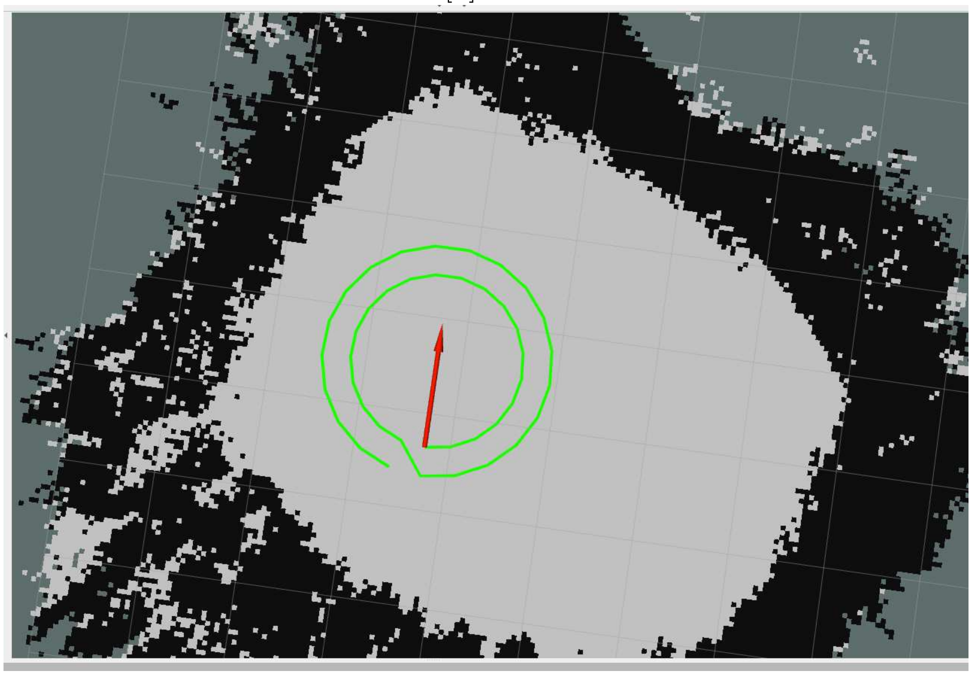


Figure 5.14: The results after applying a threshold for minimal neighbors. Also, candelabrum was removed, since it was clearly not well represented. Notice the change in the classification of calc2, this is due to calc2 having few high-resolution detections. Most of calc2's detections were blurred and captured from a distance. Overall 6 classification errors were made, resulting in a classification accuracy of 0.77

That being said, systematically searching for the optimal set of parameters serves little purpose given the size of the experimental dataset, as it would only overfit the scarce amount of data and is therefore left as a subject for future work. One approach worth considering in this endeavor would be the use of dynamically determined parameters based on the current size of memory and the uncertainty of an open-set classification as some classes will be inevitably more represented in memory than others.

