Pablo Azagra Millán

Learning from human-robot interaction

Departamento

Informática e Ingeniería de Sistemas

Director/es

Murillo Arnal, Ana Cristina
Civera Sancho, Javier

ISSN 2254-7606

ISSN 2254-7606

Universidad Zaragoza

1542

Tesis Doctoral

LEARNING FROM HUMAN-ROBOT INTERACTION

Autor

Pablo Azagra Millán

Director/es

Murillo Arnal, Ana Cristina
Civera Sancho, Javier

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2020

PhD. Thesis

Learning from human-robot interaction

Author

Pablo Azagra Millán

Directors

Ana Cristina Murillo Arnal

Javier Civera Sancho

ESCUELA DE INGENIERÍA Y ARQUITECTURA
May 2020

Agradecimientos

Antes de entrar en los pormenores de mi tesis doctoral, quiero utilizar esta sección para nombrar y agradecer a todas esas personas que de una manera u otra me han ayudado a lo largo de los años que ha durado esta tesis. Primero decir que para mí el realizarla ha sido un viaje muy interesante, con sus altibajos (maldita rodilla), y estoy muy contento con los resultados obtenidos. Solo espero que lo que me depare los próximos años sea igual o más interesante.
Las primeras personas que quiero agradecer son aquellas que han hecho posible esta tesis: Ana Cris y Javi. Creo que os lo he dicho alguna vez, pero me siento muy afortunado de que la hayáis dirigido. Habéis tenido bastante paciencia conmigo y me habéis guiado en nuestras reuniones.

Quiero agradecer a todos los integrantes del proyecto IGLU dentro del cual se ha realizado esta tesis. También agradecer a la gente del laboratorio del INRIA en Burdeos por su colaboración y por dejarme usar su robot Baxter para realizar la grabación de los datos que se han utilizado en esta tesis. Agradecer a Manuel Lopes por colaborar conmigo durante la tesis y por permitirme realizar la estancia en Lisboa bajo su tutela, en la cual aprendí mucho y me sirvió para realizar experimentos muy interesantes. Gracias a Margarita Chli por la colaboración en el tema de la interacción con los drones y por la estancia en Zurich.

Durante esta tesis he tenido la suerte de conocer a compañeros de laboratorio en la universidad. Gracias a Berta, Chema, Richard, Carlos, Leo, Melani, . . . Las charlas distendidas en el laboratorio servían para despejar la mente de tanto experimento. También quiero agradecer a los ángeles de Ana: Iñigo, León, Pilar, Bea, Alberto, Sara, Abel y Joaquín. Nuestras reuniones, alguna de las cuales yo mismo malamente dirigí, han sido siempre interesantes y divertidas (y espero que así sigan).

Me gustaría pasar ahora a los agradecimientos más personales. Los primeros como no podría ser menos son los Zaragoza Avengers. Gracias Dani, Merche, Adri, Elena, Pili y Diego por ser como sois y por nuestras quedadas tan random, sin ellas no sería tan fácil desconectar de la tesis para volver el lunes con más energía. Que las quedadas continúen durante muchos años y que nuestra amistad dure para siempre.

Gracias a la Guardia de Hyrule (Dani otra vez, Diego, Susi y Cris) por darme un lugar donde poder hablar de videojuegos de una manera tan divertida. Gracias a Juan por nuestras quedadas esporádicas y tan aleatorias. Gracias a Sara, Álvaro y Jorge por las quedadas y nuestras partidas a los juegos de mesa. Gracias a Rubén, Antonio, Belén y Laura por todas las risas que caen siempre que quedamos.
Quiero agradecer a toda mi familia:
Gracias a mis padres por su paciencia en estos años y por su apoyo e interés. Sin su apoyo cuando tenía la moral baja, los tupers de comida de mi madre y las tardes de domingo viendo una película con mi padre esta tesis no se hubiera completado. Gracias a los Madrileños (Diego, Patri, Lucas y Miguel) por sacarme una sonrisa siempre que nos vemos (aunque sea menos de lo que me gustaría). Gracias Miguel por tus charlas tan interesantes y Lucas por tu sonrisa contagiosa y esos mofletes tan adorables. Gracias a mis tíos Javi y Pepa por vuestro interés y por nuestras charlas. Gracias a mi prima Vanesa con sus pequeñajos por sacarme una sonrisa con sus videos.

Y como se suele decir, lo mejor para el final. Gracias Clara por nuestras noches de videojuegos, series y koreanos, por tus ruidos de bicho mono cuando estaba ofuscado, por todos los abrazos sorpresa y por todas tus visitas a mi laboratorio con una sonrisa. Gracias *SuperCop*.

Gracias a todos por ayudarme durante esta tesis.

Abstract

In the last years, robots are more and more present in many aspects of our daily lives, like domestic household appliances, autonomous cars or personal assistance devices. The interaction between users and these robots is one of the key aspects in service robotics. Such interactions need to be comfortable and intuitive to be useful. They are necessary for the robot to learn and update the world model and its affordances. There are many components needed for the well functioning of these interactive robotic systems.

This PhD Thesis focuses on the visual perception system. For humans, visual perception is an essential component, allowing tasks like object or people recognition and 3D pose estimation. Given the great success of deep learning-based approaches for recognition tasks, recent works focus most on data-driven models, typically trained offline. However, models trained offline on large datasets cannot, in general, address common challenges in real home environment data. Some of these challenges are due to the nature of home environments. For example, new objects that did not exist at the time the training dataset was created appear often. Another relevant challenge is the long-tail distribution of object classes, i.e., objects with sparse apparition and with few or none training samples in common datasets.

This work has been developed within the context of the IGLU (Interactive Grounded Language Understanding) [2] project. Within this context, the overall goal of this PhD Thesis is to **investigate novel methods for a robot to learn incrementally from multimodal user interaction**. Towards this objective, the main contributions of the thesis are:

- *The construction of benchmarks more adequate for learning tasks from natural human-robot interaction.* Most datasets for object learning focus just on images that contain the objects. During this work, to be able to explore learning tasks from human-robot interaction, new data has been collected that combines user interaction with object information.

- *New strategies for object learning from multimodal human interactions.* Object detection typically attempts to find any known object, from previously learned models, visible at certain scene. Differently, this thesis includes novel strategies that analyze the user interactions to focus on the object of interest, and learn information from it incrementally.

- *A novel incremental learning method that can handle fully incremental scenarios*, i.e., new object classes appearing in the scenario or objects that change over the time. In this thesis, a system that can learn classes from scratch and is able to update information on-the-fly is developed and evaluated.

- *An end-to-end prototype for the incremental and multimodal learning from human interactions.* To complete the main objective, a complete prototype with all the steps integrated has been developed.

[2]`https://iglu-chistera.github.io/`

Resumen

En los últimos años cada vez es más frecuente ver robots en los hogares. La robótica está cada vez más presente en muchos aspectos de nuestras vidas diarias, en aparatos de asistencia doméstica, coches autónomos o asistentes personales. La interacción entre estos robots asistentes y los usuarios es uno de los aspectos clave en la robótica de servicio. Esta interacción necesita ser cómoda e intuitiva para que sea efectiva su utilización. Estas interacciones con los usuarios son necesarias para que el robot aprenda y actualice de manera natural tanto su modelo del mundo como sus capacidades.

Dentro de los sistemas roboticos de servicio, hay muchos componentes que son necesarios para su buen funcionamiento. Esta tesis esta centrada en el sistema de percepción visual de dichos sistemas. Para los humanos la percepción visual es uno de los componentes más esenciales, permitiendo tareas como reconocimiento de objetos u otras personas, o estimación de información 3D. Los grandes logros obtenidos en los últimos años en tareas de reconocimiento automático utilizan los enfoques basados en aprendizaje automático, en particular técnicas de *deep learning*. La mayoría de estos trabajos actuales se centran en modelos entrenados 'a priori' en un conjunto de datos muy grandes. Sin embargo, estos modelos, aunque entrenados en una gran cantidad de datos, no pueden, en general, hacer frente a los retos que aparecen al tratar con datos reales en entornos domésticos. Por ejemplo, es frecuente que se de el caso de tener nuevos objetos que no existían durante el entrenamiento de los modelos. Otro reto viene de la dispersión de los objetos, teniendo objetos que aparecen muy raramente y por lo tanto habia muy pocos, o ningún, ejemplos en los datos de entenamiento disponibles al crear el modelo.

Esta tesis se ha desarrollado dentro del contexto del proyecto IGLU (Interactive Grounded Language Understanding) [3]. Dentro del proyecto y sus objetivos, el objetivo principal de esta Tesis doctoral es **investigar métodos novedosos para que un robot aprenda de manera incremental mediante la interacción multimodal con el usuario.**

Desarrollando dicho objetivo principal, los principales trabajos desarrollados durante esta tesis han sido:

- *Crear un benchmark más adecuado para las tareas de aprendizaje mediante la interacción natural de usuario y robot.* Por ejemplo, la mayoría de los datasets para la tarea de reconocimiento de objetos se centra en fotos de diferentes escenarios con múltiples clases por foto. Es necesario un dataset que combine interacción usuario robot con aprendizaje de objetos.

- *Mejorar sistemas existentes de aprendizaje de objetos y adecuarlos para aprendizaje desde la interacción multimodal humana.* Los trabajos de detección de objetos se focalizan en detectar todos los objetos aprendidos en una imagen. Nuestro objetivo es usar la interacción para encontrar el objeto de referencia y aprenderlo incrementalmente.

- *Desarrollar métodos de aprendizaje incremental que se puedan utilizar en escenarios incrementales*, p.e., la aparición de una nueva clase de objeto o cambios a lo largo del tiempo dentro de una clase objetos. Nuestro objetivo es diseñar un sistema que pueda aprender clases desde cero y que pueda actualizar los datos cuando estos aparecen.

- *Crear un completo prototipo para el aprendizaje incremental y multimodal usando la interacción humana-robot.* Se necesita realizar la integración de los distintos métodos desarrollados como parte de los otros objetivos y evaluarlo.

[3]`https://iglu-chistera.github.io/`

Contents

Chapter 1

Introduction, motivation and context

1.1 Introduction

Humans are born like a blank canvas. As they grow, they learn concepts, ideas and actions in a continual manner. They learn both from interaction with the environment, objects or other humans and from previous knowledge transmitted by others. Machine learning is the field that researches how to obtain models for computers to learn concepts, ideas or actions. These methods are usually based on a mathematical approach or a biological-inspired approach. With the learned models, machines are able to execute algorithms to perform autonomously certain tasks of a broad range of complexity. Many of these applications are related to computer vision, such as detecting events [1], navigation [2], automatic inspection [3] or recognizing objects in the environment [4].

Figure 1.1: Many tasks can be performed learning static models a priori[1] (left). However, tasks related to learning in domestic scenarios need to be able to learn and update their models incrementally as new objects appear in the scenarios [2] (right).

Learning to recognize any kind of object is a very common and widely studied problem within the fields of computer vision and machine learning. In general, works on this topic attempt to create representations of different objects and learn how to recognize them if they occur again later. This idea of learning a static model of the objects to be recognized later is interesting and useful if the system goal is to model a fixed subset of the objects that exist in the world. However, these systems are indeed bounded by the limits of the defined subset. While most common object recognition systems are limited by this pre-defined subset, there is an increasing amount of research on strategies to expand, on demand, the number of categories that can be learnt.

Robotics has many applications, such as factory automated tasks [5], that fit static

[1] `https://www.cognex.com/products/deep-learning/visionpro-vidi`

[2] `https://www.visiononline.org/vision-resources-details.cfm/vision-resources/Why-More-Retailers-Are-Adding-Computer-Vision-to-Their-Shopping-Lists/content_id/7468`

models learned from a pre-defined set of fixed training examples. However, there are many other applications where the environment changes and the robot needs to adapt to such changes, as depicted in examples from Figure 1.1. For example, factory applications that intend to bring robots to work autonomously in complex tasks or alongside humans [6, 7], develop robots to do specific dangerous tasks [8] or help and interact with humans [9, 10]. Because of this desire to enable robotic systems to learn how to perform varied tasks autonomously, nowadays machine learning is a key field for robotics. In particular, object learning is an essential ability when the task involves interaction with objects in the environment. Many proposed approaches [11, 12, 13, 14, 15] create an offline subset of objects that the robot is supposed to interact with. Differently, this Thesis is focused on the robot learning the objects incrementally as they appear, without any constraint or prior knowledge.

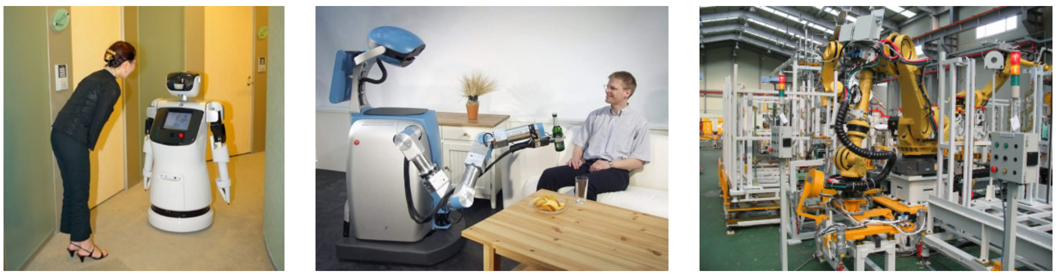
This PhD Thesis explores novel strategies to improve existing machine learning techniques for object learning harnessing the human robot interaction potential.

1.2 Motivation, objectives and contributions

This work has been developed within the context of the IGLU (Interactive Grounded Language Understanding) [3] project. As explained in the project objectives: *Interactive Grounded Language Understanding is an ability that develops in young children through joint interaction with their caretakers and their physical environment. At this level, human language understanding could be referred as interpreting and expressing semantic concepts (e.g. objects, actions and relations) through what can be perceived (or inferred) from current context in the environment. The project goal is to, through a developmental approach where knowledge grows in complexity while driven by multimodal experience and language interaction with a human, research an agent that will incorporate models of dialogues, human emotions and intentions as part of its decision-making process.* Within the context of this project, the main goal of this PhD Thesis is to **investigate novel methods for a robot to learn incrementally from multimodal user interaction**.

There are numerous fields of applications of intelligent and autonomous robots. In particular, the motivation of this work is related to service robotics, towards robotic systems able to learn from natural interactions with human users that require the robot services. Indeed, it is important to note the differences between a manufacturing robot, that works in workshop-like scenarios usually more constrained and repetitive, and a service robot, that works in more varied human environments, as shown in

[3] `https://iglu-chistera.github.io/`



Service Robot Service Robot Manufacturing Robot

Figure 1.2: Service robots vs Manufacturing robots. Variations and dynamic changes are much more frequent in the first case scenarios.

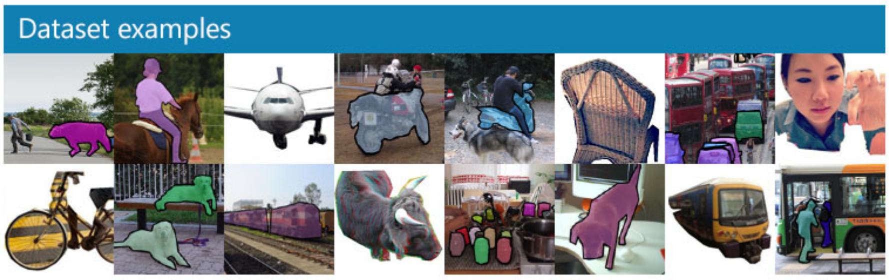
the examples in Figure 1.2. Most service robot scenarios are everchanging, whereas typically a workshop changes little over time and robots in it are meant to do the same actions over and over.

Challenges. Service robots' adoption at our homes is becoming more frequent in the recent years, and robotic systems can be found often in our daily life, in applications such as smart cars and household or personal assistance devices. However, there are still several challenges that prevent broader adoption of these systems. The interaction between users and these systems is one of the key aspects to be developed in service robotics. For these systems to have good usability, the interaction needs to be comfortable and intuitive for the user. Such interactions are important for the system to online learn the world model and capabilities using the user's knowledge and behaviour as natural as possible.

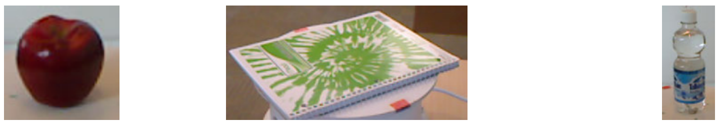
Other open challenges related to better usability of service robotic systems include the adaptability to the environment changes, which is one of the objectives in this work. For humans, visual perception is essential, allowing us key abilities like object category and instance recognition or position estimation. Given the great success of deep learning-based approaches for recognition task, recent works focus most on data-driven models trained offline. However, models trained offline (e.g., [4]) on large datasets (e.g., [16]) cannot, in general, address common challenges in real home environment data, as highlighted in Figure 1.3. These challenges appear due to the nature of home environments, where new objects appear often and they did not exist at the time the dataset was created, or the long-tail distribution of object classes, i.e., there are objects with sparse occurrences and with none or few training samples in common datasets.

Objectives. Towards solutions and advances regarding these challenges, this Thesis develops the following more concrete objectives:

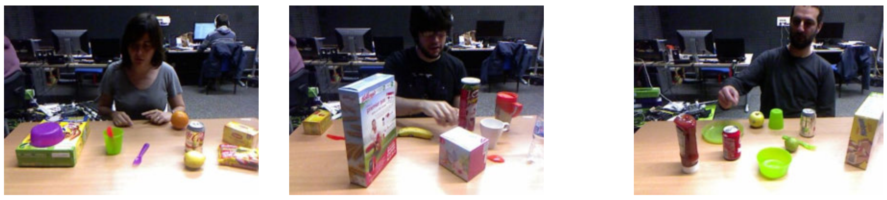
- *To build benchmarks more adequate for learning tasks from natural human-robot*



Coco dataset



Washington RGB-D Object dataset



MHRI dataset

Figure 1.3: Comparison between data from different sources. *Coco dataset*, that contains several thousands of scenes with objects. *Washington RGB-D Object dataset*, that contains 300 common household objects. The presented *MHRI dataset*, that contains 22 kinds of objects. This new dataset brings together properties from the other two, capturing household objects in a realistic scenario, and adding multimodality and human interaction aspects.

interaction. Most datasets for object learning focus just on images that contain the objects. During this work, to be able to explore learning tasks from human-robot interaction, new data needs to be collected that combines user interaction with object information.

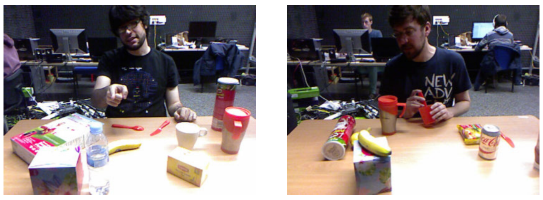
- *To improve existing systems for object learning from multimodal human interactions.* Object detection typically attempts to find any known object, from previously learned models, visible at certain scene. Differently, novel strategies will be designed that analyze the user interactions to focus on the object of interest, and learn information from it incrementally.

- *To develop incremental learning methods that can handle fully incremental scenarios* i.e., new object classes appearing in the scenario or objects that change over the time. In this Thesis, a system that can learn classes from scratch and is able to update information on-the-fly will be designed and tested.

- *To build an end-to-end prototype for the incremental and multimodal learning from human interactions.* To complete the main objective, a complete prototype with all the steps integrated will be developed.

Contributions. In order to fulfill these objectives, during this Thesis, the following contributions have been developed:

- Two novel datasets focused on benchmarking natural human interaction with robotic systems. One of them is designed to benchmark object learning from human robot interaction; another dataset is designed to benchmark robot guidance through natural interaction. Figure 1.4 shows sample images from these datasets, that are detailed in Chapter 2.


MHRI Dataset

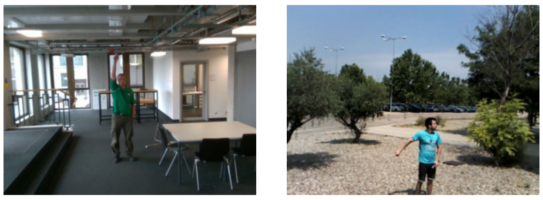
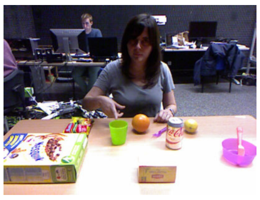

DDIR Dataset

Figure 1.4: Datasets recorded and published as part of this PhD Thesis.

- Incremental algorithm. An algorithm has been developed and used for several incremental tasks within the Thesis, including action and object recognition. It is based on incremental clustering, as explained in more detail in Chapter 3.

- Two novel interaction recognition framework. One framework recognizes human interactions in the object teaching scenario, where an user is on front of the robot teaching an object that is on the scene. The other framework recognizes pointing directions to the drone scenario, where the user in in front of the drone pointing the desired movement direction. Figure 1.5 shows an example of both scenarios. Both frameworks are detailed and evaluated in Chapter 4.


Object teaching scenario

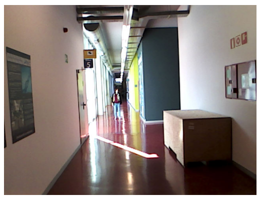

Pointing direction to the drone scenario

Figure 1.5: Scenarios studied in the human robot interaction frameworks in this PhD Thesis.

- Strategies for object segmentation guided by human interaction. Following the framework developed to recognize interactions in Chapter 4, three strategies (one for each type of human interaction recognized, as shown in Figure 1.6) to segment the object that the user is referring to have been developed. They are detailed and evaluated in Chapter 5.

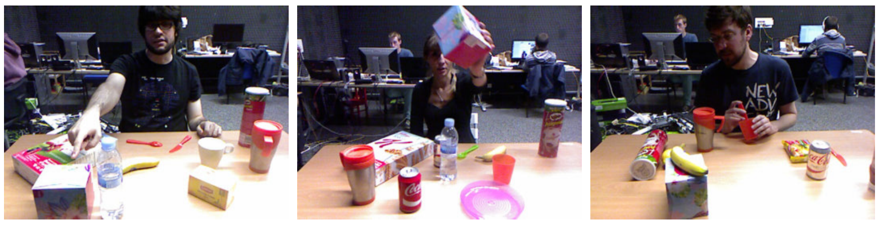

(a) *Point* (b) *Show* (c) *Speak*

Figure 1.6: Examples from the three interaction types studies in this PhD. Thesis.

- Incremental object learning. An analysis of the incremental method in the object learning scenario is done. This analysis compares the influence of static descriptors and parameters in the performance of the incremental algorithm presented in Chapter 3.This study is presented in Chapter 6
- A end-to-end framework to learn object classes from Human-Robot interactions that integrates all the modules developed in previous contributions of this Thesis. This framework is summarized in Figure 1.7 and analyzed in detail in Chapter 7.

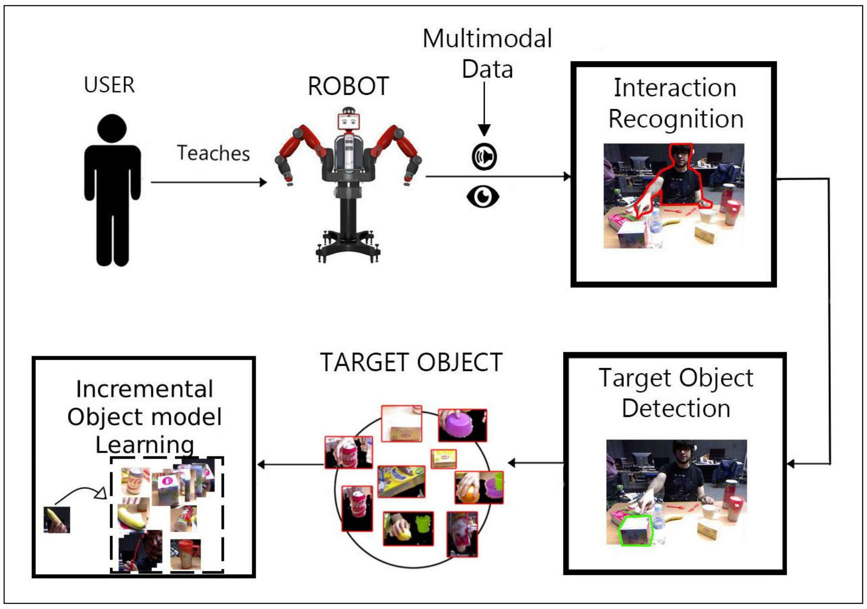


Figure 1.7: Overview of the presented approach for incremental learning from human-robot interactions. A human user teaches a robot new objects through natural interactions (e.g., pointing to it). The robot recognizes the type of interaction from the multimodal recordings, finds the target object region on its camera views and updates the object model incrementally.

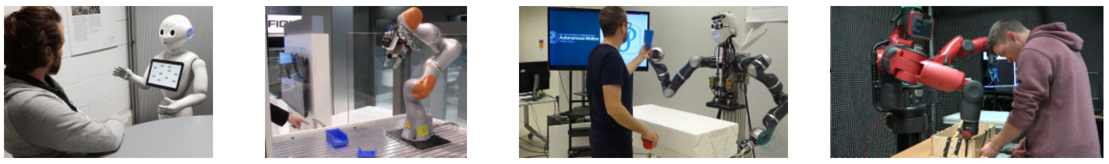

Dialog[5] Gesture-Pointing[6] Gesture-Showing[7] Physical interactions[8]

Figure 1.8: Common interactions in the literature of Human Robot Interaction.

- A novel approach to online descriptors. This approach is focus on the object learning scenario. It uses deep learning descriptor that can be retrained as new data appear using the incremental algorithm. This approach is explained and evaluated in Chapter 8.

1.3 Incremental learning from human interaction

Incremental learning, Human interaction and Object learning are well-known problems both in the Robotics and Machine Learning communities, and therefore, extensively present in the literature. This section presents a quick overview of existing works that bring the three topics together, since it is the context of this Thesis goals and work. Along the following chapters, more specific related works on particular topics are discussed in detail.

A key aspect to consider when discussing these approaches is how the interactions are performed. Figure 1.8 illustrates the most common interactions considered: dialog, hand gestures and physical interaction. *dialog* is one of the most natural interactions for humans and allows the user to have free hands. Krause et al. [17] present a one shot object learning approach where user and robot see a common scene and, through the dialog, the robot learns the properties of all objects in the scene. Similar to them, Skočaj et al. [18] present the same scenario but add a long term memory and multiples dialogs to incrementally learn object concepts.

Related work often uses *hand interaction* (showing or pointing) to lead the robot's teaching. It is actually one of the main interactions considered in this Thesis work. Pasquale et al. [20] present a work where the object is shown in front of the robot (iCub) and the object label is said by the user. They use a CNN as a feature extractor

[5] CATHI - Cognitive Assessment Through Human-Robot Interaction from IBM UK Lab Campus https://festival-of-innovation.eu-gb.mybluemix.net/page55.html

[6] Static pointing gesture identification from FourByThree https://www.cobot-systems.com/software/human-robot-interaction/static-pointing-gesture-identification/

[7] Human interaction with Apollo from Max Planck Institute for Intelligent Systems [19]

[8] 3rdHand project from INRIA FLOWERS team https://www.youtube.com/watch?v=6zdN4QVlRBQ

and a Recursive-Least-Squares to classify. Similar to them, Siam et al. [21] present a dataset with two different settings: teaching objects and robot manipulation tasks. They focus on object segmentation and manipulation with the user help. Another approach learning from hand interactions is the work in Kasaei et al. [22]. They present an approach where the interaction consists of the user pointing to the object in the scene. Differently from the other examples, this work focuses on the 3d object model learning.

Closer to this Thesis goals, there are works that use a *combination of interactions*. Often, one interaction is the main one and the other is for clarification. In this line, He at al. [23] present a framework that learns basic object shape and color through the user pointing and speaking. It creates a mental world for the robot and changes as the user and the environment changes. This mental world is the one that defines the robot actions. Similar to them, Valipour et al. [24] present a scenario where the user is working and asking for an object to the robot. If the object is not found, the user clarifies by pointing to it leading to update the object representation.

Finally, there are also works that learn semantics from objects or physical interactions. For example Aksoy et al. [25] present an approach that uses data from the user point of view. It learns which action and which object is interacting with. Another related example is Toussaint et al [26], where the robot learns through reinforcement learning the collaboration needed to achieve a goal with the user.

Our work belongs to the set of groups using combinations of interactions, since we use dialog and hand gestures. Differently from existing incremental learning works, this Thesis work considers different interactions (including different hand gestures) in realistic and natural conditions for the user. Another key component in this work is the automatic object detection strategy specific for each interaction, all combined in end-to-end systems.

Chapter 2

Human-Robot interaction datasets

2.1 Introduction

One of the first and main aspects to consider in any machine learning problem is which data needs to be gathered or is available. Benchmarking datasets are a key aspect in the literature, enabling to advance research and compare different approaches in a fair manner. In the recent years, datasets are becoming larger as researchers are able to collect more data and share it.

This chapter details the two datasets (*Multimodal Human-Robot Interaction (**MHRI**) dataset and Direction Dataset for Interaction with Robot (**DDIR**) dataset*) that were collected during this Thesis. Besides, it discusses related work and another public dataset (Core50) used for the evaluations on different chapters. For the two datasets that were recorded, the main motivation was to fill existing gaps in the public data corpus that did not allow a proper evaluation of certain aspects in Human Robot Interaction (HRI).

HRI datasets usually focus only on the interaction, normally with only one type of interaction per dataset. The objectives in this work require richer data that contains object teaching information from an HRI perspective using multimodal sensors. Multiple modalities are useful to learn in the target scenarios. Besides, to explore interaction recognition from the sensors of an aerial vehicle, it was required to acquire data from a drone perspective.

2.2 Related datasets

To study and evaluate different approaches for object recognition, the research community has released plenty of public datasets. For example [27, 28] are two well-known datasets targeting object recognition from *RGB-D* images. However, most of the existing datasets focus on offline visual learning and RGB images, like Coco dataset [16] and Imagenet [29]. Interactive, multi-sensor, and multimodal datasets are scarcer.

Multiple aspects should be considered on a dataset targeting interactive learning. We focus, in particular, on multimodality and on realistic HRI settings. In such scenarios, images are expected to have very different appearance than in the previously mentioned datasets for offline learning. The objects are shown by a human through different ways of interaction and the images are seen from the robot point of view. This causes noticeable domain shift: recognizing a pedestrian from a close-up view from a service robot is immensely different from performing the same task with the raw video from a distant wide-angle surveillance camera.

Vatakis et al. [30] shows a multimodal recording approach similar to the set up

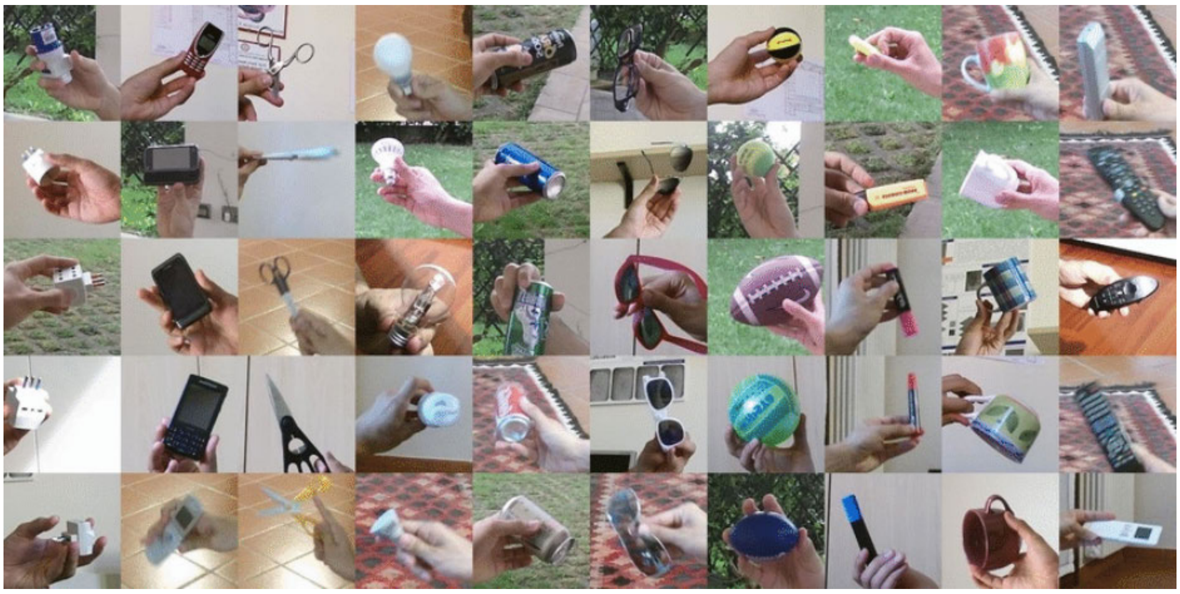


Figure 2.1: Example images of the 50 objects in CORe50. Mosaic from `https://vlomonaco.github.io/core50/index.html#dataset`. Each column denotes one of the 10 categories.

of this Thesis data acquisition, but the purpose of their dataset was to capture the reactions of users to stimuli with objects or images in a screen. Datasets like [31] or [32] capture human-robot interaction from a third-person point of view (POV). It is useful in some cases, but in the context of service robotics, information must be taken from the onboard sensors to be realistic. Temel et al. [33] shows a dataset with the same object in both real and unreal environments and different challenging conditions. Besides the different target applications, the majority of the datasets lack multimodal sensor data, common in human-robot interactive scenarios, like the user speech.

2.2.1 Core50

CORe50 [34], specifically designed for (C)ontinual (O)bject (Re)cognition, is a collection of 50 domestic objects belonging to 10 categories: plug adapters, mobile phones, scissors, light bulbs, cans, glasses, balls, markers, cups and remote controls. There are two types of classification: per instance (50 classes) and category level (10 classes). There are three type of scenario proposed depending on the data separation per training (*New instances, New classes and New instances and classes*).

As explained in their paper [34]: *Objects are hand hold by the operator and the camera point-of-view is that of the operator eyes. Several examples of this dataset are show in Figure 2.1. The operator is required to extend his arm and smoothly move/rotate the object in front of the camera. A subjective point-of-view with objects at grab-distance is well-suited for a number of robotic applications. The grabbing hand (left or right) changes throughout the sessions and relevant object occlusions are often*

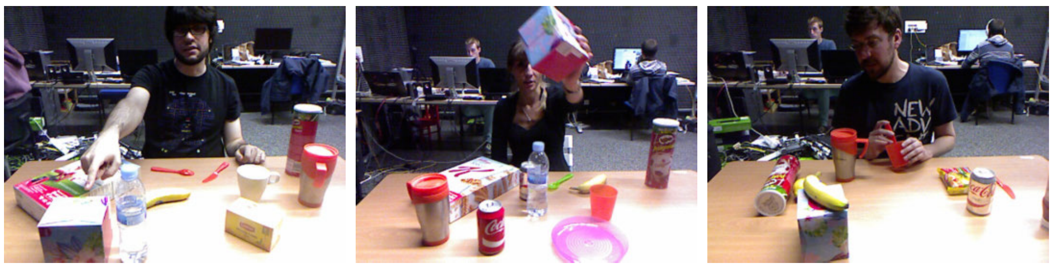


Figure 2.2: Examples from the three interaction types in MHRI dataset. The user says, respectively, (a) "This is a box", while pointing at the box, (b) "This is a box", while holding the box, and (c) "The box is next to the chips and has a banana on top."

produced by the hand itself.

Technical information. The dataset has been collected in 11 distinct sessions (8 indoor and 3 outdoor) characterized by different backgrounds and lighting. For each session and for each object, a 15 seconds video (at 20 fps) has been recorded with a Kinect 2.0 sensor delivering 300 RGB-D frames.

Annotations. The dataset annotations include, for each video, the object label and the category label as well as object position and mask segmentation.

2.3 Multimodal human-robot interaction dataset

One of the contributions of this Thesis is the Multimodal Human-Robot Interaction (MHRI) dataset [1]. It captures the most common natural interactions to teach object classes to a robot, namely *Point*, *Show*, and *Speak*, from a robocentric perspective. Figure 2.2 shows an example for each considered interaction type (captured from the robot frontal camera):

- *Point*: the user points at an object on the table and announces its name.
- *Show*: the user grabs an object, moves it closer to the robot, and utters its name.
- *Speak*: the user describes where a certain object is in relation to other objects.

Table 2.1 summarizes the contents of the dataset. It contains recordings from 10 users and each user performed 10 object interactions of each of the 3 types (*Point*, *Show*, *Speak*), for a total of 300 multimedia short clips. The aforementioned 10 objects per user were picked randomly out of a pool of 22 objects and used by that user for

[1] Available at `http://robots.unizar.es/IGLUdataset/`

Table 2.1: Summary of the dataset content

Users	10	
Interaction Type	3	*Point, Show, Speak*
Interactions per User	30	10 of each type. 1 object per interaction.
Object Pool	22	*Apple, Banana, Big Mug, Bowl, Cereal Box, Coke, Diet Coke, Glass, Fork, Ketchup, Kleenex, Knife, Lemon, Lime, Mug, Noodles, Orange, Plate, Pringles, Spoon, Tea Box, Water Bottle*

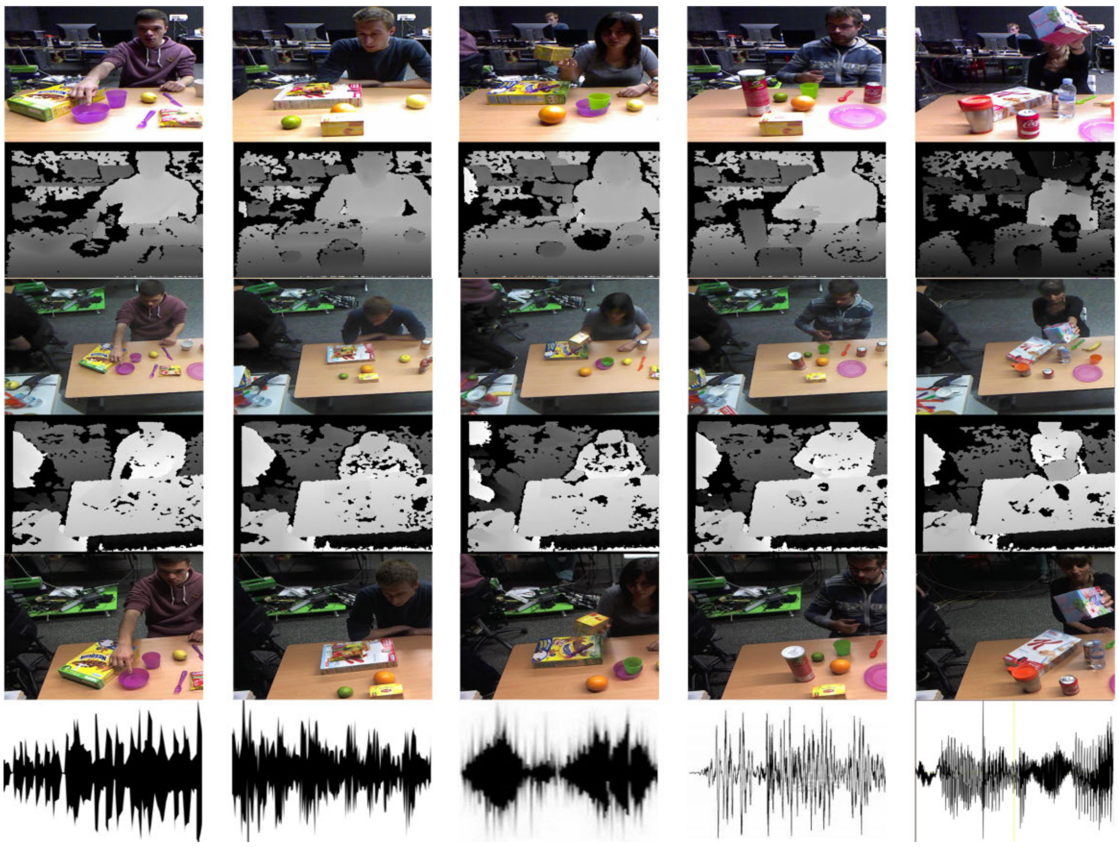


Figure 2.3: Five examples (one user per column) from MHRI dataset. Each row displays a different sensor modality. From top to bottom: *Frontal*-RGB, *Frontal*-depth, *Top*-RGB, *Top*-depth, HD camera, and audio. See Figure 2.4 for the placement of each sensor in the robot.

all their recordings. Figure 2.3 illustrates the different sensor modalities of the dataset for different users.

Technical information. The dataset contains four synchronized streams of data: 2 RGB-D video feeds, from frontal and top point of views, acquired with *Kinect v1* sensors), 1 RGB video feed from a 1280×720 HD camera, and 1 audio feed captured with a studio microphone. Table 2.2 shows the specific data formats available and Figure 2.4 shows the cameras placement in the Baxter robot used for the acquisition.

Table 2.2: Dataset format specifications

Device	**Data**	**Format**
RGB-D Cameras (Frontal & Top)	RGB frames	640x480 JPEG
	Depth frames	640x480 PNG
HD Camera	RGB frames	1280x720 JPEG
Microphone	Audio file	44.1kHz Stereo WAV

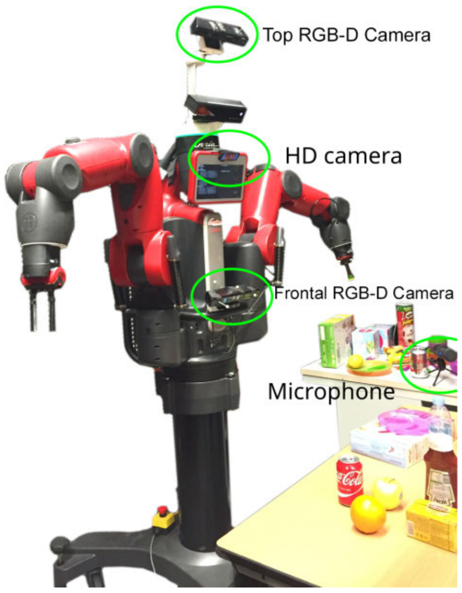


Figure 2.4: Baxter robot used to acquire the dataset. The three cameras and the microphone locations are highlighted.

The *Frontal* RGB-D camera is mounted on the robot chest to give a frontal view of the user and the table. The *Top* RGB-D camera is mounted at the highest point of the robot and has a holistic overview of the scene.

Annotations. The dataset annotations include the list of the objects each user interacted with, the first uttered word (which is either “this”, “that” or “the”), and the label of the object in question for each interaction. Additionally, each frame is timestamped (using ROS[2]) and labeled with the type of interaction (*Point*, *Show*, *Speak*).

2.4 Direction dataset for interaction with robots

The motivation for acquiring this dataset is to train a system to understand the directions pointed by users from a drone perspective. As there was no prior published data on this problem, the *Direction Dataset for Interaction with Robots (DDIR)* has been recorded and released. The data is organized in five subsets (DDIR 1 to 5) depend-

[2] `http://ros.org/`

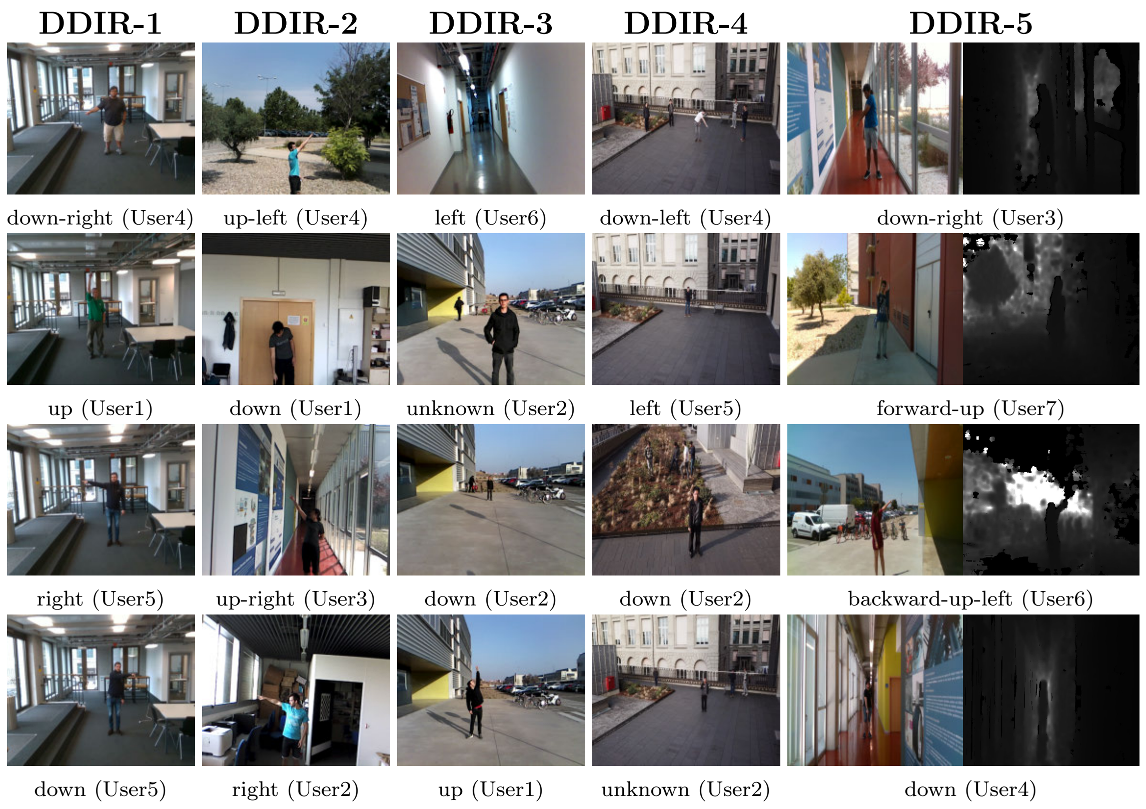


Figure 2.5: Examples of the five sets of the *Directions Dataset for Interaction with Robots* (**DDIR**). Each example specifies its interaction name and the user.

Table 2.3: Summary of the DDIR dataset.

Set	**DDIR-1**	**DDIR-2**	**DDIR-3**	**DDIR-4**	**DDIR-5**
Image resolution	640×480	640×480	823×480	823×480	$823 \times 480^{+}$
# users	5	3	6	5	7
# actions per user	8*	8*	48*	64*	52*
User distance (m)	5	2.5-5	2.5-10	5-10	2.5-5
# Indoor/Outdoor scenarios	1/0	3/1	2/1	0/2	4/3
# direction classes	8	8	8	8	26
# frames, direction classes	1393	3212	16430	11711	22628
# frames, *unknown*-class	2035	2093	6356	16520	4726

* Evenly distributed for each class.

$^{+}$ RGB-d recording.

ing on different characteristics. Figure 2.5 shows sample frames of each of these five sets, which are detailed next. Figure 2.6 shows representative examples of the dataset classes. Table 2.3 summarizes the data technical specifications.

DDIR-1 This set was recorded within a single indoor scenario (at ETH Zurich), with the camera plugged into the base processing station. Since this set is used for training and testing, the data was split following a cross-validation strategy, where for each fold we always leave one user data out of the training set (**DDIR-1, *train-fold*:** images

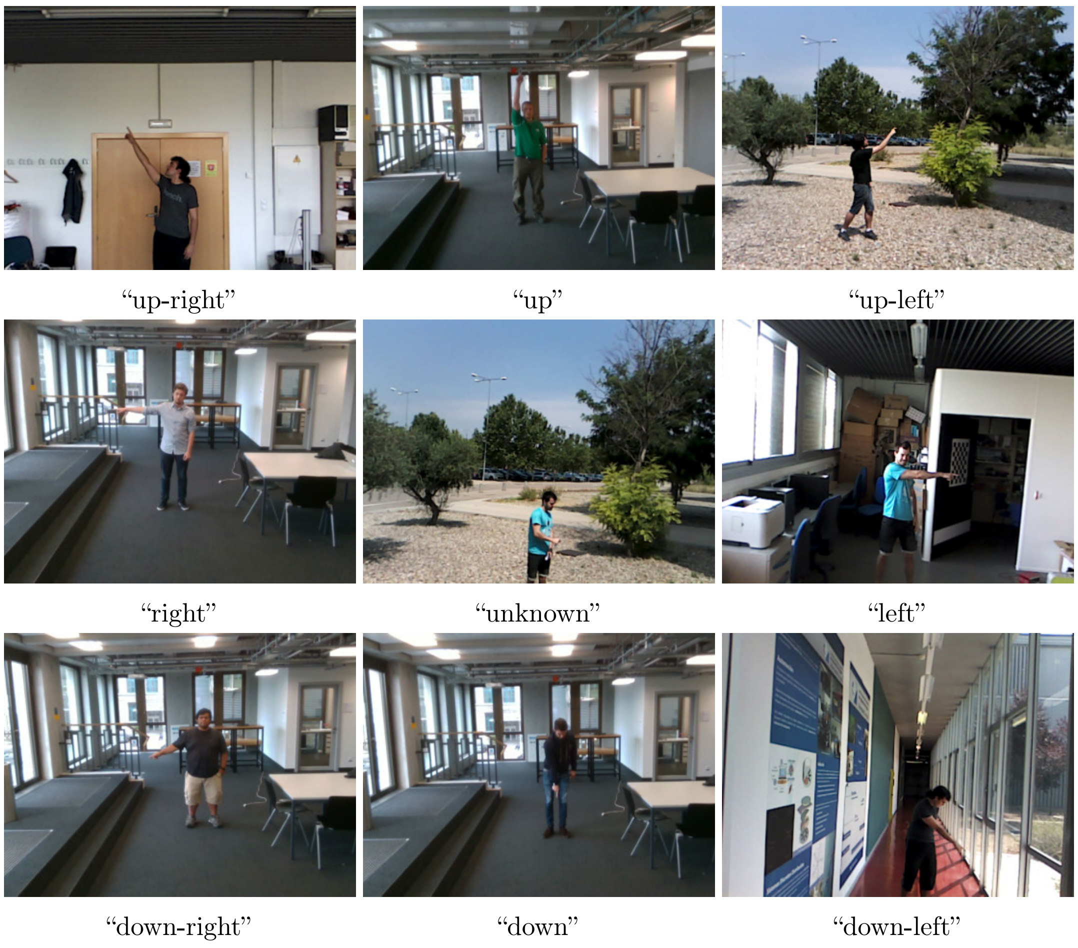


Figure 2.6: Example of each 2D direction class considered in our data. The label of each class (right under each image) is the direction in which the person is pointing. The "unknown" class is the most heterogeneous because it covers every image in which the pointing direction is not clear (or the user is not pointing).

from 4 of the 5 users in the dataset. **DDIR-1val, *validation-fold*:** images of the remaining user, used for the validation).