[b]0.35



[b]0.36

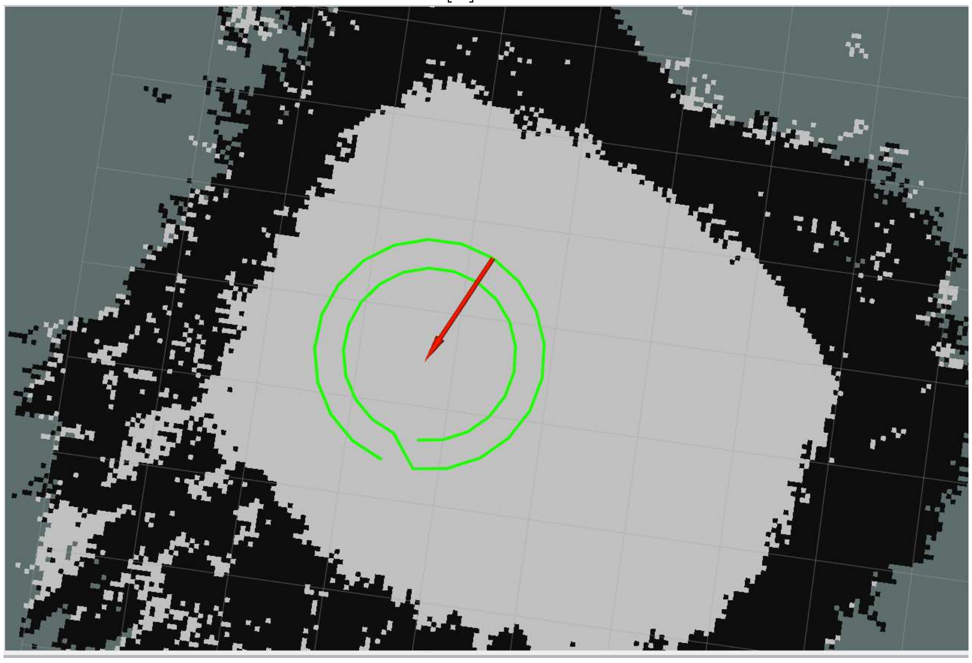


Figure 5.1: Circular trajectory around objects with goal poses along the circle looking at the objects. This ensures that ARI captures the objects at different angles, and viewpoints

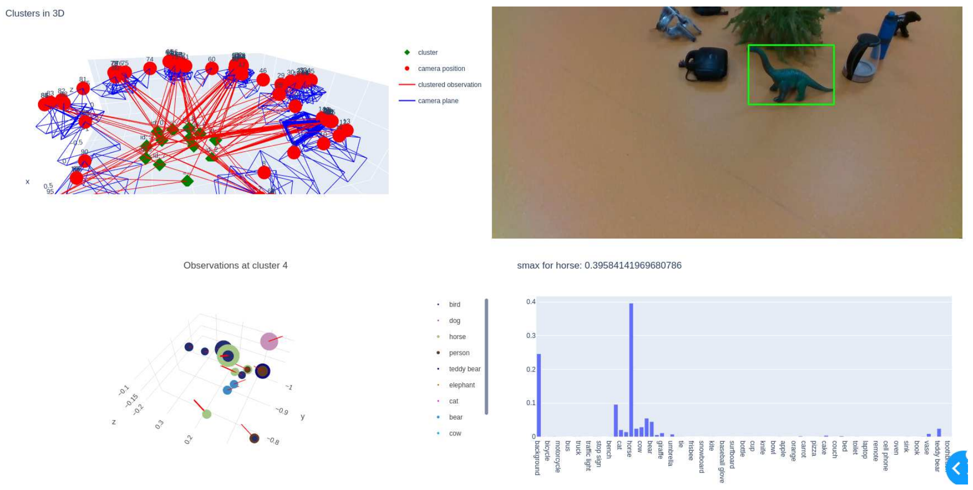


Figure 5.3: The interface used for evaluation. The top left contains an interactive 3D visualization of the camera and cluster positions. Below is a visualization of detections in a specific cluster from Figure 1. The top right contains an image with a specific detection bounding box. Below is the YOLACT softmax score for that detection.

Chapter 6

Closing chapter

6.1 Conclusion

This thesis represents an initial foray into the integration of machine learning, computer vision, and natural language processing within the ARI humanoid robot, particularly focusing on the challenge of learning and recognizing unknown objects. Through the development of a 3D object detection and classification pipeline, the project sought to enhance ARI's ability to interact with its environment by identifying objects not previously encountered.

Key to this endeavor was the application of open-set recognition techniques, allowing the robot to categorize objects outside its pre-defined dataset. This approach showed potential in expanding ARI's understanding of novel objects, though the results also highlighted the complexities inherent in real-world applications. Incremental learning methodologies were explored to enable ARI to continually update its knowledge base, an essential step towards adaptive learning in robotic systems.

Experiments conducted provided preliminary insights into the system's performance, revealing both successes in controlled settings and areas for improvement toward more dynamic environments.

Looking ahead, there are several avenues for further exploration and development. Improving the accuracy and robustness of the object detection system under varying conditions remains a key challenge. Integrating more nuanced NLP models could potentially enhance ARI's interactive and communicative abilities. In summary, this thesis is a step towards understanding how robots like ARI can better learn and adapt in complex environments. While the work conducted is preliminary, especially given the constraints of an undergraduate project, it contributes to the broader dialogue on the future of interactive and adaptive robotics, setting the stage for further research in this exciting field.

Bibliography

[1] Nikita Sokovnin *Recognizing Unknown Objects for Open-Set 3D Object Detection*[Bachelor thesis] CVUT http://hdl.handle.net/10467/94428, 2021

[2] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning.* MIT Press, 2018.