DDIR-2 This set was recorded in three different indoors and two outdoors scenarios (at I3A Zaragoza), also with the camera plugged into the base processing station. This data is used to evaluate robustness to scene and user changes. The training is done on DDIR-1 set and then test on this set. The domain change is challenging, but allows us to demonstrate how our algorithm generalizes.

DDIR-3 This set was recorded in three different scenarios: two indoors and one outdoors (at I3A Zaragoza). The different users perform the pointing gestures 2.5, 5 or 10 metres away of the camera. This dataset is used together with the next one for testing the complete system.

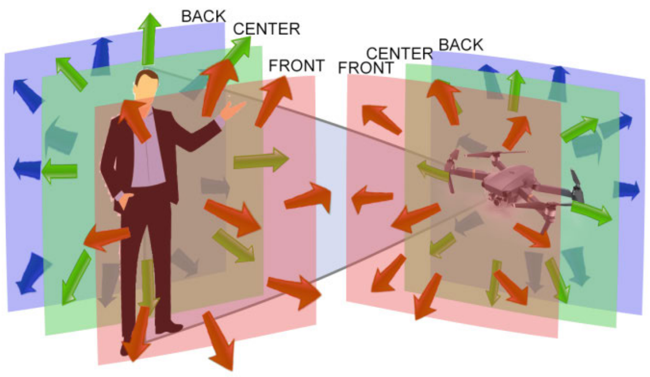


Figure 2.7: Illustration of the 26 navigation directions recognized by our system. In the experiments where only 8 directions are named, we use the CENTER plane directions.

DDIR-4 This set was recorded in two different outdoor scenarios (at ETH Zurich), and in this case, with the camera aboard the drone. The drone is hovering at 3 metres above the ground. The different users perform the pointing gestures 5 and 10 metres away from the drone. In half of the footage there are other people in the background while the user is pointing. This data is essential to demonstrate robustness to different image viewpoints due to the fact of having the camera on the base station or attached to the drone.

DDIR-5 This set was recorded in seven different scenarios: four indoors and three outdoors (at I3A Zaragoza). The users perform gestures at approximately 5 metres from the camera. The camera used in this set has infrared sensors that let us also record depth information of the footage. This set is used for expanding the system to 3D movement, so the pointing directions cover all the 26 shown in Figure 2.7. This set uses a data splitting similarly to set DDIR-1: **DDIR-5** is the ***train-fold*** and **DDIR-5val** is ***validation-fold***.

2.4.1 Annotations

The dataset annotations include, for each frame, the corresponding 2D pointing direction out of a 8-bin representation (except DDIR-5 which has 26 possible directions in 3D) and an *unknown* class label. The labels are text-based ("up","down",...).

Chapter 3

Incremental learning algorithms

3.1 Introduction

The need for incremental learning systems comes from the fact that models trained offline on large generic datasets cannot, in general, address certain challenges and problems from home environment real data. One typical challenge is the long-tail distribution, i.e., objects that appear rarely and for which few or none training samples exist in generic datasets. Another challenge is the changing nature of the environments, with new objects appearing, e.g., food products that did not exist when the large training datasets were created. In order to address these and other problems, robotic perception should have lifelong learning components.

This chapter details the proposed algorithm for incremental learning. This algorithm is based on incremental clustering with data selection strategies. The validation and evaluation of this algorithm is included in following chapters, together with the details of the applications using it.

3.2 Related work

In recent years, significant advances have been made in the field of incremental learning, many of them applying deep learning techniques. *Open-class* approaches, i.e., those able to add new categories as the data comes, are particularly relevant for our work.

Li et al. [35] create and train new classification layers as new classes are added and fine-tune the rest of the network to maintain the outputs for older classification layers. Following this work, Rannen et al. [36] use a similar setting, one shared model and several classification layers, but at training time they add one feature-autoencoder per class. The new autoencoder is trained on the data for the assigned class and uses a loss to keep the build-up error on the old ones. More similar to our work, Rebuffi et al. [37] use deep learning to obtain image representations that can be incrementally updated. They set a limit in the total number of stored examples and classify using the average feature of each class examples. With our approach, we outperform this work in the Core50 dataset at a lower cost, as we do not fine-tune the network.

Other works apply deep learning techniques to incrementally bind multimodal attributes to the objects models, like Xing et al. [38]. Here, the Perception Coordination Network acquires and binds multimodal concepts between different sensory modules in an online manner. It uses two levels of neurons inspired by the brain structure and separates lower neurons depending on the modality of the input. Our work only applies deep learning techniques to extract image features, because training a network online implies too high computational cost and the binding of concepts is out of our scope.

Besides deep learning based strategies, many former approaches for incremental or online learning can be found in the literature. Passive-Aggressive algorithms [39] use an offline learning algorithm as a base and incrementally modify their parameters. [40] present a variation of SVM that is able to change the support vector online. We can also find online variations or combinations of K-means clustering algorithms. Murty et al. [41] present an approach that combines the k-means algorithm with multilevel representation of the clusters. Likas et al. [42] present a global K-means that adds a new cluster at a time and dynamically updates the other clusters by applying the k-means algorithm multiple times. More recently, Mensink et al. [43] present an incremental Nearest Mean Classifier which uses nearest neighbor with the mean of each class for classification and also for generalization.

Other group of approaches apply a data transformation based on Self-Organizing Maps (SOM), as a base to incrementally update the nodes in the network. For example, Furao et al. [44] present an online unsupervised system with an incremental update of a neural network based on SOM (SOINN). Xing et al. [45] present a more recent variant of the Self-Organizing Incremental Neural Networks that incrementally transforms the nodes in the layers of the SOINN using the local distribution. Gepperth et al. [46] use SOM to reduce the dimensionality of the data in the hidden layer, but it needs to keep all the data in memory for re-training.

These approaches work with seeds for each class based on the existing data and cannot add new classes over time. Differently, the goal in this Thesis work is to be able to learn completely from scratch and increase incrementally the number of classes.

Incremental learning is a paradigm very suitable for robotics, where the data typically arrives sequentially, and the robot needs to keep the best model up to date at real time, such as mapping in [47] or inverse dynamics incremental learning in [48]. The same way, Angeli et al. [49] present an incremental method to build a model to recognize visual loop-closures. There are multiple examples that propose how to incrementally adapt environment visual models as the robot moves. These approaches are often based on Gaussian Mixture Models that can be easily updated and maintained to recognize regions of interest for the robot [50, 51]. In robotics, there are situations where the robot interacts directly with the scene, e.g., grasping and moving an object, to build an incremental object model [52, 53, 54]. The proposed approach is complementary to these works, as this Thesis focuses on the human interaction. This interaction is needed in real scenarios, e.g., if the object to be learned or explored is out of reach of the robot.

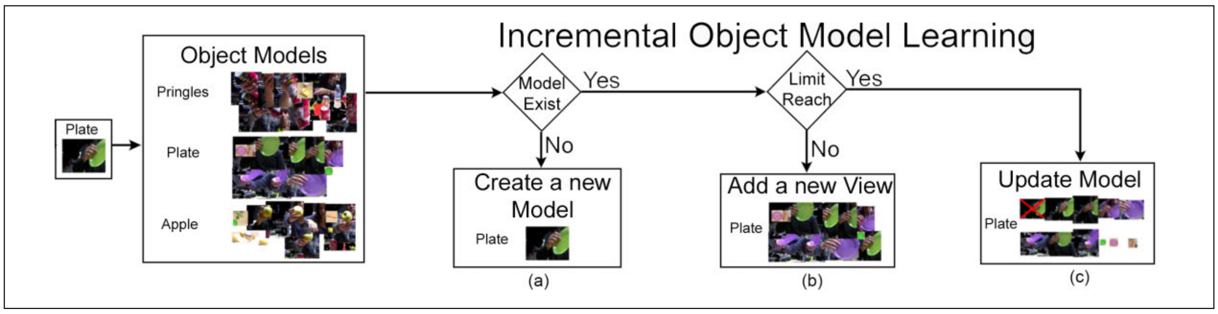


Figure 3.1: Summary of the use cases considered in the proposed online learning algorithm.

3.3 Our incremental learning approach

The online learning algorithm proposed in this work is inspired by incremental clustering approaches, and it is used in two different stages of our pipeline: interaction recognition and object model learning.

The algorithm works as follows. We represent each class model $\mathcal{C}$ with a hierarchical set of clusters in the descriptor space $\mathcal{C} = \{C_1, \dots, C_l, \dots, C_N\}$, where C_l represents the set of clusters for class l, $C_l = \{C_l^1, \dots, C_l^j, \dots, C_l^s\}$.

Each cluster C_l^j groups a representative subset of j descriptors for class l, most of the time corresponding to a specific viewpoint. We assign an integer score τ_l^j to each cluster C_l^j to gather evidence of the suitability of such cluster via consensus.

As new samples arrive, existing clusters evolve and update their centroids (used as representative descriptors) and scores. Besides, new clusters can be created for new classes. Figure 3.1 represents the possibilities when a new training sample is given to the incremental learning algorithm. The total number of classes N is not limited by construction but, in order to avoid unlimited growing, the number of clusters per class is limited by a predefined size k. Algorithm 1 summarizes the proposed strategy, and its main components are discussed next.

Algorithm 1 Incremental learning algorithm

```
 1: procedure Inc.Train($e$,$l$)
 2:     $C_l^e$ = Create_Cluster($e$,$l$)
 3:     $C_l$.add($C_l^e$)
 4:     if $l$ not in $system$.Labels then
 5:         $system$.add_label($l$)
 6:     else
 7:         if $C_l$.is_full($k$) then
 8:             if $system$.updates % $n_{updates} \neq 0$ then
 9:                 // Merge similar cluster
10:                 max_distance = 0
11:                 min_distance = inf
12:                 for each $x$ in $C_l$ do
13:                     for each $y$ in $C_l$ do
14:                         if $x \neq y$ then
15:                             distance = $D_B(C_l^x, C_l^y)$
16:                             if distance < min_distance then
17:                                 min_distance = distance
18:                                 $\hat{x} = x$
19:                                 $\hat{y} = y$
20:                             end if
21:                             if distance > max_distance then
22:                                 $\hat{z} = x$
23:                                 max_distance = distance
24:                             end if
25:                         end if
26:                     end for
27:                 end for
28:                 $C_l^s$ = Merge($C_l^{\hat{x}}$,$C_l^{\hat{y}}$)
29:                 update_score($C_l^s$,+1)
30:                 update_score($C_l^{\hat{z}}$,−1)
31:             else
32:                 // Remove worst scored cluster
33:                 $w$ = $C_l$.get_worse_score()
34:                 Erase($C_l^w$)
35:             end if
36:             $system$.updates +=1
37:         end if
38:     end if
39: end procedure
```

3.3.1 Incremental model update

The input to our algorithm is a descriptor e corresponding to a new training sample, and its label l. As represented in Figure 1.7(a), a new cluster C_l^e is created with e as its centroid and l as associated label. The new cluster C_l^e is added to the list of clusters C_l. If label l does not exist in the *system*, it is added to it. If C_l has reached the maximum number of associated clusters k, as represented in Figure 1.7(b-c), the following two cases can happen.

- **The first** $n_{updates}$ **times**, the algorithm computes the pairwise distance of each cluster in C_l with respect to the rest. We compute the distance D_B between the cluster centroids and find the pairs $\{\hat{x}, \hat{y}\}$ and $\{\hat{w}, \hat{z}\}$, corresponding respectively to the maximum and minimum intercluster distances. In the experiments, we compare differents type of distance (Euclidean, cosine, Battacharya,...) with different kind of descriptors.

$$\begin{aligned} \{\hat{x}, \hat{y}\} &= argmin_{x,y}\{D_B(C_l^x, C_l^y)\} \ni x \neq y \\ \{\hat{z}, \hat{w}\} &= argmax_{w,z}\{D_B(C_l^w, C_l^z)\} \ni z \neq w \end{aligned} \tag{3.1}$$

The two clusters at minimum distance, $C_l^{\hat{x}}$ and $C_l^{\hat{y}}$, are merged into one single cluster C_l^s. The resulting cluster is assigned the centroid of cluster with the better score between τ_l^x and τ_l^y and its score incremented by one. The score (τ_l^z) of the cluster at maximum distance $C_l^{\hat{z}}$ is decremented by one.

- **After** $n_{updates}$, the cluster $C_l^{\hat{w}}$ with the worst score $(\tau_l^{\hat{w}})$ is removed and replaced by the cluster of the new sample.

$$\hat{w} = argmin_w \tau_l^w \tag{3.2}$$

In addition to the approach to update the model described above, several simple baselines were considered (*random* and *always similarity*, where the merging happens always with the closest clusters) as alternative criteria for the cluster reorganization. This approach prevents too much intercluster similarity by merging certain clusters, and also penalizes and eventually removes clusters too different from the rest (possibly corresponding to outlier data). However, when using an online descriptor the *always similarity* gives a better performance since the penalty for outlier data is less important than the intercluster similarity. Since the available data is likely to contain significant noise and have a different distribution than typical public datasets, there is no clear benefit from pre-training the proposed models on such data.

3.3.2 Classification of new samples

A new sample is assigned to the existing classes applying a k-Nearest Neighbor (k-NN) approach, following eq.(3.3). The distance from the new sample descriptor e is computed to all the cluster centroids in the proposed model (C). The algorithm sorts them and obtains the k-top clusters ($x_{0:k}$ in the eq. 3.3). Each existing cluster, x, has a label assigned, l^x, and the new sample is classified as class $l^{\hat{x}}$, where $l^{\hat{x}}$ is the $Mode$ from the top k labels obtained.

$$\begin{aligned} x_{0:k} &= \text{sort}(D_B(e,\ C))[0:k] \\ l^{\hat{x}} &= Mode(l^{x_{0:k}}) \end{aligned} \tag{3.3}$$

Chapter 4

Human-robot interaction recognition

4.1 Introduction

Humans interact with other humans for a variety of reasons. When humans interact with robots, the number of reasons decrease since all social interactions are not necessary. Usually, robots are the ones interacting with humans in order to help them or support them in a task. However, humans might also need to interact with robots when they need to teach them new things. Towards a long term goal of allowing robots to learn from human users, the first step is to know the type of interaction that is being used for teaching. In this chapter, the following approaches to HRI recognition are presented, evaluated and discussed:

- Offline Human-Robot interaction: This approach evaluates each frame of the video against a offline model and makes a consensus between them. This model is pre-trained on the interactions the system needs to recognise.
- Online Human-Robot interaction: A different approach where there is an active interaction needed, since the user may need to do some clarification. If the model is not certain of the recognition, it will ask the user if it is correct.
- Human-Drone interaction: The scenario for this case is different from the previous two ones. It consists of a user giving directions to a drone.

4.2 Related work

There are plenty of applications where a service robot assists a human user and learns or updates its models interacting with him/her. Bohg et al. [55] present a survey on interactive perception and how it can be leveraged for robotic actions, with specific references to interactive object modeling. Some works focus on the value added by the robot motion. For example, Park et al. [56], present a robot that interacts with users to perform daily routines, and Reiser et al. [57], present a robot with perception, navigation and manipulation capabilities that interacts with a user via a touchscreen. Other works focus on the interaction in a workshop like scenario, more closer to our setup. For example Baraglia et al. [58] aim to decide if it is the robot or the human who should take the initiative in collaborative work, and Dumora et al. [59] present an approach where the robot decides the action to perform next based on a set of haptic cues from the human user.

More similar to our work, other approaches study how the user can teach the robot, like Skočaj et al. [18] and Krause et al. [17]. There, the user maintains a conversation with the robot to teach objects and attributes from a common view of the table scenario.

Valipour et al. [24] present a work where the user can correct the robot when an object is not found using voice commands and pointing. In Siam et al. [21], HRI help to improve the segmentation of a target object and to learn a better model. Other works focus on teaching actions like Aksoy et al. [25], which is able to incrementally learn semantic event chains (SECs) extracted from actions using human demonstration.

Very related to our work, Pascuale et al. [20] use Convolutional Neural Network (CNN) based features and Support Vector Machine (SVM) classification for teaching visual models to a robot. The training data consists of egocentric images, where a human presents an object in front of the robot. Camoriano et al. [60] harnessed that data (vision-only data and user interactions consisting only of users showing the objects to the robot) and uses a variation of Regularized Least Squares for incremental object recognition. Similarly, Kasaei et al. [22] use the point cloud to obtain a 3D descriptor and incrementally learn objects looking where the user points.

Other set of works explore when the robot is able to interact also with the objects. Lyubova et al. [61] learn objects models using point-feature descriptors and Bag of Words (BoW) models in two steps. The first one is based on just observation (either from the table or from a human showing the object) and the second one includes the robot interaction with the objects. He et al [23] present an incremental network, called Adaptative Neural Gas (ANG) that learns shape and color of simple objects in the robot workspace using visual-audio input and the possibility of the robot to ask for more information. The contribution of our proposal over these works is a more generic strategy, towards a more natural human-robot interaction, using multimodal data and enabling different types of user interactions (point, show and speak) to learn new objects.

4.3 Offline human-robot interaction recognition

Classifying the type of interaction performed by a person using only visual data is considerably challenging. The work of [62] show that the combination of language and vision can lead to a substantial improvement. Our offline Human-Robot interaction recognition approach uses visual and language features in a nested SVM-based classification.

4.3.1 Recognition algorithm

We propose the following interaction recognition, using the language and visual features, based on two nested classifiers:

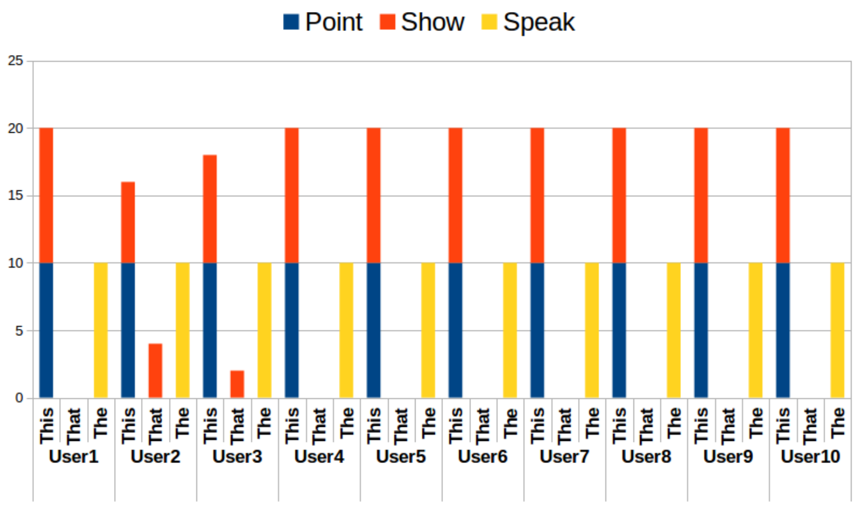


Figure 4.1: Language feature occurrences in all recordings from the MHRI dataset. They are grouped per type of interaction and per user.

1. Binary discrimination between *Speak* videos and the other two types using the language features.

2. SVM classification into *hand* vs *no-hand* classes of sliding window-based patches, trained with random patches of the dataset and manually selected patches of hands. This step only uses the *HC* descriptor due to its high efficiency and good performance at removing most of the *no-hand* patches.

3. SVM classification of resulting *hand* patches into *Point* or *Show* classes. Here we use both the *HC* and the *HOG* descriptors.

4. Assign a label, *Point* or *Show*, to each video according to the label obtained by the majority of its frames. All windows from each video are labeled as that action for the next step.

Language features. The built system uses a simple language feature consisting of the first word of the user's narration. In the acquired dataset this word is either *this* or *that* for *Point* and *Show* interactions or any other word for the more descriptive *Speak* interaction. This feature is not discriminative enough to separate the three interaction classes, as we show in Figure 4.1. It clearly separates *Speak* interactions, but cannot differentiate between *Point* and *Show*. Separating *Speak* is particularly valuable, as there are no specific visual patterns associated with this interaction.

Visual features. Before computing the visual features, in order to focus on the user and table regions, the background is removed using two strategies: a standard background removal procedure, based on sliding-window average of all the previous

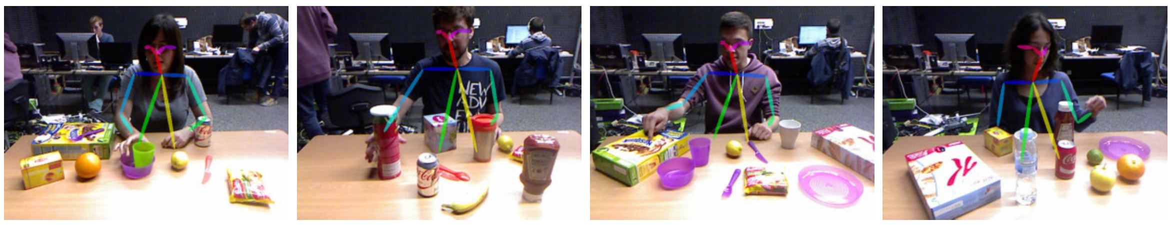

(a) *Show* (b) *Show* (c) *Point* (d) *Point*

Figure 4.2: Examples of skeleton detections. Notice that, in 4.2b, the hand joint is predicted even when it is occluded.

frames, and a depth map based filter, where all image pixels with a depth value over a threshold of $1.7m$ are removed (based on the distance to the user and the table). These two filters are applied on the image and a sliding-window filter (window size of 100×100 pixels, stride of 10 pixels) runs over the masked image to reject windows where more than 30% of the pixels were removed by either one of these filters. Then, visual descriptors are computed on the accepted windows. The following two different descriptors are evaluated:

- *Color histograms* $HC = [H_r \ H_g \ H_b]$, with $H_i = \sum_{x,y} p_i(x,y) \bmod B$, where p_i is pixel i component value and B the number of bins.
- *Histogram of Gradients* (*HOG*), as described in [63].

4.4 Online human-robot interaction recognition

The recognition of the type of interaction has several challenges. First, gestures are very different between users. Second, such gestures are also highly dependent on the camera viewpoint. Because of these two challenges, together with the small amount of data in our dataset, an online approaches to this problem is developed.

Due to the low number of users available, none of them converged to a model that generalizes well for new users, giving in most cases random accuracy. Instead, the incremental learning algorithm described in Chapter 3 is used. It uses interactive supervision from the user if the classifier output is not conclusive.

4.4.1 Visual analysis of the user

To recognize the user interaction, an analysis of the possible users in the robot field of view and then focus on the main user hand.

To identify all skeleton joints for all the people in the image, a CNN-based approach by Cao et al [64] is used in the pipeline. The locations and associations are learned jointly using a specific representation (denoted as Part Affinity Fields or PAFs) with all the individuals on the scene. Any skeleton segmentation strategy could be use in the pipeline, but this particular approach was chosen for several reasons. It shows good performance in our images, even with considerable occlusions of the user, and its computational load is reasonably low. Figure 4.2 shows several examples of estimated skeletons where its accuracy can be appreciated.

After the skeletons have been detected, the pipeline focuses on the largest individual. A 200×200 patch around the hand joint is extracted, which is expected to contain the hand. The visual features for the interaction classification, detailed next, are computed over this patch.

4.4.2 Multimodal features

Language features. In this online version, the language features is the same as the offline version. These features consist on the first word of the user. As shown in Figure 4.1, it is not discriminative enough, but it is helpful enough to separate *Speak* from the other two interactions. This is important since there are no specific visual patterns associated with this interaction.

Visual features. *Histogram of Gradients* (*HOG*) [63] is computed in the depth channel of the hand patch. This method focuses on the hand shape and it is quite independent of variations in the hand color.

4.4.3 Incremental interaction recognition

This process is detailed in Algorithm 2. At the start, there are no training samples for a given user. For the first n_vids samples (in our experiments $n\_vids = 4$), the algorithm chooses one type of interaction randomly and asks the user if the predicted interaction is correct. Depending on the answer, the label is corrected, and the labeled video is used to train the incremental model. After n_vids video samples, each video frame is classified according to the hand patch found on it. The video is assigned the class of the majority of the frames classified with high confidence. If this majority is less than min_prob, the algorithm asks the user for the actual interaction type and the model is re-trained.

4.5 Human-drone interaction

With aerial robots, such as Unmanned Aerial Vehicles (UAVs), promising great value in a variety of applications ranging from industrial inspection to search-and-rescue and

Algorithm 2 Incremental interaction recognition.

```
min_prob = 85%
videos_processed = 0
n_videos = 4
min_dist = 0.56
function EXTRACT_DESCRIPTOR(Frame_RGB)
    Skeleton = obtain_skeleton(Frame_RGB.Front)
    Hand_patch = Crop_Hand(Frame_RGB.Front,Skeleton.wrist)
    Descriptor = Calculate_HOG(Hand_patch)
    return Descriptor
end function
function TRAIN_INCREMENTAL(Video_RGBD)
    Interaction = Ask_User()
    for each Frame_RGB in Video_RGBD do
        Descriptor = Extract_Descriptor(Frame_RGB)
        Inc.Train(Descriptor, Interaction)                    ▷ See Alg. 1
    end for
    return Interaction
end function
function INTERACTION RECOGNITION(Video_RGBD,speech)
    if speech.get_first_word() ≠ ("This" | "That") then
        return "Speak"
    else
        if videos_processed < n_vids then
            videos_processed += 1
            Interaction = Train_Incremental(Video_RGBD)
            return Interaction
        else
            videos_processed += 1
            Votes = []
            for each Frame_RGB in Video_RGBD do
                Descriptor = Extract_Descriptor(Frame_RGB)
                Class, Distance = Inc.Test(Descriptor)
                if Distance < min_distance then
                    Votes.add_vote(Class)
                end if
            end for
            Interaction,Confidence = Process_Results(Votes)
            if Confidence > min_prob then
                return Interaction
            else
                return Train_Incremental(Video_RGBD)
            end if
        end if
    end if
end function
```

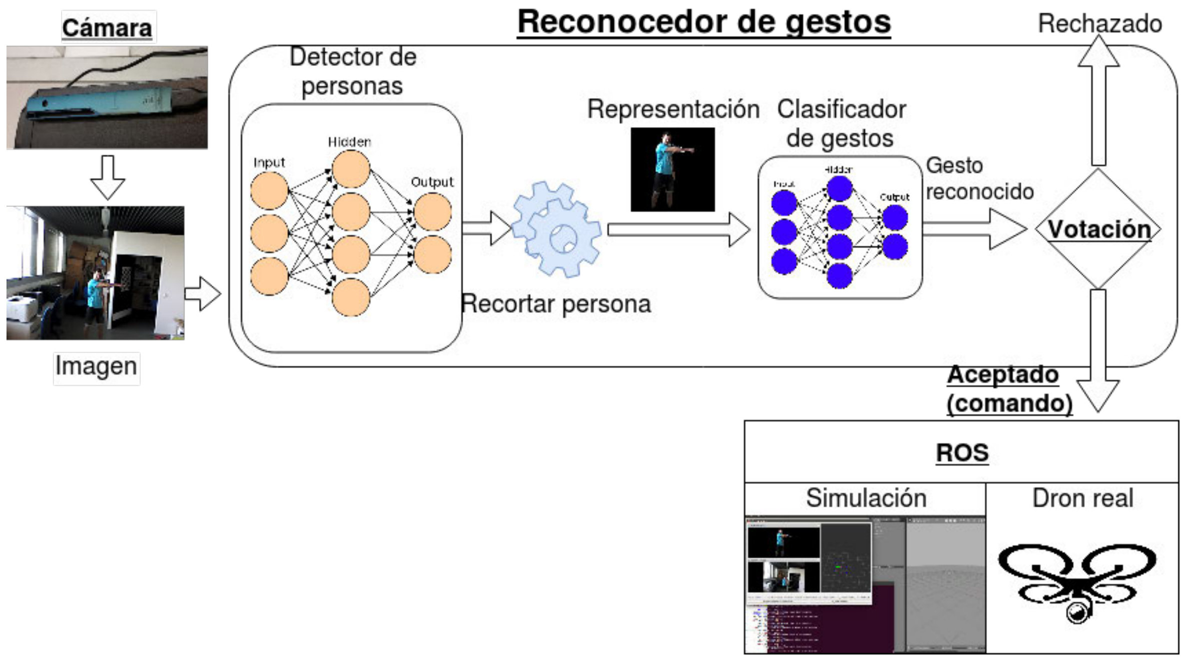


Figure 4.3: Drone guidance by natural pointing. The presented system receives the captured video as input, estimates the user pointing direction seeking consensus across a window of frames and sends the command to the drone using ROS.

crop monitoring, the demand for user-friendly interfaces that eliminate the need of an expert pilot becomes more evident. In this section a framework to recognize directions from the UAV POV is presented, and next section will show experimental results.

4.5.1 Overview

As summarized in Figure 4.3, the pipeline for high-level drone guidance through natural pointing interactions has the following components:

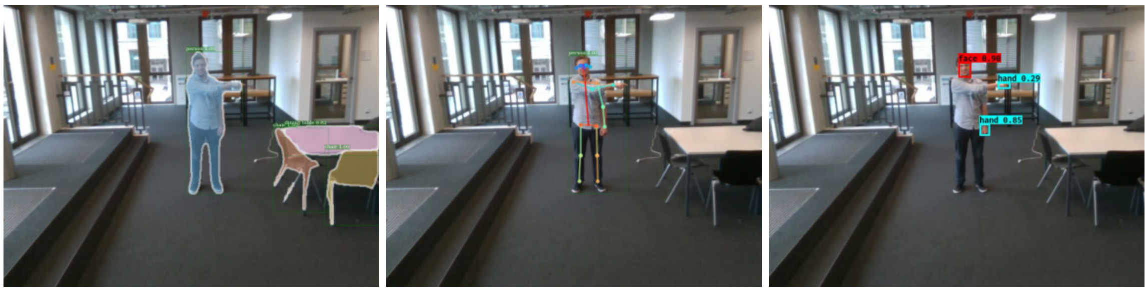
1) Input. Two setups were considered, the camera is placed on board the robot (to enable configurations where the robot operates near the human user) or at a base station (if the robot operates at scenarios not accessible for the human).

2) Pointing direction recognition. This is the core component of the pipeline and consists of three stages.

Person detection and representation (per frame). The detection of the user is processed each frame, exploring three alternatives detailed in section 4.5.2.

Direction classification (per frame). To estimate the direction to where the detected person is pointing, the space of possible navigation directions (see Figure 2.7) is discretized and formulate this step as a classification problem. The alternatives explored are also detailed in section 4.5.2.

Consensus (over temporal window). The classification of natural gestures addressed in this work is challenging due to the high variability and limited training data. A



(a) (b) (c)

Figure 4.4: Example of the three alternatives for person detection and representation. (a) Segmentation. (b) Skeleton. (c) Hands&Faces

voting scheme over several frames within a small sliding window (5 frames in our experiments) is applied as an effective way to add robustness to our classification. This consensus block receives, from the previous block, the estimated pointing direction and a confidence value for each frame within the window. It only accepts a command if 4 of the 5 last images agree on the command. If the confidence on the classification of a frame is below certain threshold (0.5 in our experiments), the vote of that frame is ignored.

3) Output (Drone command). Once there is consensus on a certain direction, the corresponding command is sent to the real or simulated UAV via network message. The message contains the x, y and z coordinates of the relative position where the drone has to move to.

4.5.2 Pointing direction recognition in RGB

This subsection details the recognition of the user pointing direction, the core component of our pipeline.

The first step is to detect the persons in the scene, to select the region of interest (ROI) and represent the data adequately for the direction classification stage. Regarding this final direction classification step, there are two main constraints to consider. First, due to the interactive application targeted, the system needs to react to the user actions in acceptable rates for an interactive application (*i.e.*, a few milliseconds). Second, since the amount and heterogeneity of the labelled data available is fairly small (see Chapter 2 for a detailed description of the *DDIR* dataset used in the evaluation) rather than training from scratch large models, simple models or transfer learning techniques need to be considered.

Following the three most common alternatives in the literature for people detection or segmentation in images, the three strategies were built and detailed next (illustrated in Figure 4.4): *Segmentation*, *Skeleton* and *Hands&Faces*.

Segmentation. This strategy is based on a well known semantic segmentation CNN, Mask R-CNN [65], to detect all the people in the scene (we use the official implementation, Detectron [66] pre-trained on the COCO dataset). In particular a publicly available model for scene segmentation[1] is used. This model labels every pixel in the image with one of the target labels (*i.e.*, semantic/instance segmentation), one of them being *person*. From this output, the image segments with the *person* label is keeped.

The person segment with the largest area is designated as the pilot. The minimum-size squared patch that contains the pilot segment is selected. The pixels that do not belong to the person are masked out. With this, irrelevant background information that could have a negative influence in next stages is removed.

For the direction classification part, several CNN architectures for image classification have been explored, offering MobileNetV2 [67] the best compromise between performance and delay. MobileNet is a well known efficient architecture very well suited for applications with execution time restrictions.

Skeleton. This strategy is also based on Mask R-CNN [65], but in this case a model pre-trained to estimate a person skeleton keypoints[2] is used. This model estimates the postural information from all the people in the image. In particular it provides a list of the coordinates of the following keypoints for each person found: {*nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, right ankle*}. These 17 keypoint coordinates are used to represent a person. As in the previous strategy, the person of largest area as the pilot is designated.

To recognize the pointing direction, the angle (θ) of the arm link between the elbow and the wrist is calculated, using the corresponding skeleton keypoints p_{elbow} and p_{wrist} coordinates:

$$\theta = \operatorname{atan2}\left(\frac{p^y_{elbow} - p^y_{wrist}}{p^x_{elbow} - p^x_{wrist}}\right). \tag{4.1}$$

θ is computed for both arms and the system chooses the arm that is farther from the "resting" position as the "pointing" arm. Numerous options to classify θ into one of the 8 possible pointing directions were explored. The most relevant are the following:

- **Nearest Neighbour (NN).** The median of the orientation of the arm link for each class in the training examples is computed. Given a new θ, it is assigned to the class of the closest median. This is a fairly simple process, which does not contemplate the possibility of having an *unknown* class.

[1]Set 12_2017_baselines, model e2e_mask_rcnn_R-101-FPN_2x [66]
[2]Set 12_2017_baselines, model e2e_keypoint_rcnn_R-101-FPN_1x [66]

Table 4.1: Detection results of the *Hands&Faces* YOLOv3 model on the DDIR dataset.

Result found	DDIR-1	DDIR-2
No face	133 (7.95%)	720 (22.41%)
Only face	97 (5.80%)	75 (2.33%)
Face and one hand	698 (**41.75%**)	668 (**20.8%**)
Face and two hands	744 (**44.50%**)	1749 (**54.45%**)
Total	1672	3121

- **SVM.** Standard RBF-kernel SVM classifier [68]. Different kernel functions and configurations have been evaluated and the RBF kernel obtained the best performance.

- **Decision-Tree.** Standard Decision Tree classifier [68]. Compound classifiers like Random Forests were also considered, but they converged to a single tree because of the simplicity of the input (a single number/angle).

Hands&Faces. This third strategy is based on the detection of the person's hands and face, rather than segmenting the whole body. It is inspired by [69], where they fine-tuned a YOLOv2 [70] model, with a new dataset they released, to detect hands and faces. Using their released dataset, a COCO-pretrained YOLOv3 [71] model was fine-tuned. This model outputs the position and size (*i.e.*, bounding box) of the hands and faces that appear in the image.

An evaluation of the detector obtained on both challenges presented by the hands and faces data authors [69], for hands[3] and faces[4] detection respectively. This evaluation obtain an AP of 87.7 and AR of 74.5 for the VIVA hand detection challenge and AP average of 0.36 in the WIDER face challenge. These results show our model does not reach the top performance in the face detection, however it is important to note that the challenge poses very general and heterogeneous face detection tasks, which are often far from the type of images expected in our system. Analyzing the detection results of the obtained model in our data, shown in Table 4.1, it detects at least a hand and a face (enough for our approach to work) in most of the images (75-80%).

Inspired by [69], the relative position of the hands and face is computed in our algorithm to identify the pointing direction. The chest position based on the detected face is approximated and trace a ray to the center of the hand, as an approximation to the arm pointing direction. The pointing angle is computed similarly to our previous approach:

$$\theta = \operatorname{atan2}\left(\frac{p^y_{chest} - p^y_{hand}}{p^x_{chest} - p^x_{hand}}\right). \tag{4.2}$$

[3]`http://cvrr.ucsd.edu/vivachallenge/index.php/hands/hand-detection/`

[4]`http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/WiderFace_Results.html`

θ is again computed for both hands and the one farthest from the "resting" position is kept. Once the angle of the link between a hand and the chest is computed, the classification of the pointing direction is computed in an identical manner to the previous approach.

There are significant differences between the three strategies built. The biggest advantage in the *segmentation* is that it works with the input image directly, which means that our objective of making this a natural pointing recognition system is easier since posture geometry is not necessary to define the gestures. However, since it is a more variant representation, the success of this strategy depends greatly on the heterogeneity of the training dataset to make sure it generalizes correctly. The *skeleton* and *Hands&Faces* representations are far more abstract and invariant to the person and their surroundings, and consist of much smaller descriptors, which facilitate an efficient classification. As mentioned, in these approaches the gestures need to be defined with posture geometry and they discard the visual information from the image, so any error in the skeleton information has much more effect on the results.

4.5.3 Pointing direction recognition in RGB-D

As a more general extension to the 2D pointing direction recognition task, a system that recognize 3D pointing directions was explored. The added value of this extension is evident due to the increase in maneuverability.

The subset DDIR-5 from the DDIR dataset contains contains RGB-D images of users performing 26 different 3D pointing gestures (the 3D directions considered are represented in Figure 2.7). The same 8 pointing directions than the 2D case were considered, but in three different depth planes: center (aligned with the person), front (closest to the camera) and back (furthest from the camera). To classify all the 3D directions was attempt on an end-to-end similar to the 2D case, but as detailed later in the experiments, the best option is to separate the 2D direction and the depth classification problems. This was solved with an additional classifier to identify the depth. The depth value is discretized into three possible classes: *back*, *center* and *front*, corresponding to the space in front of the user (*front*), at the same depth that the user (*center*) or the plane behind the user (*back*).

The approach used for this additional module is based on the *skeleton* approach. The skeleton keypoints are detected using the *skeleton detector* and use the x, y and depth from those points to calculate the x-, y- and z-angle of the vector that goes from the centroid of the skeleton to each keypoint.

Table 4.2: Interaction recognition accuracy.

	Point	**Show**	**Speak**
Point	**72.85%**	76.01%	55.46%
Show	12.36%	**12.88%**	20.00%
Speak	14.78%	11.11%	**24.54%**

(a) Vision-Only Classification

	Point	**Show**	**Speak**
Point	**85.71%**	22.03%	0.00
Show	14.29%	**77.97%**	0.00
Speak	0.00%	0.00%	**100.00%**

(b) Multimodal Classification

4.6 Evaluation

The evaluation of the three approaches is reported in this section. First, the evaluation of the offline approach to the Human-Robot scenario. Then, the online approach is evaluated and compared with the offline version. Last, the versions presented of Drone-Human scenario in previous section are evaluated and compared.

4.6.1 Offline human-robot interaction

In order to demonstrate the benefits of multimodal data, the interaction type is classify by using only visual data and SVM. Table 4.2(a) shows the confusion matrix.

With the speech modality using the first word of the user speech (*this/that/the*), as explained in Sec. 4.3, the model is augmented. Table 4.2(b) shows the confusion matrix obtained by this classifier, which improves the results for all classes, discriminating the *Speak* interaction and improving *Point* and *Show* from 72.85% to 85.71% and 12.88% to 77.97% respectively.

4.6.2 Online human-robot interaction

This approach presents several variations in comparison to the offline version. Since it works incrementally and it uses interaction to clarify, a direct comparison is not possible. However, a comparable error rate can be extracted at the end of the evaluation.

To evaluate our online interaction recognition, the Algorithm 2 is run for all videos in the dataset (100 *Point*, 100 *Show* and 100 *Speak*), in order to incrementally learn and classify them into the considered interaction types. The specific values for the parameters detailed in Section 4.4 that were used in all our experiments are: $n\_vids = 4$, $min\_prob = 85\%$, $n\_updates = 5$ and $min\_distance = 0.56$.

The algorithm maintains a stable accuracy for the different users. Figure 4.5 shows how the interaction recognition accuracy barely changes as we incrementally process more users. In this plot, *Error* means the output of the classifier is erroneous because it classifies a video with confidence (probability above *min_prob*) into a wrong type of interaction. *Correct* means the algorithm classifies a video with confidence into the

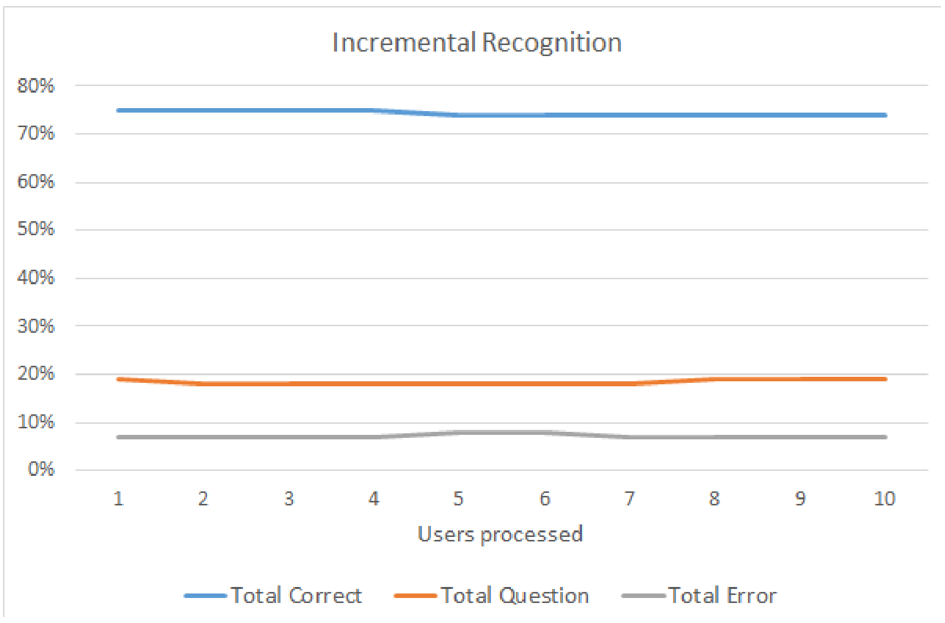


Figure 4.5: Interaction recognition results.

Table 4.3: Interaction recognition results per class (10 users).

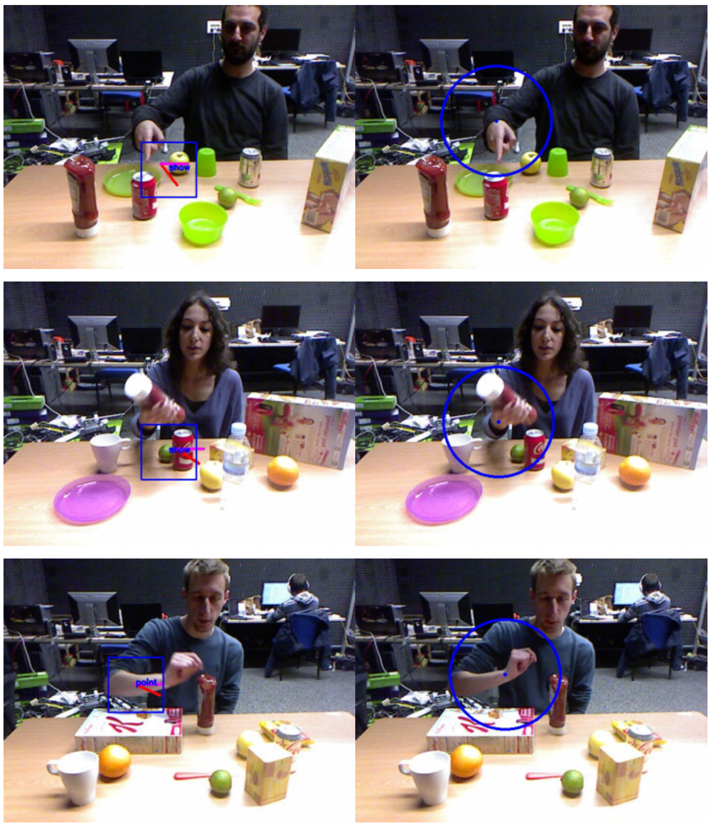
	Correct	**Question**	**Error**
Show	0,60	0,31	0,10
Point	0,62	0,27	0,11
Speak	1,00	0,00	0,00
Total	0,74	0,19	0,07

correct interaction. *Question* means the confidence of the classification output is below *min_prob* and the algorithm needs to ask the user for clarification.

Looking into the results per class, Table 4.3 shows that *Point* and *Show* achieve similar accuracy. Compared to the offline version, an improvement (13% error rate in the offline versus 7% error rate here) and more balanced results (14% and 22% error rate for *Point* and *Show* vs 11% and 10% error rates here) are obtained. The key difference between the offline and online versions is the skeleton detection. Thanks to this, the method improve from an average of 33 valid frames found per video to an average of 47. Besides, the hand patches extracted now present a higher quality, as shown in the examples in Figure 4.6. This also benefits the general performance of the pipeline because the hand patch is used in following steps.

4.6.3 Human-drone interaction

We analyze the different alternatives of our approach in the Drone-Human scenario and then analyze the performance of our best system configuration. All the experiments were run with an Intel® Core^{TM}i7-6700 CPU@3.40GHz×8, 32GB RAM and GPU



(a) Offline version (b) Online version

Figure 4.6: Interaction recognition sample results.

Geforce GTX 1070 8GB DDR5.

4.6.4 Analysis of design choices and alternatives

Per frame classification with different representations. For all these experiments, the models were trained using cross validation on the ***train-fold*** from **DDIR-1** set, validated on the ***validation-fold*** from the same set, and tested on the **DDIR-2** set for additional verification. In order to select the most promising configurations to continue with the complete system evaluation, we computed the recall for each variation, to understand the amount of frames that each method was able to identify.

Table 4.4(a) corresponds to the **segmentation representation** results. Models were trained, as previously explained, fine-tuning a MobileNetV2 model pretrained on ImageNet. Fine-tuning was run during 100 epochs, with parameter $\alpha = 1.0$ and *learning rate* set to 10^{-4}. A second version (MobileNetV2-D) has been trained using additional data augmentation to account for larger scale varieties, consisting of random image re-sizes from 1:2 to 1:0.5, which achieves better results. Table 4.4(b) shows the results obtained with variations of the **skeleton representation**. All the alternatives for this representation achieve comparable results, slightly better for the NN, also the

Table 4.4: Different representations trained on DDIR-1 train set and evaluated on different scenarios.

	Test on: **DDIR-1val**		Test on: **DDIR-2**	
	8 classes	8 classes+ "unknown"	8 classes	8 classes+ "unknown"
(a) Segmentation representation				
MobileNetV2	90.1 (8.2)	68.3 (22.3)	86.5 (12.0)	71.9 (22.9)
MobileNetV2-D	**93.0** (4.9)	**76.2** (18.6)	**90.6** (9.2)	**78.8** (19.2)
(b) Skeleton representation				
NN	93.0 (10.8)	N/A	**91.9** (6.5)	N/A
SVM	95.3 (6.6)	**76.5** (30.6)	90.4 (8.3)	**76.2** (28.1)
Decision-Tree	**95.5** (3.2)	57.8 (18.9)	89.6 (10.0)	57.2 (24.7)
(c) Hands&Faces representation				
NN	54.4 (26.3)	N/A	47.0 (24.5)	N/A
SVM	60.3 (27.0)	**46.6** (33.9)	50.9 (25.7)	**42.9** (28.0)
Decision-Tree	**61.5** (27.7)	40.9 (23.7)	**51.6** (25.5)	32.4 (22.9)
Using only images with a detected face and at least one hand				
NN	71.4 (17.6)	N/A	71.2 (25.1)	N/A
SVM	78.8 (15.4)	**58.3** (33.6)	76.4 (24.1)	**63.5** (31.4)
Decision-Tree	**80.4** (16.0)	51.5 (21.4)	**77.6** (23.6)	46.1 (22.4)

simplest to implement.