[3] Bonner, M.F., Epstein, R.A. *Object representations in the human brain reflect the co-occurrence statistics of vision and language.* Nat Commun 12, 2021. https://doi.org/10.1038/s41467-021-24368-2

[4] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean *Efficient Estimation of Word Representations in Vector Space* arXiv preprint arXiv:1301.3781, 2013.

[5] J. Pennington, R. Socher, and C. D. Manning *GloVe: Global Vectors for Word Representation* Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean *Distributed Representations of Words and Phrases and their Compositionality* arXiv preprint arXiv:1310.4546, 2013.

[7] Levy, Omer, et al. *Improving Distributional Similarity with Lessons Learned from Word Embeddings* Transactions of the Association for Computational Linguistics 3, 2015

[8] Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* North American Chapter of the Association for Computational Linguistics, 2019

[9] Jinghua Zhang, Li Liu, Olli Silvén, Matti Pietikäinen, Dewen Hu *Few-shot Class-incremental Learning: A Survey* arXiv preprint arXiv:2308.06764, 2023

[10] Ilya Sutskever, Oriol Vinyals, Quoc V. Le *Sequence to Sequence Learning with Neural Networks* arXiv preprint arXiv:1409.3215, 2014

[11] Karen Simonyan, Andrew Zisserman *Very Deep Convolutional Networks for Large-Scale Image Recognition* arXiv preprint arXiv:1409.1556, 2014

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition* arXiv preprint arXiv:1512.03385, 2015

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* arXiv preprint arXiv:2010.11929, 2021

[14] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, Jeff Clune *Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos* arXiv preprint arXiv:2206.11795, 2022

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever *Learning Transferable Visual Models From Natural Language Supervision* arXiv preprint arXiv:2103.00020, 2021

[16] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak ... *Multitask Prompted Training Enables Zero-Shot Task Generalization* arXiv preprint arXiv:2110.08207, 2021

[17] Ashish Vaswani,Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin *Attention is All you Need* Neural Information Processing Systems, 2017

[18] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu and Mark Chen *Hierarchical Text-Conditional Image Generation with CLIP Latents* arXiv preprint arXiv:2204.06125, 2022

[19] Haotian Liu, Chunyuan Li, Qingyang Wu, Yong Jae Lee *Visual Instruction Tuning* arXiv preprint arXiv:2304.08485, 2023

[20] Golnaz Ghiasi, Xiuye Gu, Yin Cui, Tsung-Yi Lin *Scaling Open-Vocabulary Image Segmentation with Image-Level Labels* arXiv preprint arXiv:2112.12143, 2022

[21] Songyou Peng, Kyle Genova, Chiyu "Max" Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser *OpenScene: 3D Scene Understanding with Open Vocabularies* arXiv preprint arXiv:2211.15654, 2023

[22] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, Matthew Tancik *LERF: Language Embedded Radiance Fields* arXiv preprint arXiv:2303.09553, 2023

[23] Thao Nguyen, Gabriel Ilharco, Mitchell Wortsman, Sewoong Oh, Ludwig Schmidt *Quality Not Quantity: On the Interaction between Dataset Design and Robustness of CLIP* arXiv preprint arXiv:2208.05516, 2022

[24] Alex Fang, Gabriel Ilharco, Mitchell Wortsman, Yu Wan, Vaishaal Shankar, Achal Dave, Ludwig Schmidt *Data Determines Distributional Robustness in Contrastive Language Image Pre-training (CLIP)* International Conference on Machine Learning, 2022

[25] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, Jenia Jitsev *Reproducible scaling laws for contrastive language-image learning* arXiv preprint arXiv:2212.07143, 2022

[26] Junnan Li, Dongxu Li, Silvio Savarese, Steven Hoi *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models* arXiv preprint arXiv:2301.12597, 2023

[27] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Kenji Kawaguchi, Hanxiao Liu, Adams Wei Yu, Jiahui Yu, Yi-Ting Chen, Minh-Thang Luong, Yonghui Wu, Mingxing Tan, Quoc V. Le *Combined Scaling for Zero-shot Transfer Learning* arXiv preprint arXiv:2111.10050,

[28] Daniel Bolya, Chong Zhou, Fanyi Xiao, Yong Jae Lee *YOLACT: Real-time Instance Segmentation* arXiv preprint arXiv:1904.02689, 2019

[29] Daniel Bolya, Chong Zhou, Fanyi Xiao, Yong Jae Lee *YOLACT++: Better Real-time Instance Segmentation* arXiv preprint arXiv:1912.06218, 2020