Table 4.4(c) shows results for the pointing direction task with our *Hands&Faces* representation strategy. The core component of the **Hands&Faces representation** is the hands and faces detector detailed in section 4.5.2. Under the same conditions as the other approaches, the NN and the Decision Tree results are very similar, and in both cases significantly lower than the other strategies. As expected, part of this is due to errors in the hands and face detection. If test images where at least a hand and a face are detected are the only ones considered, the results are significantly better but still lower than the other approaches as shown in the same table.

Discussion. Including an *unknown* class, corresponding to images with non-pointing gesture, drops the performance of our system significantly (around 10% difference between the columns "8 classes" and "8 classes + *unknown*" in all configurations). Therefore, training was done only for the 8 direction classes and a different strategy to account for robustness to ambiguous actions (*i.e.*, non pointing). The described small temporal consensus stage that filters the classification results in section 4.5.1 was included.

The results from the Hands&Faces strategy are far from the results from the other two strategies, regarding accuracy and robustness, so this option was discarded for further analysis in the following experiments. Note that the skeleton representation is

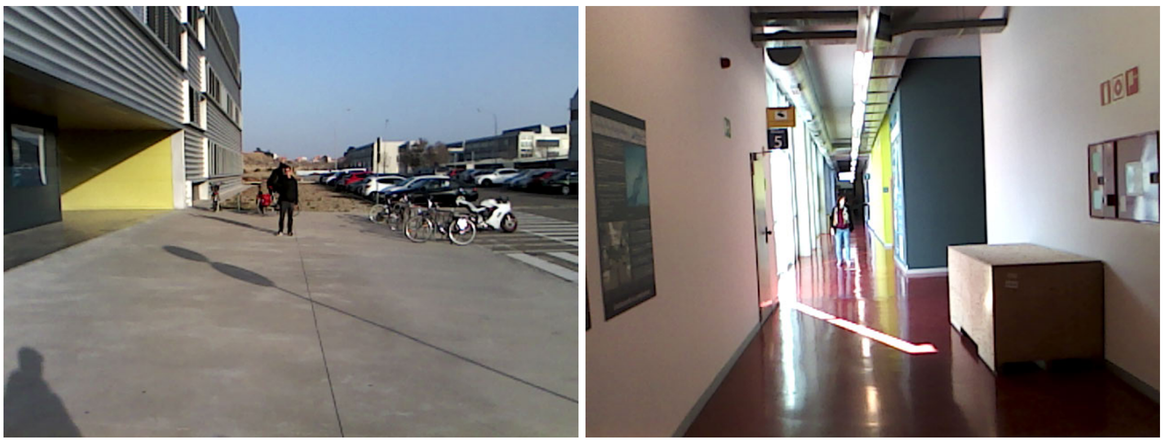


Figure 4.7: Example of a limitation of our system. For distances larger than 6 meters, our pipeline fails to identify the direction due to the low resolution of the user.

Table 4.5: Models trained on DDIR-3&4 and tested on DDIR-3&4val which contain data acquired at different distances.

Skeleton-NN	Test at 5m	Test at 10m
Trained at 5m	83.3 (7.6)	70.8 (12.7)
Trained at 5m&10m	83.2 (7.8)	70.7 (12.8)
Segmentation-MobileNetV2-D		
Trained at 5m	89.0 (4.7)	70.9 (6.1)
Trained at 5m&10m	88.4 (6.5)	70.6 (7.6)

very compact, which is convenient for efficiency but it may lose useful information such as the appearance. Both the *Skeleton* and *Segmentation* results (in Table 4.4(a) and (b) val/test columns) show that when the test data is from a domain further to the training one, the average results remain almost intact demonstrating good generalization of our models.

Robustness to various camera distances. The two best configurations (Skeleton-NN and Segmentation-MobileNetV2-D) were further evaluated for robustness, on a similar experiment that the one shown in Table 4.4 but this time training on the DDIR-3&4 training sets and evaluated on DDIR-3&4val, as shown in Table 4.5. This experiment shows that our system performs best when the user is between 2 and 6 metres from the camera. At larger distances the person is imaged at an extremely low resolution in the cameras we used to record the datasets (see Figure 4.7 for two examples). The data augmentation done on the training sets is enough for a model to reach the same accuracy at long distances (more than 6 metres) than models trained directly with data recorded at those distances, as shown in Table 4.5. These results point that the segmentation approach is more robust to scale changes due to different distances to the camera, therefore it is the most suitable strategy for the system.

4.6.5 Video classification system

The following experiments evaluate in more detail the best configuration of our system. The per frame classification is combined with a more robust consensus strategy for the final video classification and different aspects of interest are discussed to demonstrate the applicability of this approach.

Consensus strategy benefits. As expected, the complete approach including the consensus stage ("Consensus improvement") obtains better results than "Per frame" classifications. Table 4.6 shows the results of a more detailed evaluation in a more challenging setup than the preliminary evaluations from previous subsection. Models were trained on DDIR-1&2 data, recorded from a base-station camera, and evaluated on DDIR-3 and DDIR-4, where DDIR-4 was recorded from an on-board drone camera. First note the consensus improves around 12% for DDIR-3 and around 10% for the most challenging test of DDIR-4. This experiment results also show that changes in perspective or camera type do not affect the good performance of the system, showing good generalization.

Robustness to user, scenario and camera variations. Besides the robustness to changes in camera type and perspective, the system presents good invariance to all the relevant changes considered with the presented dataset. Robustness to user variations can be analyzed in all experiments since different users appear in all datasets. Even multiple users appearing on the background of several scenes is not an issue for the system, as long as the user providing the command is the closest to the camera (as assumed by the system). It is also relevant to note that training in one environment (DDIR-1&2) and evaluating in a completely different one (DDIR-3 or DDIR-4) provides very good results, see table 4.6. This demonstrates the good generalization of the model learned to different scenarios.

System performance. The final implementation runs the Mask R-CNN for *Segmentation* on the GPU, while simultaneously the *Pointing direction classifier* is run, using MobileNetV2-D, on the CPU.

The processing time of one image, running the complete system, is an average of 225ms (the detector takes more than 90% of the time). This means that the proposed final system can run at 4.5 fps. The best compromise between usability and accuracy was found using a 5-frame window for the consensus. Longer windows can improve the performance but require the gesture to be performed for longer and it becomes less natural for the user.

Table 4.6: Segmentation strategy trained on DDIR-1&2 and evaluated on DDIR-3 and DDIR-4. Precision-Recall running independent *Per frame* classification (PF) vs applying Consensus (C).

	Per Frame (+ Consensus improvement)			
	DDIR-3		DDIR-4	
Class	Precision	Recall	Precision	Recall
up	63.2(+13.1)	56.1(+10.9)	89.1(+6.4)	34.6(+10.2)
up-right	82.3(+8.3)	65.2(+7.3)	85.7(+6.0)	83.4(+8.7)
right	85.8(+5.8)	63.5(+9.5)	93.0(+2.8)	69.4(+11.4)
down-right	81.7(+10.4)	70.9(+13.6)	69.4(+13.1)	88.6(+4.9)
down	45.3(+14.8)	87.6(+6.0)	68.7(+10.3)	77.8(+11.7)
down-left	83.4(+8.8)	78.6(+9.8)	76.4(+12.4)	87.4(+8.6)
left	79.9(+10.1)	72.7(+11.8)	66.7(+14.3)	76.5(+10.7)
up-left	67.5(+13.8)	70.9(+10.9)	66.6(+20.2)	76.2(+12.8)
Avg PF	73.6	70.7	77.0	74.2
Avg C	**85.3**	**83.2**	**87.7**	**84.1**

In hopes of exploring the use of this system in a drone without a base station, the system was installed and measured on a Jetson AGX Xavier. The system takes 1638ms on average to process one image, which means it could run in it independently at 0.61 fps.

The most significant limitations of the current system are the following. As the whole pipeline runs at 4.5 fps and the consensus system requires 5 images to decide, the user has to keep pointing for at least 1 second. While it is a reasonable time, it also means that quick gestures are not recognized by our pipeline. Besides, the performance decreases with distances larger than 10 meters of the user to the camera. This means that the set up has two possibilities: either the camera is on board the robot with the robot not further than 10 meters from the pilot, or the camera should be placed on a base station near the pilot.

Chapter 5

Object detection from HRI

5.1 Introduction

In order to identify the relevant visual information for the object model to be learned, it is essential to use the interaction type classification that we described in the previous section. This classification allows us to build different strategies to find the regions of interest (RoI), as summarized in Figure 5.1. In particular, three different strategies are proposed, one for each type of interaction:

- *Show interaction.* As described in the previous section, the hand position was estimated to recognize the interaction type. By using such position, filtering it by height and segmenting it, we can obtain a patch containing the object.
- *Point interaction.* A candidate object segmentation can be obtained using a combination of algorithms. The final candidate is selected by using the pointing direction of the hand patch.
- *Speak interaction.* Following the same candidate segmentation, the objects are pre-recognized by a learning method. Then, the user speech gives us the anchor object, the target object and the direction.

In this chapter, we present the three approaches and the evaluation of its performance. We evaluate this module without taking into account previous steps. An advantage of this approach is that if different interactions appear, they could be included without deteriorating the others.

5.2 Related work

State-of-the-art methods in object detection are mostly based on deep learning. One main differentiation between methods is the number of stages to process the images. Many works, like YOLO [72], SSD [73], Retinanet [74], FSAF [75] and NAS-FPN [76], use a one stage algorithm to process the image and obtain the object boxes and their classes. Other works uses two stages, like Faster R-CNN [77], FPN [78], Mask R-CNN [65], Cascade R-CNN [79] and Li-bra R-CNN [12], to obtain class-agnostic proposals and class-specific detections. Our algorithm uses Mask R-CNN to obtain candidates in combination with a classic method that give us the best amount of candidates.

There are few works that join human robot interaction to guide the object detection. Canal et al. [80] presents a framework that is similar to our approach in speech but using a conversation to obtain feedback from the users. In our case, the framework doesn't need to know the target object class before the interaction.

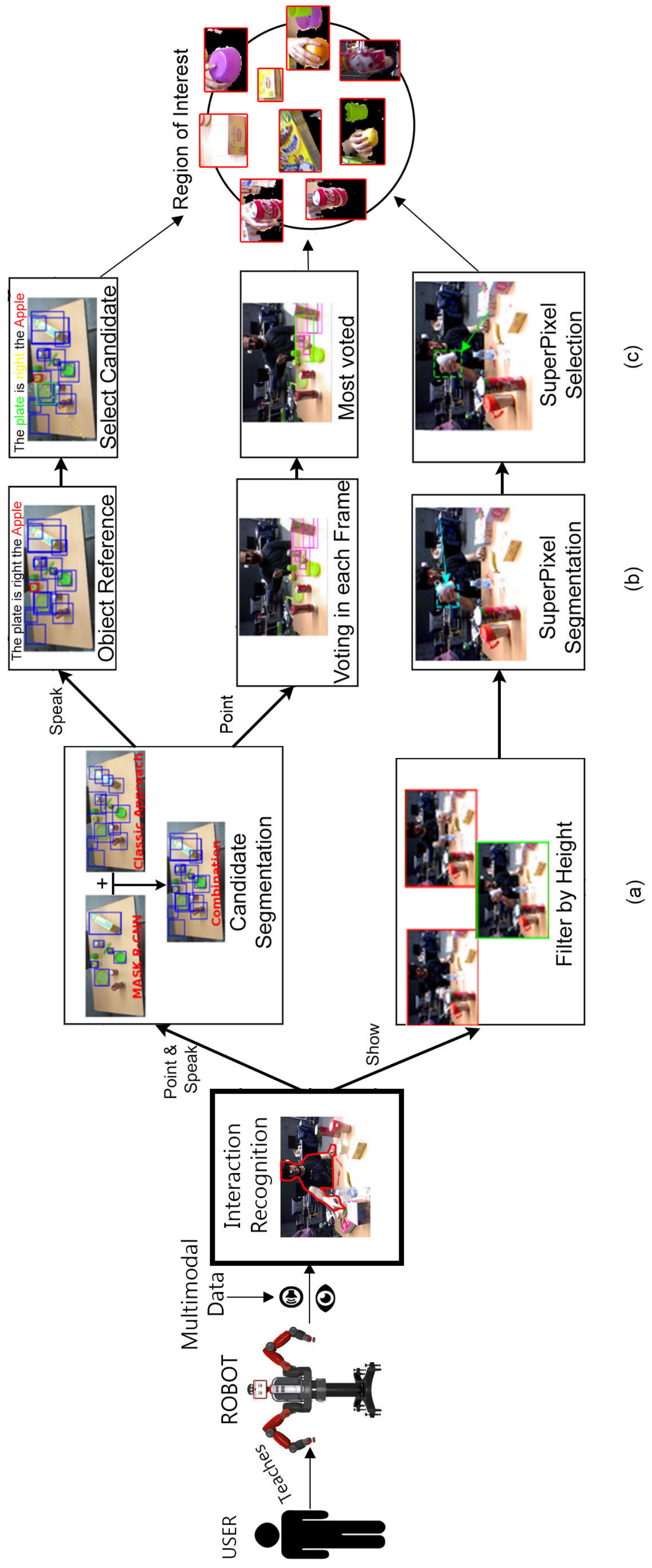


Figure 5.1: Region of Interest Extraction from Human Robot Interaction. We use the speech and the visual data to recognize the type of interaction. Each pipeline is divided in three steps: a) Initial information: Pre-process step to discard useless information or obtain possible candidates, b) Reference step: Obtaining the reference patch depending on the interaction and c) RoI extraction: Extraction of the Region of Interest based on the Reference step.

Sadi et al. [81] presents a system able to recognize gestures usually employed in human non-verbal communication. Similar to us, they use the pointing location estimation to obtain the candidate object.

There is a large literature on object proposal methods that do not use deep learning. Widely used object proposal methods include those based on grouping super-pixels, e.g., Selective Search [82], CPMC [83], MCG [84], and those based on sliding windows, e.g., objectness in windows [85], or EdgeBoxes [86].

5.3 Show interaction

This strategy considers when the user grabs the object and lifts it, bringing it closer to the robot cameras. The key steps are discussed next and detailed in Algorithm 3.

Selecting the best frame to extract the hand patch. This usually happens when the hand is at a high position, as occlusions are less likely at that moment. Therefore, we select the subset of frames where the hand is above 70% of the highest vertical hand position along the clip.

Selecting image regions most likely to contain the object. Each image is segmented using SLIC [87] superpixels. Our algorithm selects superpixels likely to contain relevant information, i.e., having a large overlap with the *hand* patch and a small distance between the superpixel and the *hand* patch centers.

Algorithm 3 Target object detection for *Show* interaction.

```
 1: min_instersected = 40%
 2: max_far = 200
 3: function OBJECT_DETECTION_SHOW(Video_RGB-D, interaction, Hand_Pos)
 4:     Patches = []
 5:     for each Frame_RGB in Video_RGB-D do
 6:         if (Hand_pos-min_height) > 0.7* (max_height-min_height) then
 7:             SuperPixels = Slic(Frame.Front) Correct_SuperPixels = []
 8:             for each SuperPixel in SuperPixels do
 9:                 intersection = get_intersection(Hand_Pos,SuperPixel)
10:                 distance_center = distance(SuperPixel.center(),Hand_Pos)
11:                 if intersection >= min_intersected &&
                    distance_center < max_far then
12:                     Correct_SuperPixels.add(SuperPixel)
13:                 end if
14:             end for
15:             Patch = Extract_patch(Correct_SuperPixels)
16:             Patches.add(Patch)
17:         end if
18:     end for
19:     return Patches
20: end function
```

5.4 Point interactions

This strategy considers when the user is pointing to an object. Differently to *Show*, where the object is easy to find because it is grasped by the user, *Point* interactions are more challenging. The main difficulties are the estimation of the pointing direction and the selection of the candidate object region from the potential candidates along such direction. The key steps of this strategy are described next and detailed in Algorithm 5.

Candidate object segmentation. This segmentation is run on the first frames acquired from the *Top* camera, before the user motion starts. The *Top* camera views facilitate better object pre-segmentation because they have less clutter and occlusions than *Frontal* camera views. We can map approximately the objects from one view into another (in this case, from *Top* to *Frontal* views) using the table plane homography.

To obtain the candidate segments, our algorithm runs two different but complementary approaches on the resulting image. In the first approach, Mask-RCNN [88] is used to segment a few candidate objects. This CNN model can reliably segment certain objects but, since our scene contains significant occlusions and small objects (see examples in Figure 5.4), it misses important candidates. In the second approach, a superpixel segmentation [89] is used to remove table pixels. Then we apply Otsu's thresholding with the Watershed algorithm (as described in Meyer et al. [90]), to obtain object candidates. From this candidates, we remove object that are too small or too large, objects that occupy more than one third of the table or less than an area of 100 pixels. This process is detailed in Algorithm 4.

Hand pointing direction estimation. Figure 5.2 shows several examples of the output of our pointing direction estimation algorithm. Hand contours are extracted using a Canny edge detector on the depth image. Then, we draw lines from the hand center at several equally distributed angles. The pointing direction is approximated by the line that intersects with the hand boundary at the furthest distance to the hand center.

Intersection between the pointing direction and candidate object segments. These intersections are obtained as represented in Figure 5.1(b). To evaluate which candidates are more promising, our algorithm computes the following score:

$$Score = \sum_{Frames} \frac{Intersect(Hand\_direction, Candidate)}{Diagonal(Candidate)}, \tag{5.1}$$

where *Intersect* computes the length of the intersection between the pointing direction and the candidate bounding box; and *Diagonal* computes the length of the bounding box diagonal.

This score helps us in normalizing by the size of each candidate. The candidate with

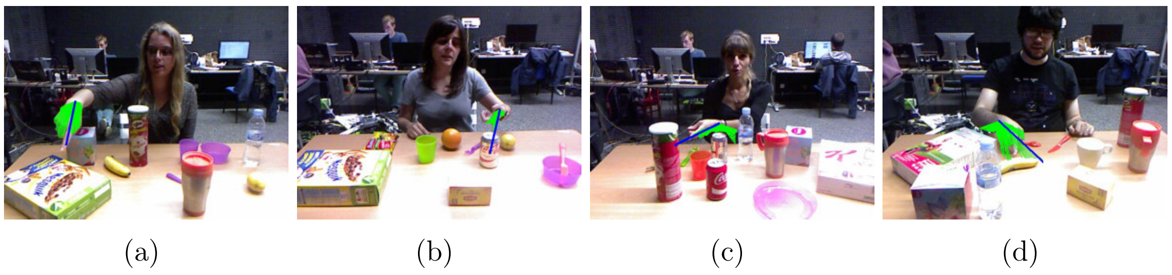


Figure 5.2: Examples for hand detection and pointing direction estimation. The green regions show equally-distributed possible directions. The blue line is the estimated direction. (a) to (c) show common correct cases; (b) shows an example where the direction is correct even for an uncommon hand pose. (d) shows a failure case, the direction is incorrect due to similar depths in the hand and the table. (best viewed in color)

Table 5.1: Examples of speech processing.

Step	**Example 1**	**Example 2**
Phrase	The apple is on front of the Coke	Cereal Box is at the right of the Mug
Nouns Extracted	['apple','front','Coke']	['Cereal','Box','right','Mug']
Target Obj.	apple	Cereal Box
Direction	front	right
Ref. Obj.	Coke	Mug

the highest score is selected, and the corresponding image patch from both cameras is extracted and used as a training sample for the incremental model.

5.5 Speak interactions

This type of interaction presents relevant challenges. Since the visual part of the action is irrelevant, we parse the user speech to extract the relevant information for the candidate patch search. We assume simple user sentences, for which standard speech processing tools like Nltk [91] can extract the target object name, reference objects and their relative positions. Table 5.1 shows two examples of the speech processed. Algorithm 6 describes our strategy for this interaction type and the main ideas are discussed next.

Object recognition on all candidate objects segmented at the top view. This recognition is run with the models available at that time. If the robot recognizes any of the objects used as reference in the description, such object is used in combination to the relative pose information to estimate a search direction.

Target area definition and candidate selection. We define the target area

Algorithm 4 Candidate object segmentation.

```
1: $min\_circle = 20$
2: $max\_circle = 60$
3: function CALCULATE_CANDIDATES(Video_RGB-D)
4:     Homography =
       get_table_homography(Frame_RGB.Front,Frame_RGB.Top)
5:     Candidates = []
6:     for each Frame_RGB in Video_RGB-D[0:5] do
7:         Plane = Calculate_plane(Frame_RGB.Top)
8:         Table_cropped = Extract_table(Frame_RGB.top,Plane)
9:         DL_Candidates = MaskRCNN(Table_cropped)
10:        Candidates.add(DL_Candidates)
11:        SuperPixels = felzenszwalb(Table_cropped)
12:        SuperPixels.filter_biggest()
13:        Filtered_Image = Otsu_threshold(SuperPixels)
14:        Heat_map = Distance_zeropix(Filtered_Image)
15:        Segments = Watershed(Peaks(Distance_map))
16:        for each Segment in Segments do
17:            Center,Radio = min_enclosing_circle(segment)
18:            if $min\_circle >= Radio >= max\_circle$ then
19:                Candidate = Extract_candidate(Segment)
20:                Candidates.add(Candidate)
21:            end if
22:        end for
23:    end for
24:    for each Candidate in Candidates do
25:        if Candidate.area < 100 or
           Candidate.area > 1/3*Plane.area then
26:            Candidates.erase(Candidate)
27:        end if
28:    end for
29:    return Candidates
30: end function
```

Algorithm 5 Target object detection for *Point* interaction.

```
1: function OBJECT_DETECTION_POINT(Video_RGB-D, interaction, Hand_Pos)
2:     Patches = []
3:     Candidates = Calculate_Candidates(Video_RGB)                 ▷ See Alg. 4
4:     for each Frame_RGB in Video_RGB-D do
5:         hand_direction =
           get_hand_direction(Frame_RGB.Front,Hand_Pos)
6:         for each Candidate in Candidates do
7:             $Score$,Intersected =
               intersect(Candidate,hand_direction)/Diagonal(Candidate)
8:             if Intersected then
9:                 Candidate.update_score($Score$)
10:            end if
11:        end for
12:    end for
13:    Selected_Candidate = Candidates.get_best_score()
14:    Patches =
       [Selected_Candidate.Front_patch,Selected_Candidate.Top_patch]
15:    return Patches
16: end function
```

Algorithm 6 Target object detection for *Speak* interaction.

```
 1: $min\_confidence = 60\%$
 2: function OBJECT_DETECTION_SPEAK(Video_RGB-D, Incremental, speech)
 3:     Patches = []
 4:     Candidates = Calculate_Candidates(Video_RGB)                ▷ See Alg. 4
 5:     Ref_label = speech.get_reference()
 6:     Reference = $None$
 7:     Ref_conf = 0
 8:     for each Candidate in Candidates do
 9:         Candidate.Label,confidence = Incremental.Test(Candidate.patch)
10:         if Candidate.Label = Ref_label &&
            confidence >= max($min\_confidence$,Ref_conf) then
11:             Reference = Candidate
12:             Ref_Conf = confidence
13:         end if
14:     end for
15:     Direction = speech.get_direction()
16:     Target_area = obtain_area(Reference,Direction)
17:     Selected_Candidate = $None$
18:     Selected_Distance = $inf$
19:     for each Candidate in Candidates do
20:         if Inside(Candidate,Target_area) then
21:             Distance = Calculate_distance(Candidate,Reference)
22:             if Distance < Selected_Distance then
23:                 Selected_Candidate = Candidate
24:                 Selected_Distance = Distance
25:             end if
26:         end if
27:     end for
28:     Patches =
         [Selected_Candidate.Front_patch,Selected_Candidate.Top_patch]
29:     return Patches
30: end function
```

using the corners of the reference patch, the corner of the images and the search direction. The selected candidate is the closest to the reference patch within the target area, similar to the *Point* interaction.

5.6 Evaluation

To evaluate the quality of the object patches, segmented by the target object detection module, we manually select which ones are a *Correct Patch*, i.e., it actually contains the correct target object. Figure 5.3 shows several correct and incorrect examples of the target object segmentation and Table 5.2 presents the quantitative results.

Point and *Speak* present lower accuracy than *Show*, as they are more challenging interactions. Both use similar strategies to segment the candidates, and they face similar challenges (small objects, large occlusions and clutter, the object potentially being anywhere in the scene). We study two different approaches to find candidate patches for the target object in *Point* and *Speak* videos: a deep learning (DL)-based

Table 5.2: Target Object patches accuracy (Acc.).

	Acc.	Total Patches	Correct Patch	Incorrect Patch
Point	**46.66** %	90	42	48
Show	**86.23** %	3210	2768	442
Speak	**47.32** %	112	53	59

approach and a superpixel (SPX)-based one, both explained in Sec 5.4. Figure 5.4 shows that DL is less robust for smaller objects, but more accurate. SPX extracts more candidates but it is less accurate. Combining both (DL+SPX), we outperform their weaknesses and obtain a better set of candidates.

Each interaction has additional challenges added to the segmentation. Most of the errors in *Point* videos are caused by incorrect pointing directions, which is a non-trivial task, as illustrated in Figure 5.5(a). Most of the errors in *Speak* videos are caused by failures recognizing the reference object. As the accuracy of the classifier improves, the quality of the *Speak* patches also improves, because the reference object detection is more reliable.

As we can see for example in Figure 6.2(b), there are significantly more target patches obtained from *Show* videos than from the rest. This is because of the strategy followed to obtain them. Since the user is moving the hand, several frames are kept,

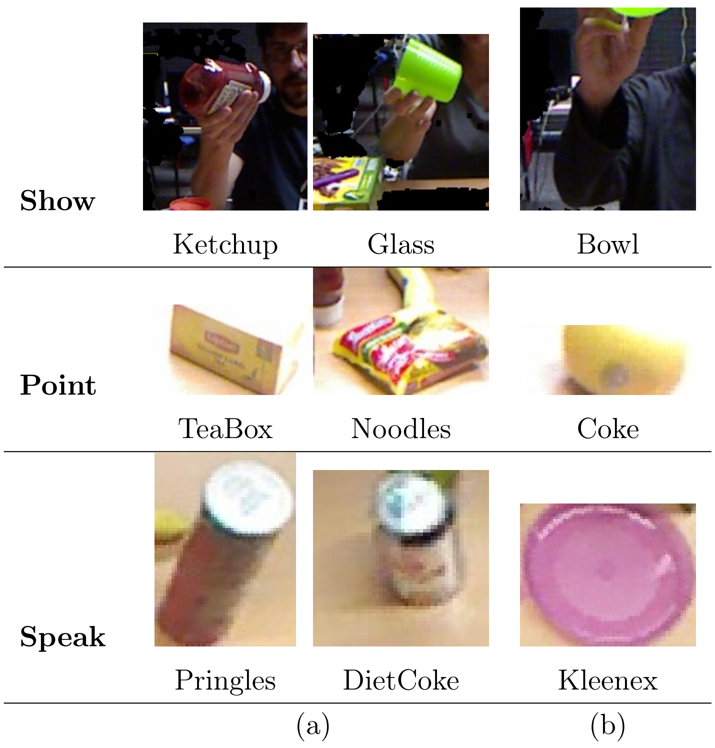


Figure 5.3: Target Object Detection sample results: (a) Correct segmentation; (b) Incorrect segmentation

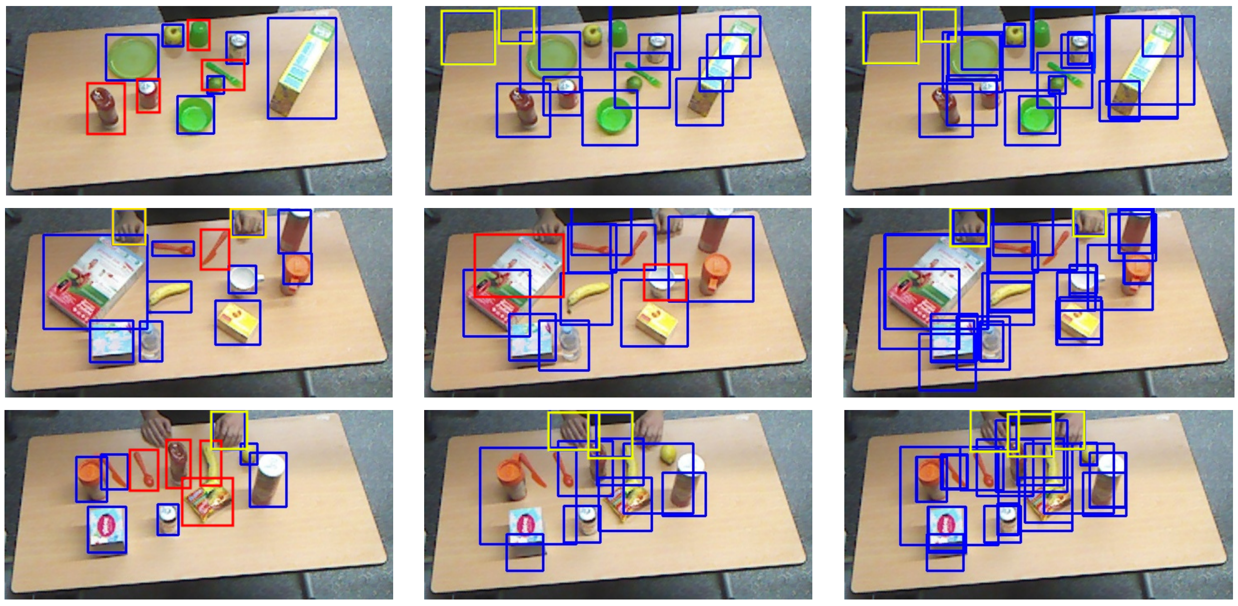
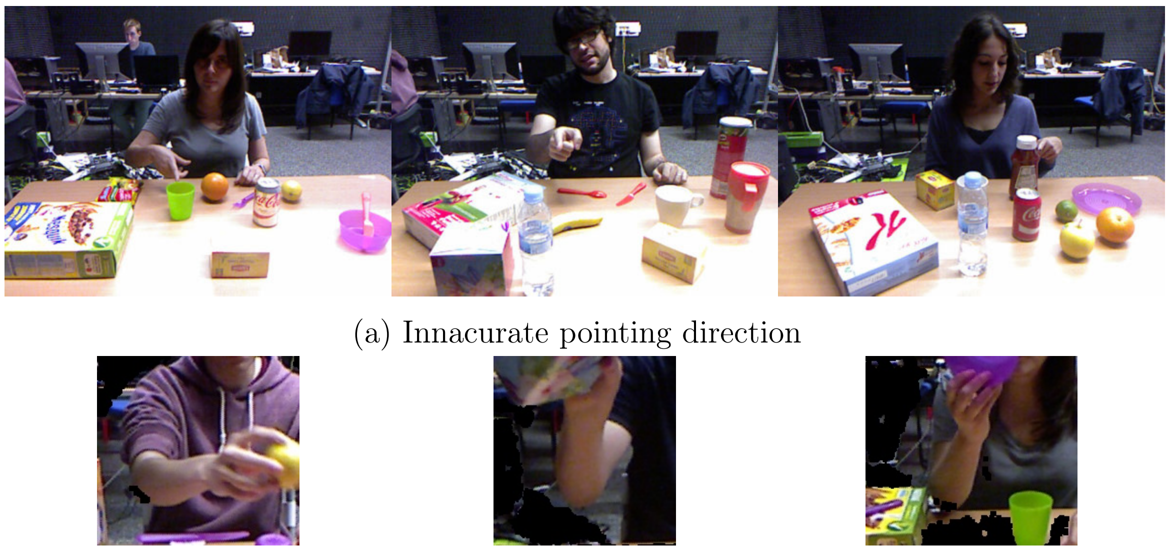

(a) DL [88] (b) SPX (c) DL + SPX (Ours)

Figure 5.4: Candidate object patches (blue rectangular regions) with the three options (DL, SPX, DL+SPX) considered. Yellow stands for false positives, and red for false negatives. For our goals, it is essential to minimize false negatives.


(a) Innacurate pointing direction

(b) Segmented patch not centered on the object.

Figure 5.5: Examples of two common difficult/failure cases for Target Object Segmentation in *Point* and *Show* videos.

because they potentially show different points of view of the target object. They are very likely to contain the target object but often not centered or fully visible (see Figure 5.5(b)), providing very noisy training data.

Chapter 6

Incremental object learning

6.1 Introduction

In this chapter, the algorithm presented in Chapter 3 is used for object learning evaluated with different descriptors. The evaluation is done using two different datasets, with a performance comparison of the descriptors.

6.2 Related work

In recent years, significant advances have been made in the field of incremental learning, many of them applying deep learning techniques. *Open-class* approaches, i.e., those able to add new categories as the data comes, are particularly relevant for our work.

In Li et al. [35], the authors create and train new classification layers as new classes are added and fine-tune the rest of the network to maintain the outputs for older classification layers. Following this work, Rannen et al. [36] uses a similar setting, one shared model and several classification layers, but at training time they add one feature-autoencoder per class. The new autoencoder is trained on the data for the assigned class and use a loss to keep the build-up error on the old ones. More similar to our work, Rebuffi et al. [37] use deep learning to obtain image representations that can be incrementally updated. They set a limit in the total number of stored examples and classify using the average feature of each class examples. With our approach, we outperform this work in the Core50 dataset and possibly at a lower cost, as we do not fine-tune the network.

Other works with deep learning focus on incrementally bind multimodal attributes to the objects like Xing et al. [38]. Here, the Perception Coordination Network online adquires and bind multimodal concepts between different sensory modules. It uses two levels of neurons inspired by the brain structure and separates lower neurons depending on the modality of the input. Our work only focus on the use of deep learning as feature extractor, because training a network online has more computational cost and the binding of concepts is out of our scope.

Other classic approaches for incremental or online learning are found in the literature. Passive-Aggressive algorithms [39] use an offline learning algorithm as a base and incrementally modify their parameters. [40] presents a variation of SVM that is able to change the support vector online. We can also find online variations or combinations of K-means clustering algorithms. Murty et al. [41] presents an approach that combines the k-means algorithm with multilevel representation of the clusters. Likas et al. [42] presents a global k-means that adds a new cluster at a time and dynamically updates the other clusters by applying the k-means algorithm multiple times. More recently, Mensink et al. [43] presents an incremental Nearest Mean Classifier which uses nearest neighbor with the mean of each class for classification and also for generalization.

Other group of approaches apply a data transformation based on self-organizing maps (SOM) Neural Networks, as a base to incrementally update the nodes in the Network. For example, Furao et al. [44] presents an online unsupervised system with an incremental update of a Neural Network based on SOM (SOINN). Xing et al. [45] presents a more recent variant of the Self-Organizing Incremental Neural Networks that incrementally transforms the nodes in the layers of the SOINN using the local distribution. Gepperth et al. [46] uses SOM to reduce the dimensionality of the data in the Hidden Layer, but it needs to keep all the data in memory for re-training.

These approaches work with seeds for each class based on the existing data and can not add new classes over time. Differently, our goal is to be able to learn completely from scratch and increase incrementally the number of classes.

Incremental learning is a paradigm very suitable for robotics, where the data typically arrives sequentially, and the robot needs to keep the best model up to date at real time, such as mapping in [47] or inverse dynamics incremental learning in [48]. The same way, Angeli et al. [49] presents an incremental method to build a model to recognize visual loop-closures. We find multiple examples that propose how to incrementally adapt environment visual models as the robot moves. These approaches are often based on Gaussian Mixture Models that can be easily updated and maintained to recognize regions of interest for the robot [50, 51]. In robotics, we find situations where the robot interacts directly with the scene, e.g., grasping and moving an object, to build an incremental object model [52, 53, 54]. Our approach is complementary to these works, as we focus on the human interaction. This interaction is needed in real scenarios, e.g., if the object to be learned or explored is out of reach of the robot.

6.3 Descriptors definition

We apply our incremental learning approach to learn object models from image patches. For an illustration of the typical patches we have available in the considered robotic settings, Figure 6.1 shows a few examples of the MHRI dataset. In robotic settings computation capability is typically limited. Therefore, we evaluate the following descriptors, that are reasonably small and fast to compute.

BoW histogram This descriptor consists of a standard Bag of Words (BoW) representation over local image features. We use ORB features [92], as they provide a good compromise between accuracy, efficiency and number of keypoints. The BoW

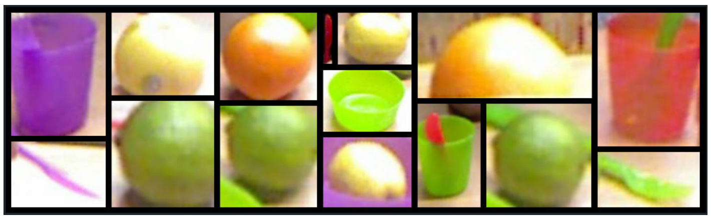


Figure 6.1: Sample objects patches from human-robot interaction. The object views are typically low-resolution patches where standard keypoints/descriptors give low performance.

descriptor is a histogram of the occurrence of different visual words from a vocabulary. The vocabulary is obtained by clustering all the features extracted on a large set of images. We build the vocabulary using the Washington dataset [27] to avoid using the same data of the online experiments. We use 1000 visual words, clustered from more than 2 million features extracted from over 12000 images. The images contain close-up views of from the categories in the dataset, and scene views containing the objects and clutter.

To obtain the descriptor BOW_{ORB} of an image *patch* we first extract ORB features, find the closest word to each of them and build BoW as a 1000-bin histogram of the frequency of occurrence t_w of each word in the image as:

$$BoW_{ORB} = [t_1, ..., t_w, ...t_{1000}] \quad ; \quad t_w = \frac{n_{\mathrm{wp}}}{n_k} \ , \tag{6.1}$$

where n_{wp} is the number of occurrences of word w in image *patch* and n_k is the total number of keypoints in image *patch*.

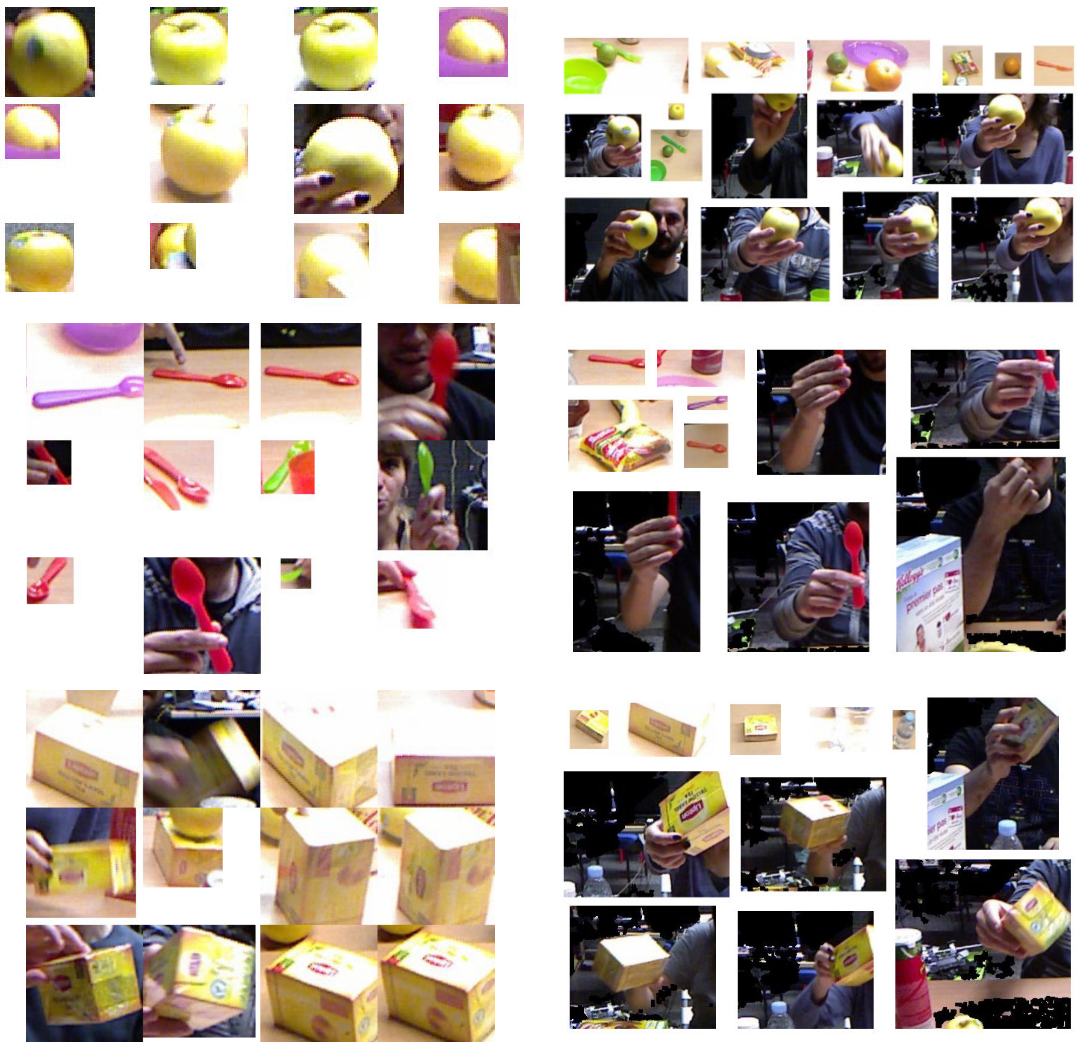
Color Histogram This descriptor approximates the color distribution in an object view. We compute three normalized 8-bin histograms (H_r H_g H_b), one per color channel, over region pixel values:

$$HC_{RGB} = [H_r \ H_g \ H_b]. \tag{6.2}$$

SIFT keypoints We obtain SIFT keypoints and their associated descriptors [93] as

$$SIFT = \{s_1, s_2, s_3, ..., s_n\}, \tag{6.3}$$

where $SIFT$ is the set of n keypoints obtained in the object view. Although it has higher computational cost than other local features, SIFT is an accurate and robust local feature appropriate as a baseline.



(a) (b)

Figure 6.2: Examples of (a) *Manually Cropped* and (b) *Automatically segmented* patches from three objects in the *MHRI* data.

CNN features We use the flattened output of the last *GAP* (Global average pool) layer from ResNet50 [4].

$$ResNet50 = ResNet50(patch).GAP \tag{6.4}$$

The experimental validation of this module, in the next section, shows that the best performing descriptor for our application is the CNN-based one. It is also the largest descriptor considered. The Color Histogram HC_{RGB} performs similarly in the evaluation of this particular module (see Sec. 6.4), but its performance decreases when evaluating the whole pipeline (see Chapter. 7).

6.4 Evaluation

This section analyzes the performance of the incremental learning algorithm, explained in Chapter 3, using the visual descriptors detailed in this Chapter. We use both the *Core50* and *MHRI* datasets. In *Core50*, there are 11 sessions, 8 for training and 3 for testing, and 50 objects. We use the same setting used in the *New Classes and Instances* experiment in [34]. In *MHRI*, there are 670 manually cropped patches from 22 classes, approximately 30 patches per class and 67 patches per user. Each experiment consists of a 10-fold cross validation, each fold keeping all the data from one user for test and using the rest of the users for training.

We analyze the influence of the main parameters of our strategy (object patch descriptor and incremental model size) and compare the performance of the proposed method with standard baselines for offline object recognition. To decouple this evaluation from data quality, we first evaluate the incremental model using **manually segmented** object patches (see Figure 6.2(a) for examples of such patches).

Object patch descriptors. We evaluate several **patch descriptors**, as detailed in Sec. 6.3: Color Histogram (HC_{RGB}), Bag of Words using ORB descriptors (BoW_{ORB}), the output of layer GAP of pre-trained ResNet [4] ($ResNet50$), and SIFT correspondences ($SIFT$).

We first run this experiment on *Core50*, comparing HC_{RGB} and $ResNet50$ (keypoint-based descriptors are expected to have worse performance). We varied k, the model size per class, between 10 to 200 (the latest uses all the data). In Figure 6.3 we can see the average accuracy for the different limits per class and descriptors. As we can see, a model size of 80 samples per class, $Resnet50$ and the cosine distance obtains an accuracy of 31.97%, outperforming the 29.56% [1] reported in [34] (We do not compare against their cumulative version since it uses all the data to fine-tune the network). Per-class limits of 80 and 100 for HC_{RGB} descriptors and Battacharya distance obtains 31.29% and 31.97%, respectively. In Figure 6.4, we can see the accuracy evolution as more samples are used for training. This figure also shows that HC_{RGB} performs better than $ResNet50$

We run a similar experiment on *MHRI*, where we additionally consider the keypoint-based descriptors ($SIFT$ and BoW_{ORB}). We use the Bhattacharyya distance for HC_{RGB} and BoW_{ORB} and the cosine distance for $ResNet50$. SIFT points are matched using FLANN [94], left-right consistency and Lowe's nearest neighbour ratio test [93].

[1] `https://vlomonaco.github.io/core50/leaderboard#keywords3`

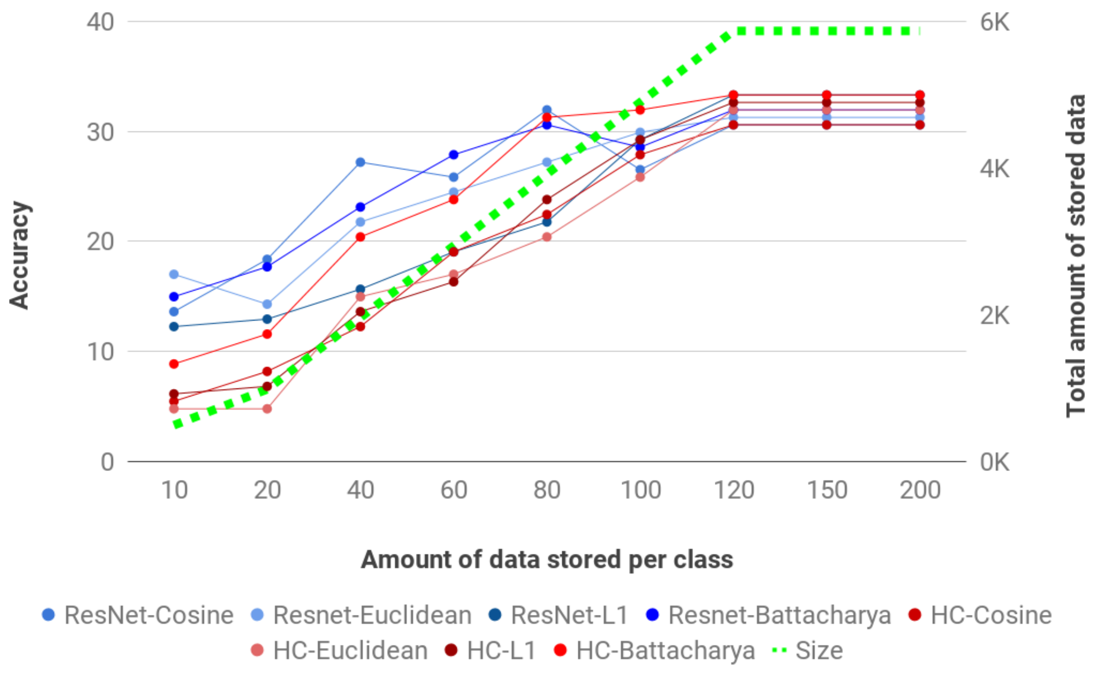


Figure 6.3: *Core50* experiments processing all data.