[30] Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, Kai Chen *RTMDet: An Empirical Study of Designing Real-Time Object Detectors* arXiv preprint arXiv:2212.07784, 2022

[31] Kemal Oksuz, Baris Can Cam, Fehmi Kahraman, Zeynep Sonat Baltaci, Sinan Kalkan, Emre Akbas *Mask-aware IoU for Anchor Assignment in Real-time Instance Segmentation* arXiv preprint arXiv:2110.09734, 2021

[32] Tianheng Cheng, Xinggang Wang, Shaoyu Chen, Wenqiang Zhang, Qian Zhang, Chang Huang, Zhaoxiang Zhang, Wenyu Liu *Sparse Instance Activation for Real-Time Instance Segmentation* arXiv preprint arXiv:2203.12827, 2022

[33] Chroma *Chroma*[software library] github https://github.com/chroma-core/chroma, 2023

[34] Markus Siemens *TinyDb*[software library] github https://github.com/msiemens/tinydb, 2023

[35] Kengo Nakata, Youyang Ng, Daisuke Miyashita, Asuka Maki, Yu-Chieh Lin, Jun Deguchi *Revisiting a kNN-based Image Classification System with High-capacity Storage* arXiv preprint arXiv:2204.01186, 2022

[36] Nancie Gunson, Garcia, Daniel Hernandez, Weronika Sieinska, Christian Dondrup, Oliver Lemon *Developing a Social Conversational Robot for the Hospital waiting room* International Conference on Robot and Human Interactive Communication, 2022

[37] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis* arXiv preprint arXiv:2003.08934, 2020

[38] Kerbl, Bernhard and Kopanas, Georgios and Leimkühler, Thomas and Drettakis, George *3D Gaussian Splatting for Real-Time Radiance Field Rendering* ACM Transactions on Graphics, 2023

[39] Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, Shao-Hua Guan *Language Embedded 3D Gaussians for Open-Vocabulary Scene Understanding* arXiv preprint arXiv:2311.18482, 2023

[40] Xingxing Zuo, Pouya Samangouei, Yunwen Zhou, Yan Di, Mingyang Li *FMGS: Foundation Model Embedded 3D Gaussian Splatting for Holistic 3D Scene Understanding* arXiv preprint arXiv:2401.01970, 2024

[41] M. Ester and H.-P. Kriegel and J Sander and X. Xu *A density-based algorithm for discovering clusters in large spatial databases with noise* KDD, 1996

[42] M. Labbé, F. Michaud *RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation* Journal of Field Robotics, vol. 36, no. 2, 2019

[43] Keiron O'Shea, Ryan Nash *An Introduction to Convolutional Neural Networks* arXiv preprint arXiv:1511.08458, 2015

[44] Morgan Quigley *ROS: an open-source Robot Operating System* IEEE International Conference on Robotics and Automation, 2009

[45] Marcos Barcina-Blanco, Jesus L. Lobo, Pablo Garcia-Bringas, Javier Del Ser *Managing the unknown: a survey on Open Set Recognition and tangential areas* arXiv preprint arXiv:2312.08785, 2023

[46] K J Joseph, Salman Khan, Fahad Shahbaz Khan, Vineeth N Balasubramanian *Towards Open World Object Detection* arXiv preprint arXiv:2103.02603, 2021

[47] Akshay Dhamija, Manuel Gunther, Jonathan Ventura, Terrance Boult *The overlooked elephant of object detection: Open set* In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020

[48] Walter J. Scheirer, Lalit P. Jain, Terrance E. Boult *Probability models for open set recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014

[49] Chuanxing Geng, Sheng-Jun Huang, Songcan Chen *Recent advances in open set recognition: A survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020

[50] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, Terrance E. Boult *Toward open set recognition* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.

[51] van de Ven, G.M., Tuytelaars, T. Tolias, A.S. *Three types of incremental learning* Nat Mach Intell 4, 2022

[52] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, Ziwei Liu *Deep Class-Incremental Learning: A Survey* arXiv preprint arXiv:2302.03648, 2023

[53] OpenAI *GPT-4*[Artificial intelligence model] OpenAI https://www.openai.com/, 2023

[54] Labbé, Mathieu and Michaud, François *RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation* Wiley Online Library, Journal of Field Robotics, 2019