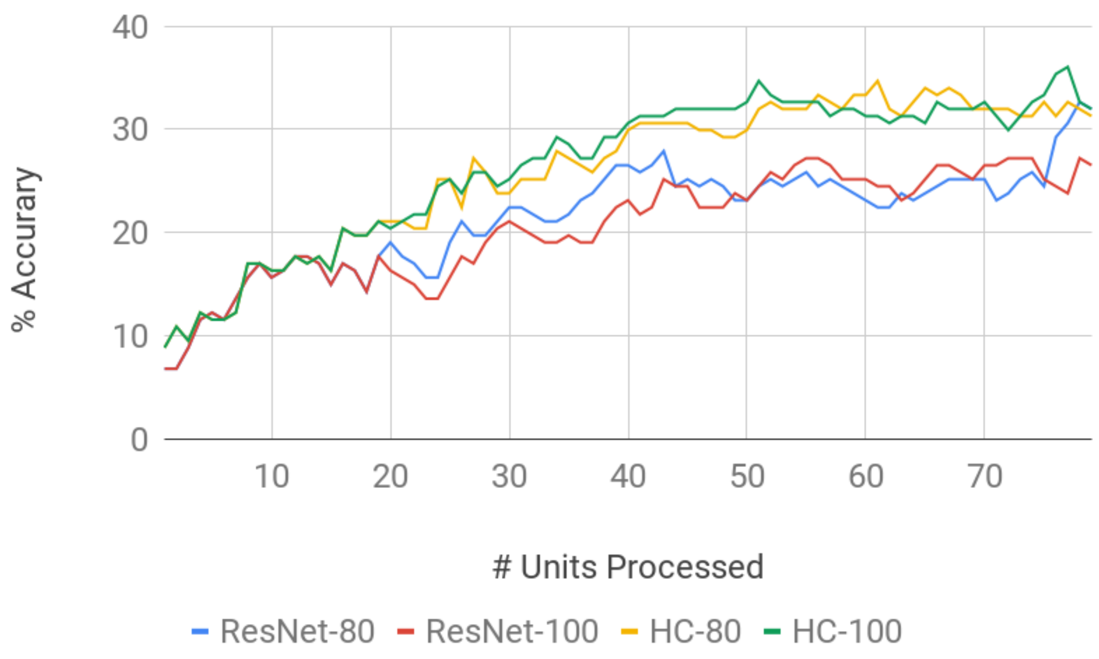


Figure 6.4: *Core50* experiments, incremental results.

Since *MHRI* is around five times smaller than *Core50*, we configured the size-limit to $10, 20, 30$, and all data. Table 6.1(a) shows the object recognition accuracy obtained with all the descriptors, and different model size limits, using Manually Cropped patches. HC_{RGB} and $ResNet50$ have the highest accuracy. This can be explained by looking at the examples in Figure 6.2. Notice that our objects have distinctive colors and poor texture, and hence descriptors based on keypoints will perform poorly.

The best results are obtained for model size 20 for HC_{RGB} and 10 for $ResNet50$. In both cases limiting the model size performs better than not limiting it, because our algorithm is able to remove outlier data. Figure 6.5 shows a graphical representation side by side of the average number of clusters in each configuration and the accuracy of the different descriptors.

Note that after a cluster-size limit of 20 the accuracy does not improve substantially, and hence it is reasonable to implement such limit in constrained platforms.

Discussion of related offline and online baselines The object recognition performance of our incremental model is compared with an offline recognition pipeline, and with two standard offline strategies for object recognition:

- $SVM + HC_{RGB}$: It uses HC_{RGB} as a descriptor and a SVM classifier trained offline.
- k-NN+*descriptor*: we run a standard k-NN classification, computing distance between the query and all the training data samples, for different descriptors. Note that it is equivalent to the limitless incremental model after processing data from 9 users.
- *Inception*-based: We have used the base Inception V3 model [95], with weights pre-trained on ImageNet, and fine-tuned it with our *Manually-cropped* patches for the 22-object classes in *MHRI* dataset.

We also compare our algorithm against an incremental Passive-Aggresive approach (PASVM) [39] applied to SVM[2], which updates the support vectors with each step, and with an incremental SoftMax Regression. We tested the PASVM with HC_{RGB} and ResNet50 descriptors and parameter $C = 2$ and the SoftMax with HC_{RGB}.

Table 6.1(c,d,e) shows the average object recognition accuracy of these baselines. For the online baselines PASVM performs poorly, but the SoftMax regression has a performance close to our approach. For the offline baselines, we can observe that our proposal (Incremental k-NN) has similar performance (35.2%) to Offline k-NN (33.3%). This result is a solid support for our incremental approach, as it shows that our strategy to limit the cluster size does not harm the performance. Our results are also improve over $SVM + HC_{RGB}$, a remarkable result taking into account that our current approach is incremental, while $SVM + HC_{RGB}$ used offline training. Among

[2]https://github.com/Zotkin/Passive-Agressive-SVM-for-online-learning

Table 6.1: Average object recognition accuracy (22 Objects) (10-fold), *Manually Cropped* patches

(a) **Incremental, k-NN**									
	# of users processed to build the model								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	10-cluster limit per class								
BoW_{ORB}	7,6	6,2	7,2	7,9	8,3	9,0	8,7	9,4	10,1
HC_{RGB}	10,7	17,7	23,7	25,8	25,9	25,9	23,4	25,1	24,3
SIFT	6,1	5,8	5,8	5,8	6,1	5,8	6,8	6,2	6,4
ResNet50	21,3	25,8	30,3	31,0	31,9	33,7	34,7	35,7	**35,2**
	20-cluster limit per class								
BoW_{ORB}	7,6	7,2	8,0	8,3	9,0	9,6	10,4	10,3	11,3
HC_{RGB}	10,7	17,9	23,1	26,0	28,0	29,6	30,8	31,1	31,4
SIFT	6,1	5,5	5,2	5,4	6,4	8,5	8,9	7,3	6,8
ResNet50	10,3	18,1	22,6	23,8	26,1	26,4	27,3	30,4	**31,5**
	30-cluster limit per class								
BoW_{ORB}	7,6	7,2	8,0	8,4	9,1	9,7	10,7	11,1	11,7
HC_{RGB}	10,7	17,9	23,1	25,8	27,5	28,4	29,7	30,7	30,0
SIFT	6,1	5,5	5,2	5,5	6,0	6,9	7,5	8,0	7,7
ResNet50	21,3	26,2	31,4	32,7	31,8	35,3	36,8	34,3	**33,1**
	No cluster limit per class (ALL)								
BoW_{ORB}	7,6	7,2	8,0	8,4	9,1	9,7	10,8	11,3	11,8
HC_{RGB}	10,7	17,9	23,1	25,8	27,6	28,4	29,2	30,3	30,2
SIFT	6,1	5,5	5,2	5,5	6,0	6,9	7,2	7,3	7,3
ResNet50	21,3	26,2	31,4	32,7	31,8	35,3	36,5	34,0	**33,3**

(c) **Incremental SVM**									
$ResNet50$	3,8	5,6	5,6	7,1	8,1	8,1	8,1	8,1	8,1
HC_{RGB}	3,6	8,0	8,0	9,0	9,0	9,0	9,0	9,0	9,0

(d) **Incremental SoftMax Regression**									
HC_{RGB}	10,6	12,6	17,4	20,7	22,1	24,9	25,7	27,7	28,8

(e) **Offline**	
k-NN+BoW_{ORB}	11,8
k-NN+HC_{RGB}	30,2
k-NN+$SIFT$	7,3
k-NN+$ResNet50$	33,3
$SVM + HC_{RGB}$	34,8
Inception-based [95]	**59,3**

the offline approaches, the *Inception*-based model obtains the best results. Notice, however, that for this approach the target object patches are manually cropped and training is done offline. Hence, it is an approach not suitable for our case of study, which is incremental learning. We consider this to be an upper bound for the performance, worth showing as reference.

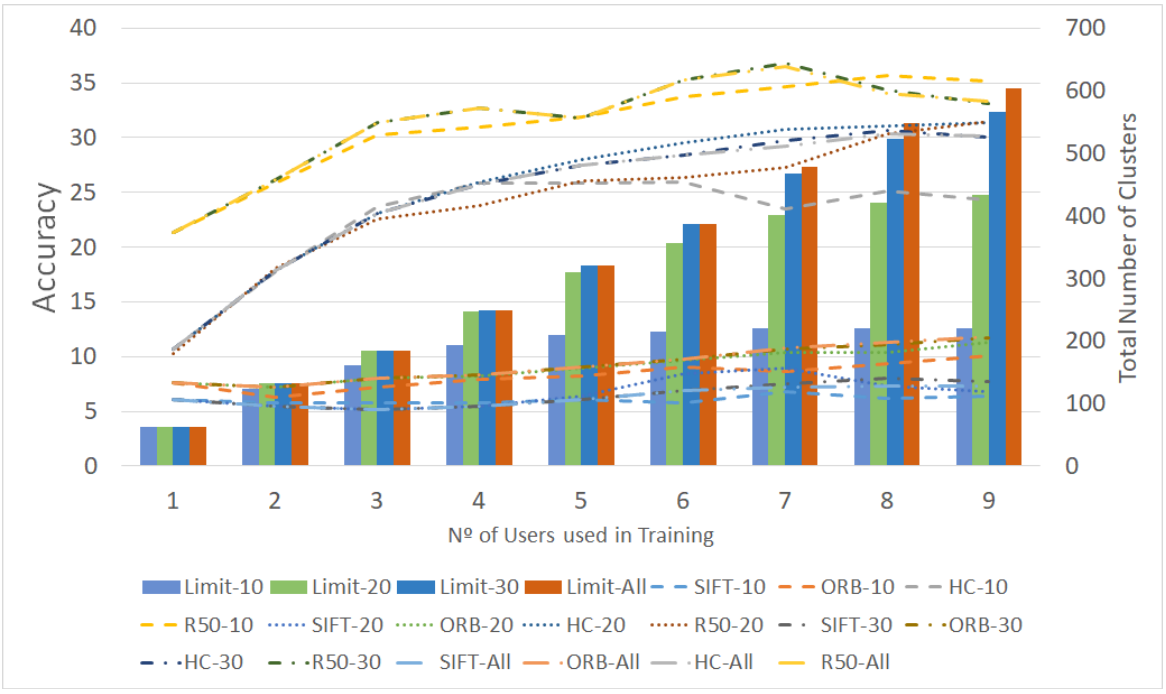


Figure 6.5: Accuracy and number of clusters of our incremental learning algorithm as the number of users increases. Descriptors reported: HC_{RGB}, FC7, SIFT, BoW_{ORB}. Model sizes evaluated: 10, 20, 30 and All.

Chapter 7

End-to-end incremental object learning from HRI

7.1 Introduction

This chapter presents the complete integration of the end-to-end pipeline that learns objects incrementally from the Human Robot Interaction. The correct behavior and performance of all the modules running together is validated using the *MHRI* dataset in a experiment. The validation is done using this dataset since is the only one that contains both interaction and object recognition in the wild.

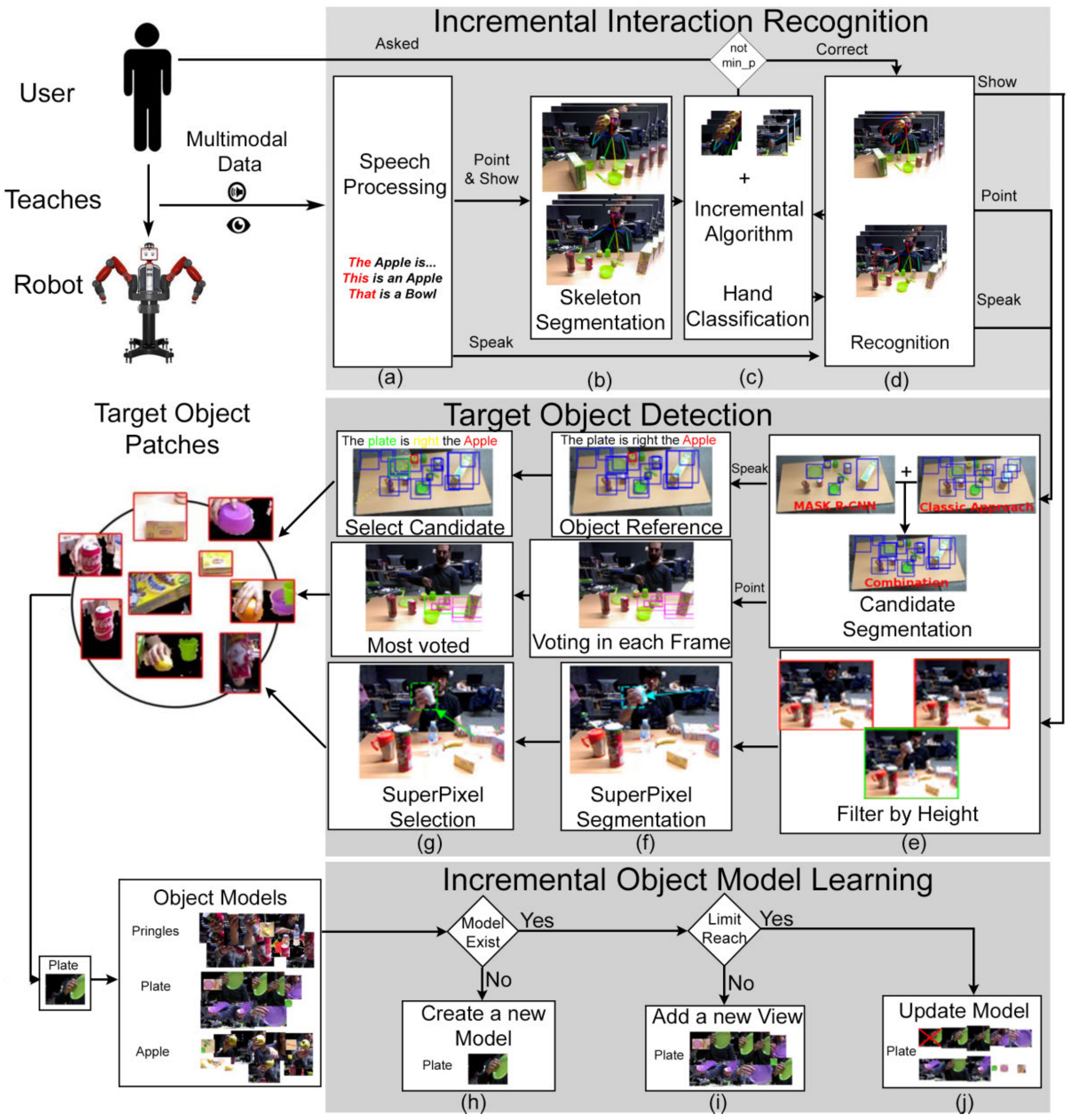


Figure 7.1: Overview of the end-to-end pipeline that learns objects from human teaching in a HRI scenario.

7.2 Overview

Following the Figure 7.1, we have integrated the modules presented in previous chapters into a complete pipeline. The pipeline works as follows:

- A user teaches an object to the robot, using the interactions presented before.
- The first module recognizes the interaction type. It can also interact with the user for clarification if the recognition is uncertain.
- The frames, hand position, type of interaction and speech is the input to the object detection module. This module obtains patches containing the target object.
- The patches obtained are then used to train our incremental model and update the object model database.

An experiment with a real robotic platform (a Baxter robot) was done following this full pipeline. This experiment is explained in Section 7.4

7.3 Evaluation

We run the complete pipeline for all the videos and extract the target object patches. Then, the incremental algorithm we propose is run with a 10-fold cross-validation, where each fold corresponds to a user, but using the automatically segmented patches. The difficulty, in comparison with the previous chapter evaluation, is also increased because each user manipulates a different object pool subset and then at some points in time, there may be no examples in the training data for some of the objects in the test data.

We run this experiment using the $ResNet50$ descriptor, which had the overall best performance in previous chapter, and the HC_{RGB} descriptor, with a more robust performance in the *Core50* data.

7.3.1 Incremental k-NN

Table 7.1 and Table 7.2 show the accuracy for object recognition with the HC_{RGB} and $ResNet50$ descriptors, respectively, at different steps of the incremental process and for the different folds.

In these experiments the descriptor that works the best is $ResNet50$, obtaining an accuracy of 18.2% with all users processed. Results show that the incremental approach varies around 20% with each user added after the third user processed, as shown in

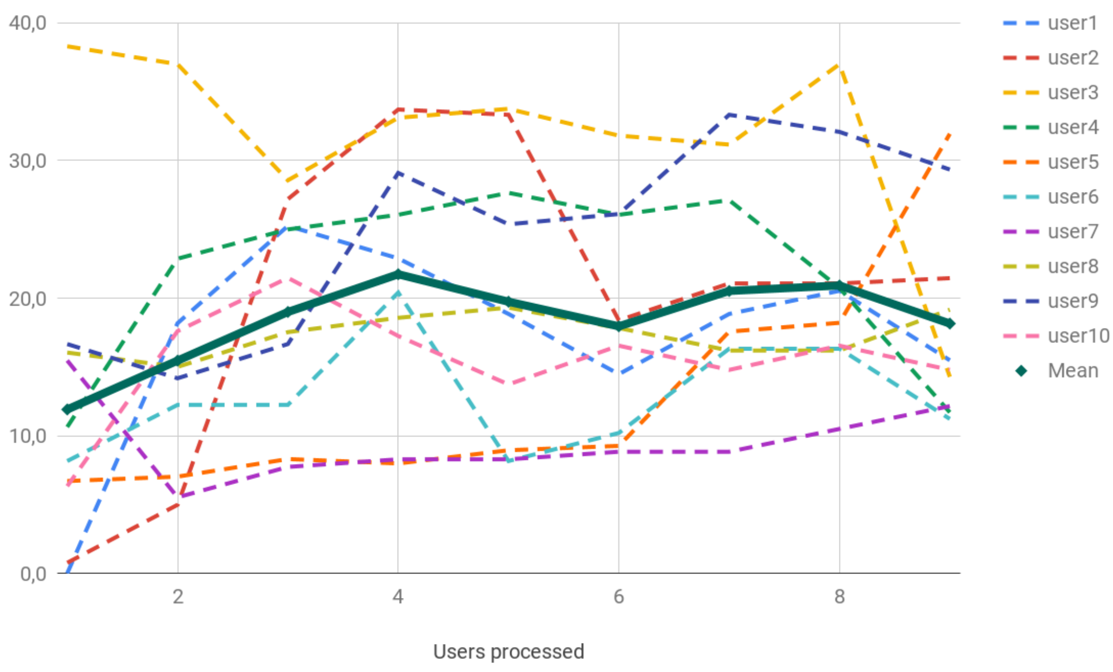


Figure 7.2: Incremental learning accuracy as more data (from more users) is used for training. Dashed lines are per-fold results, solid line is the average. Table 7.2 contains the numerical results for this graph. (best viewed in color)

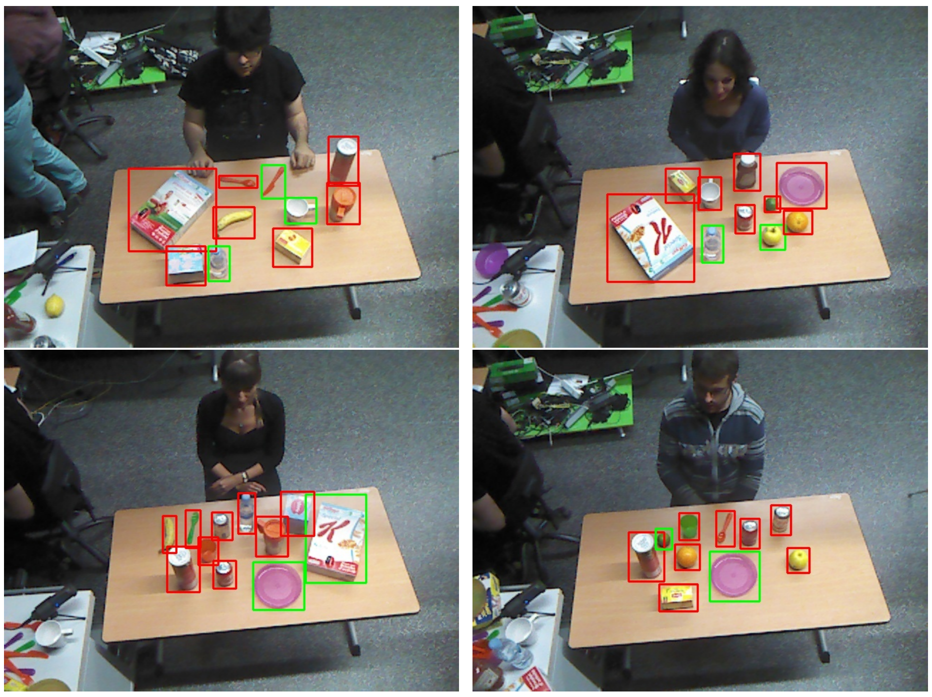


Figure 7.3: Examples of the recognition results after the incremental learning was run. The objects in green are correctly labeled. Best viewed in color.

Figure 7.2. HC_{RGB} obtains worse results, with 13.9% of accuracy, but shows a more constant progress as more users are processed. It is also interesting that, as shown in Table 7.4, the size of both models is small compared to other baselines.

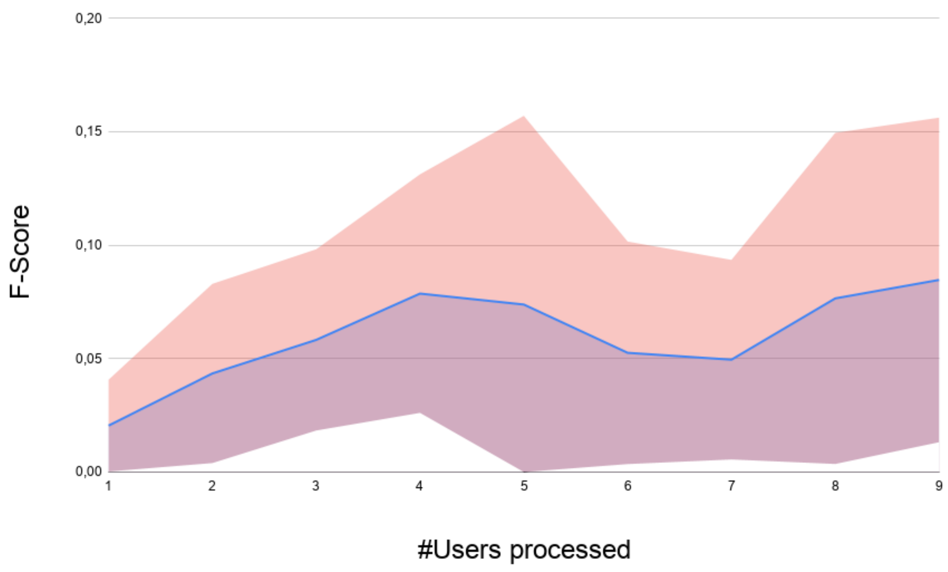


Figure 7.4: Evolution of the F_1 score with the amount of training data over 10 folds. The blue line stands for its average, and the colored area for its standard deviation.

Table 7.1: Object Recognition Accuracy (HC_{RGB}, model size 20). Columns: # training users. Rows: Results per fold.

# users	1	2	3	4	5	6	7	8	9
user1	0,3	0,3	1,1	0,9	0,9	1,7	0,6	0,9	**1,4**
user2	2,0	4,0	1,6	6,0	3,6	6,0	1,2	4,4	4,8
user3	17,0	4,4	5,6	8,5	18,1	17,0	23,0	24,8	15,9
user4	3,3	7,4	2,8	5,6	6,5	4,7	5,1	5,1	11,2
user5	4,1	10,1	9,9	11,2	7,7	2,5	8,2	7,4	9,3
user6	12,2	20,0	8,9	3,3	1,1	12,2	13,3	23,3	21,1
user7	26,3	3,2	20,6	18,6	12,6	17,0	19,8	30,4	**29,6**
user8	4,7	6,5	13,7	12,2	15,7	16,9	15,2	14,5	13,5
user9	14,0	19,5	25,5	11,0	11,5	15,5	16,0	14,5	15,0
user10	2,0	1,5	3,0	27,3	29,8	27,8	23,7	22,2	17,2
Avg.	8,6	7,7	9,3	10,5	10,7	12,1	12,6	14,8	13,9

7.3.2 Comparison with incremental and offline baselines

As a reference **incremental approach baseline**, we run incremental SoftMax Regression. Table 7.3 shows its performance. Its performance is significantly lower compared to the experiment in Table 6.1 (it decreases from 28% to 8.6%). This is a consequence of the lower quality of the data (for the results of Table 6.1, *Manually Cropped* patches were used). Notice that, in our approach, the degradation is not as significant, and we outperform this baseline.

Table 7.2: Object Recognition Accuracy (*ResNet*50, model size 10). Columns: # training users. Rows: Results per fold.

# users	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
user1	0,0	18,2	25,3	22,9	18,9	14,5	18,9	20,5	15,5
user2	0,8	5,0	27,2	33,7	33,3	18,4	21,1	21,1	21,5
user3	38,3	37,0	28,6	33,1	33,8	31,8	31,2	37,0	14,3
user4	10,6	22,9	25,0	26,1	27,7	26,1	27,1	20,7	11,7
user5	6,7	7,0	8,3	8,0	8,9	9,3	17,6	18,2	**31,9**
user6	8,2	12,2	12,2	20,4	8,2	10,2	16,3	16,3	**11,2**
user7	15,5	5,5	7,7	8,3	8,3	8,8	8,8	10,5	12,2
user8	16,0	15,0	17,5	18,6	19,3	17,8	16,2	16,2	19,2
user9	16,7	14,2	16,7	29,1	25,4	26,1	33,3	32,1	29,4
user10	6,3	17,6	21,5	17,3	13,7	16,5	14,8	16,5	14,8
Avg	11,9	15,5	19,0	21,7	19,7	18,0	20,5	20,9	18,2

Table 7.3: Object Recognition Accuracy with SoftMax Regression (HC_{RGB}). Columns: # training users. Rows: Results per fold.

# users	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
user1	14,4	0,4	0,4	0,2	0,2	4,0	0,0	11,3	7,5
user2	20,8	0,0	22,0	0,0	0,0	13,6	0,0	0,4	0,4
user3	1,5	0,4	0,7	11,9	0,7	0,7	0,4	1,9	17,8
user4	10,2	16,3	0,5	0,9	0,5	2,3	1,4	12,6	0,9
user5	0,3	0,0	15,1	0,0	0,0	0,3	4,1	3,0	7,9
user6	3,3	1,1	1,1	2,2	2,2	3,3	18,9	0,0	11,1
user7	0,0	4,0	0,4	0,4	6,9	1,2	0,4	0,8	**21,1**
user8	1,5	9,0	12,5	0,3	11,5	0,2	8,7	0,0	**0,2**
user9	0,0	0,0	13,6	0,0	0,0	2,7	14,6	0,7	13,6
user10	2,0	0,0	14,6	25,3	5,6	1,0	0,0	0,0	0,5
Avg.	12,2	0,3	7,7	4,0	0,3	6,1	0,1	4,5	8,6

As a reference **baseline for our incremental end-to-end approach**, since up to our knowledge there is not another available of similar characteristics to ours, we show our earlier work presenting the dataset [96]. It consists of a SVM classifier trained offline using HC_{RGB} as patch descriptor. We consider two configurations, depending on the data used for training, with the best result obtained for each of them:

- **Automatic** $SVM + HC_{RGB}$: SVM trained with all the patches extracted automatically, i.e., including significant amount of noisy patches.
- **Inspected** $SVM + HC_{RGB}$: SVM trained with automatic patches manually

Table 7.4: Recognition results using *Automatic Patches* (22 classes, 10-fold cross validation, random acc. 4.45%)

	Accuracy	**STD**	**Size**
Previous Work (offline) [96]:			
Automatic $SVM + HC_{RGB}$	7.95	6.6	-
Inspected $SVM + HC_{RGB}$	11.45	10.53	-
Other offline baselines:			
Automatic *Offline k-NN(ResNet)*	23.98	8.85	10MB
Automatic *Offline k-NN(HC)*	13.6	9.6	2MB
Automatic *Inception-based*	35.5	6.51	92MB
Incremental:			
SoftMax	8.6	8.7	2.5MB
Incremental-ResNet	18.2	7.4	3MB
Incremental-HC	13.9	8.0	220KB

inspected to keep only correct ones.

Besides, the same **offline baselines** from previous section are shown as reference. However, this experiment runs them using automatically segmented patches:

- **Automatic Offline k-NN:** standard nearest neighbour classification using *Automatic patches.*
- **Automatic *Inception*-based:** fine-tuned CNN model as in our previous experiment, but using *Automatically segmented* patches.

Table 7.4 shows the average accuracy (of the 10-folds) obtained for the different approaches run to learn object models. The performance of *Inception* and *Offline k-NN* decreases to an 35.5% and 23.98%, from the $59,3\%$ and $33,3\%$ they reached training with *Manually Cropped* patches in earlier experiments. This is not surprising and confirms the challenging set up we are working with. The decrease in performance is due to error accumulated from running each of the modules and the lower quality of the data used for training. Figure 6.2 examples show that there is high amount of noise, partial views of the objects and heterogeneous patch sizes. As we already discussed in the object segmentation evaluation, only around 60% of the patches actually contain the object targeted.

Our incremental approach also suffers a decrease in performance but it is able to outperform the *Automatic* $SVM + HC_{RGB}$ and *Inspected* $SVM + HC_{RGB}$ baseline of [96] processing only 20% of the data, using the *Resnet* descriptor. In case of the *Inspected* $SVM + HC_{RGB}$ is really important because it uses manually pruned data.

It is also worth noticing, that the size of data stored by our incremental approach is several times smaller than the offline baselines, therefore requiring less resources. Note that in this case the other offline baselines are not much better than our incremental approach, which highlights the challenging data and setup considered and leaves open research problems in learning for service robotics.

7.4 Integration in a real robotic platform

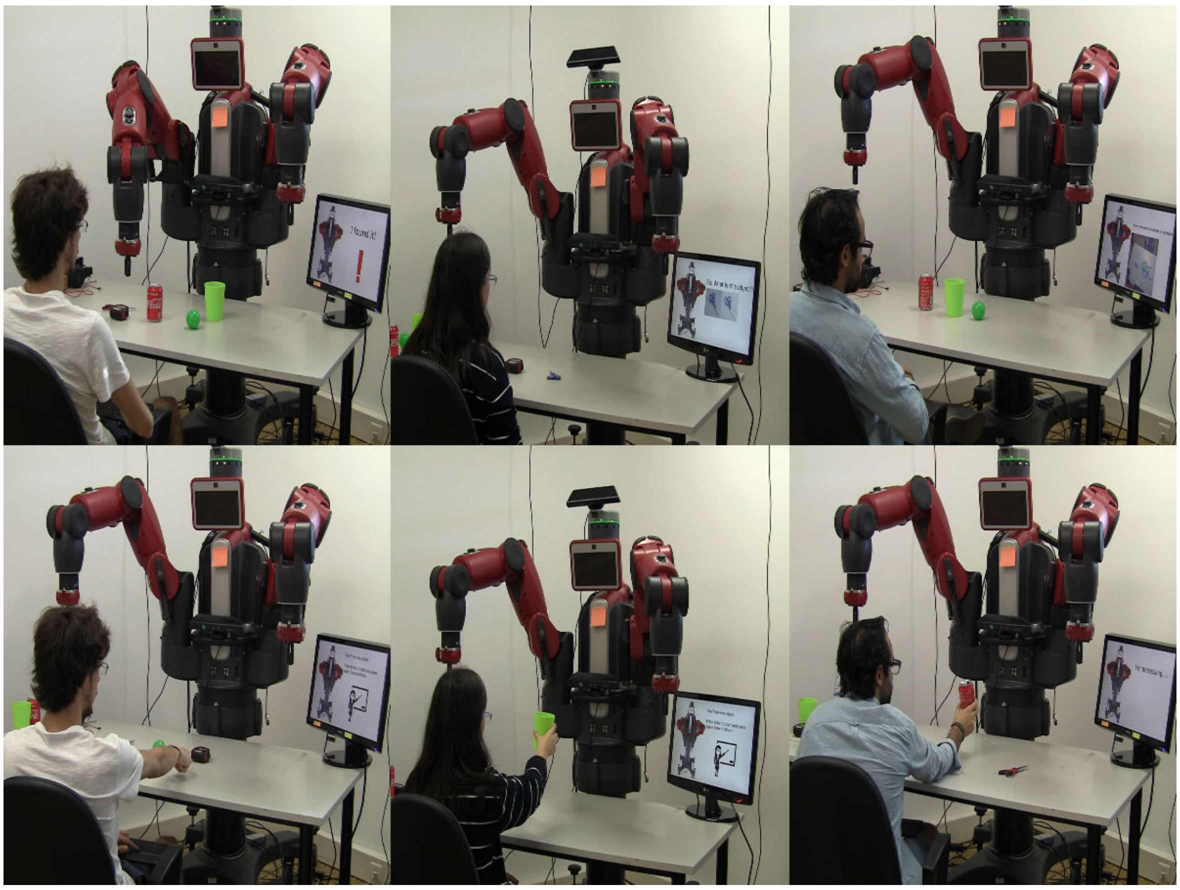


Figure 7.5: Photos of the three users in the demo done with the Baxter robot. Observe the experimental setup: The robot is in front of the table and the user is teaching objects. The screen helps the user to follow the robot instructions. The user speech and synthesized speech for the interaction can be heard in the video.

A live demonstration with a real Baxter robot was performed in the GAIPS laboratory of the *Instituto Superior Tecnico* of Lisbon. For the purpose of this demonstration, a new feature was added to the framework. The user could ask for an object and the robot either points to the predicted object in the table or says that it is not found. The objects in the table are segmented using our point and show strategies and the patches are processed by the incremental algorithm. The demonstration was done by three different users following these steps:

- The robot start without previous knowledge.
- The user teaches two or three different objects, each one with a different type of interaction.
- The user asks for one object of the table. The robot points to the object.
- The user teaches a new object, and can ask the robot to look for the ones that the user already taught.

In Figure 7.5 a qualitative result of the experiment can be seen. The bottom image shows the setting and the top image shows the computer screen illustrating the different steps. The video showing the experiment is publicly available [1].

[1]`https://www.youtube.com/watch?v=V_72tyBK8Go`

Chapter 8

Incremental object learning with online descriptors

8.1 Introduction

In the previous chapter a complete pipeline for interactive and incremental object learning was presented and evaluated. The experiments showed that the performance of the incremental algorithm suffers from the noisy data that is extracted using the pipeline. A robust and discriminative image description is essential to reduce this deterioration and obtain a performance similar to the cleaned data experiment. The work presented in this chapter explores the idea of adapting the descriptor as new data appears, in an attempt to have a more suitable and discriminative descriptor.

The presented approach follows the strategy summarized in Figure 8.1. A deep embedding is used as patch description, and shows the benefit of updating the encoder that computes the embedding. This update uses a reduced set of representative patches and only runs after a batch of new patches has been processed. The evaluation of this approached is run on two public datasets for continual learning. This evaluation includes a careful analysis of the effects of *catastrophic forgetting* on different variations, demonstrating the accuracy of our strategy and how we significantly reduce the amount of training samples that need to be stored. The experiments also demonstrate how lighter encoders can be used without degrading the final accuracy, thanks to the suggested encoder retraining.

8.2 Related work

Deep learning is being adopted by the incremental learning community, but several challenges remain. The main challenges studied in this work are adaptive representations and model updates that enable new categories without degrading the performance on categories learned earlier.

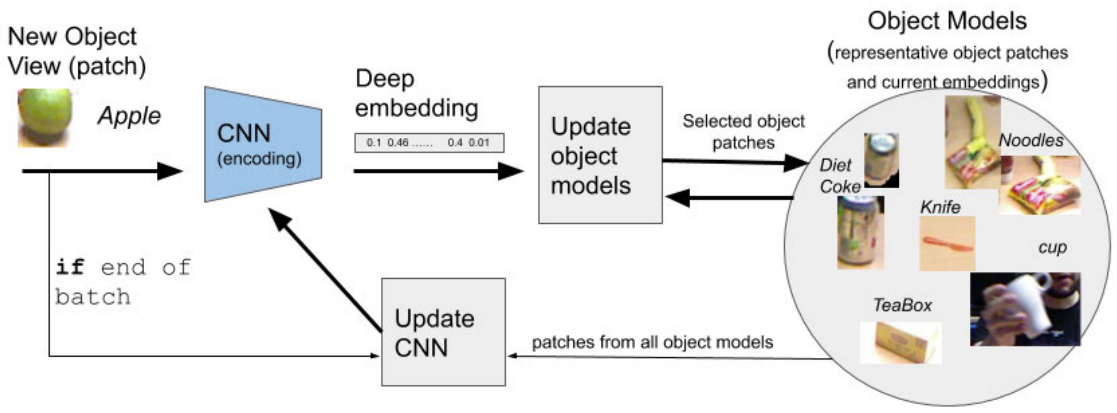


Figure 8.1: Incremental object learning. This chapter explores the effects of incrementally updating the image representation (CNN-based embedding) as new object classes are added and learned.

Similar to our work Rebuffi *et al* [97] use deep learning to encode image content. They use this embedding to learn models incrementally using Near Mean Classification. Lomonaco *et al.* [98] present a study where they use different set of the training data for the Core50, one that extend the new class apparition in comparison with the original one, and its effect on the metrics.

In order to update (re-train) the networks without forgetting old data, several works use the distillation loss. For example, Castro *et al.* [99] use several classification layers and keep a reduced set of data from each class to maintain a low value for the distillation loss. Hou *et al.* [100] use the same idea but with three loss functions: One for new data, one for distillation and one for inter-class separation. Lagunes *et al.* [101] use triplets to train a new embedding with the known data and test it on novel data using K-NN. On class-incremental learning, where new classes can be added over time, Maltoni *et al.* [102], combine architectural and regularization strategies. Sodhani *et al.* [103], use Recurrent Networks with Gradient Episodic Memory.

More recently, Parisi *et al.* [104] present a growing dual memory architecture using Self-Organizing networks. They focus on how to learn the representation of their dual memory (two growing recurrent networks) and, differently from us, they use image embeddings obtained from a static CNN model. Our work focuses on the complementary task of dynamically update image embeddings as new data and classes appear.

8.3 Online trained descriptors

As a general overview, we follow the same scheme: as new data is received, a limited amount of selected representative *object views* is stored and redundant non-informative views are discarded. The new step explored in our system, and our main contribution, is updating the model used to describe these views, i.e., the possibilities and effects of re-training the CNN used to compute the embeddings.

8.3.1 Object recognition strategy

Each *object model* in our database consists of a limited set of representative views (which can be interpreted as cluster centroids) that is incrementally updated as detailed in next subsection. Each representative view is encoded with a *deep embedding*, learned using common CNN architectures. Specifically, we compare three different options (VGG16, ResNet and MobileNet) in our experiments. Our goal is to incrementally update the CNN encoder to achieve a more discriminative description across objects as the data comes.

The recognition step using the incrementally learned model is run following a standard k-Nearest Neighbor (k-NN) classification. The distance between the embedding of a new view and the embedding representing each existing model cluster is computed, and the view is assigned the label according to the most frequent ($Mode$) label within the closest k neighbours found as follows:

$$\begin{aligned} D_v &= ||emb_{view} - emb_{cluster}||, \ \forall cluster \in database \\ x_{0:k} &= \text{sort}(\text{distances})[0:k] \\ l^{\hat{x}} &= Mode(l^{x_{0:k}}) \end{aligned}$$

where *sort* is the function that sorts the cluster elements by the calculated distances and emb_x is the descriptor obtained from cluster x.

8.3.2 Incremental model

Model initialization. Initially our system assumes no prior knowledge on the actual target classes of the system, i.e., our database is empty. A CNN pre-trained on a large object dataset is used as initial encoder, Enc, to obtain the embedding of new object views as they arrive. As frequently done, the embedding is obtained as the normalized output of the last global pooling layer before the final classification layer.

Model update. To run a full update step, a batch containing several new views is required, although these views can be partially processed as they arrive.

The embedding for each new view is computed using the current encoder model (Enc). A new cluster is initialized with the embedding as centroid and annotated with a given view label L. Two alternatives can happen next:

- **Label L does not exist in the system.** L is added to the database, with its new object model composed only of the cluster that was just initialized.
- **Label L already exists in the system.** We incorporate the new cluster to the model of the corresponding object. In case the number of clusters associated to L has reached the limit per class (S), we select the most representative information to be kept. We explored several options for this selection and, as shown in the experiments, the best results were obtained computing the inter-distance between all cluster centroids with label L, and merging the closest pair of them.

After a whole batch of new views has been processed, the encoder is updated to learn a more discriminative representation given the current database content. The final classification layer of the CNN is changed to match the number of objects in the database and the views corresponding to the centroids of all clusters in the database are used to finetune the encoder CNN *Enc*. Note these are the only images stored.

8.4 Evaluation

Method variations. In the experiments that we present next, all strategies use a model for object recognition as described in Sec. 8.3.1. The two base strategies considered are; a first one built offline (**Off**), i.e., with all images processed at the same time, and a second one built incrementally (**Inc**). We analyze the effect of applying a data selection step (+**d**) and encoder re-training step (+**r**). No data selection means there is no limit on the amount of training samples that can be stored ($S = \infty$). No re-training means all the embeddings are the ones of the base encoder models.

Object recognition configuration. The following parameter values are set for our approach in the experiments: maximum number of clusters per object (S) is 20 for MHRI and 120 for Core50, and $k = 5$ for the k-NN classification in all cases. The encoders considered to compute the deep embedding in our experiments are MobileNet [67], because of its compromise between performance and resource consumption, ResNet50 [105], because it is a commonly used architecture for object recognition, and VGG16 [106], because it was used in the Core50 baseline methods.

Training configuration. Regarding the incremental models, the update of the encoder *Enc* used for the embeddings consists of a two step fine-tuning. In the first step, all layers are frozen except the last one for 5 epochs with an SGD and a learning rate of 0.001. In the second step, all the network layers are finetuned for 25 epochs with an SGD and a learning rate of 0.0001. Experiments with more epochs showed the accuracy did not improve significantly.

For the offline baselines, the re-training is done for 50 epochs with SGD and learning rate 0.001.

Evaluation metric. We report the accuracy (correct predictions over the total) for all our experiments. For MHRI, we report the average accuracy over the 10-folds (each fold is created by separating all the data corresponding to a user). For Core50 the accuracy reported corresponds to the one of the official test split.

Table 8.1: Accuracy in MHRI using our incremental system (Inc+d+r) trained with different data selection strategies.

	MobileNet	ResNet
Closest Cluster	**72,87**	**74,44**
Combination	69,69	59,09
Random	72,1	65,31

8.4.1 Ablation study and variations our approach

Data selection strategy

We explored several options to select the most representative views that form our object model: 1) Our strategy (Closest Cluster), that consists on computing the distances between all cluster centroids of the same label L, and merging the closest pair of clusters. 2) The selection strategy that combinates merging closest and discarding spurious data (Combination). 3) A random selection strategy (Random). Table 8.1 shows the object recognition results after training our incremental approach (Inc+d+r) applying the different strategies using two different encoders (MobileNet and ResNet). The best results were obtained with the Closest Cluster selection. Differences in their computational cost are not significant compared to the rest of computations.

Ablation study

We analyze the effect of the data selection and retraining steps of our approach with two public benchmarks.

Results with MHRI are summarized in Figure 8.2. The plot represents how accuracy changes as additional incremental update steps are run (i.e., new data batches are sequentially processed) for the different method variations described in Section 8.4. It also displays the total amount of training time spent. The offline results are shown as an additional reference to value better the overall results. We see how the incremental approaches get better results because they do adapt to the new data as opposed to the offline baseline, which works best only at the beginning). Table 8.2 details the object recognition accuracy and training time for the different approaches after all data has been sequentially processed.

Results with CORE50. Similarly to previous experiment, Figure 8.3 represents how accuracy changes as the incremental steps are run using the Core50 dataset, and Table 8.3 details the accuracy and training times corresponding to the end points of the plot (all data has been processed).

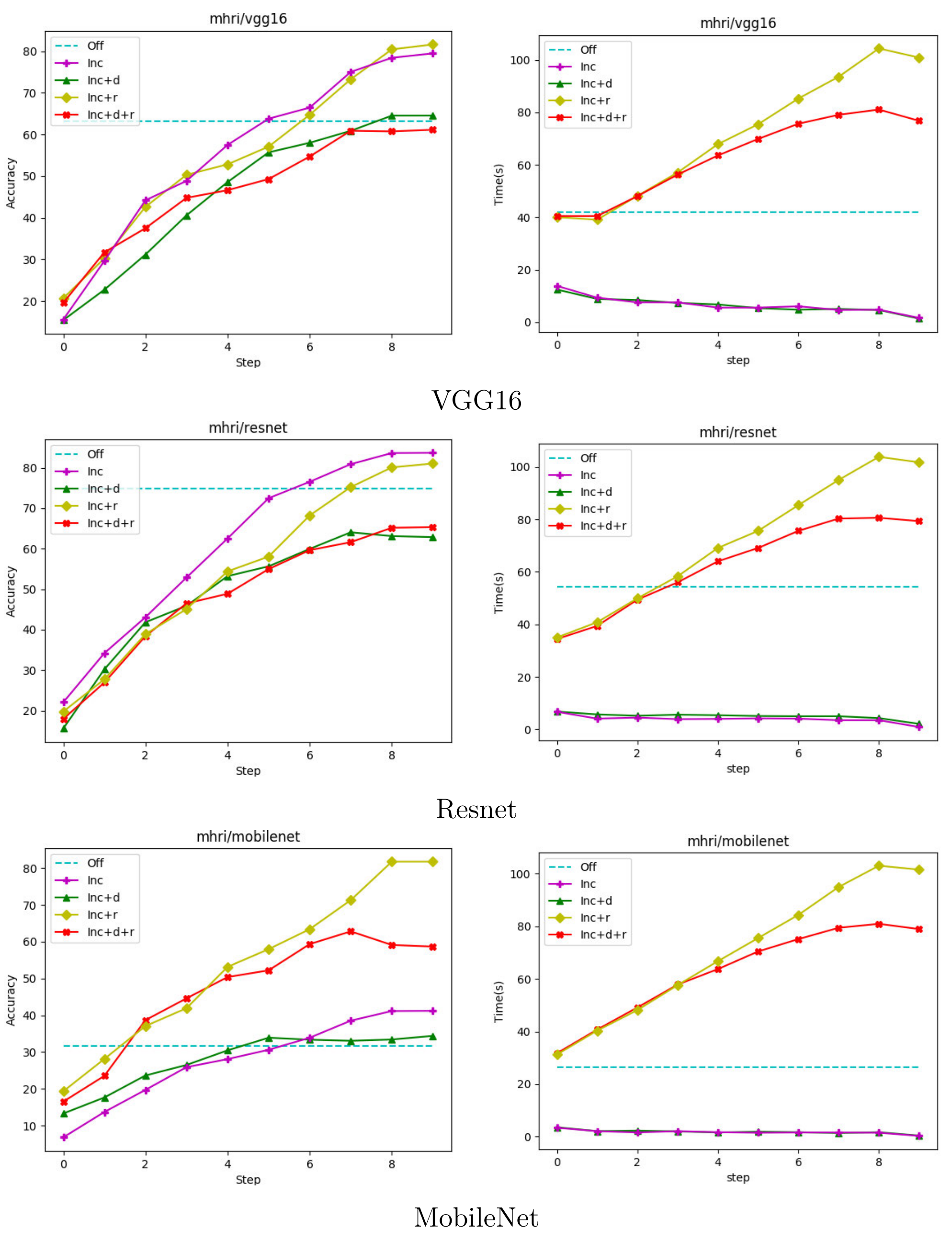
Table 8.2: Accuracy (*acc*) standard deviation (*std*) and training time (T) in seconds (10-users cross-validation) using MHRI.

	MobileNet			**ResNet**			**VGG16**		
	acc	*std*	T	*acc*	*std*	T	*acc*	*std*	T
Off	31.6	5.6	26	**74.7**	5.1	54	63.3	4.7	42
Inc	41.2	6.7	17	**83.7**	3.5	39	79.5	8.1	67
Inc+d	34.4	7.2	19	62.9	8.2	50	**64.5**	8.7	66
Inc+r	**81.8**	4.8	703	81.0	6.1	715	81.6	5.9	712
Inc+d+r	58.7	7.6	628	**65.3**	8.3	628	61.1	9.3	632

Discussion. We can observe the results are very similar with both datasets. The main difference comes from the effect of the *data selection step*. It always enables the system to bound the training time, but while in MHRI the accuracy degrades a little bit, in Core50 it is able to keep the same level of accuracy. This is probably due to the more noisy and cluttered set up presented in MHRI data, which makes it harder to store a very limited set of sample views without degrading representativity of the object models.

Re-training the encoder does not improve much when using ResNet or VGG16, where all the curves present a similar behaviour. In particular for the ResNet encoder, results suggest that the capacity of the network is big enough to learn a wide array of patterns from large datasets. However, the *re-training step* improves the performance significantly when using MobileNet as encoder. It achieves lower accuracy without re-training (**Inc** and **Inc+d**) but obtains similar or better results than VGG16 and ResNet after re-training (**Inc+r** and **Inc+d+r**). This is interesting because MobileNet is a lighter architecture than ResNet, as well as faster to train. Even though it starts with a less generic encoder, our incremental strategy allows it to adapt to the new domain and match the performance of much more complex networks (such as ResNet50 in this case) at a reasonable cost. Therefore our results point to MobileNet as a more suitable option for this type of incremental strategies.

These results justify the main steps included on the proposed incremental learning strategy, the data selection as well as the encoder re-training. They allow to improve the performance of an efficient network such as MobileNet as the target domain changes, since that architecture presents higher efficiency but lower generalization capabilities unless updated. Interestingly, the computational cost of such incremental update is reasonably low, which is relevant in a continual learning scenario where the computing resources might be limited.

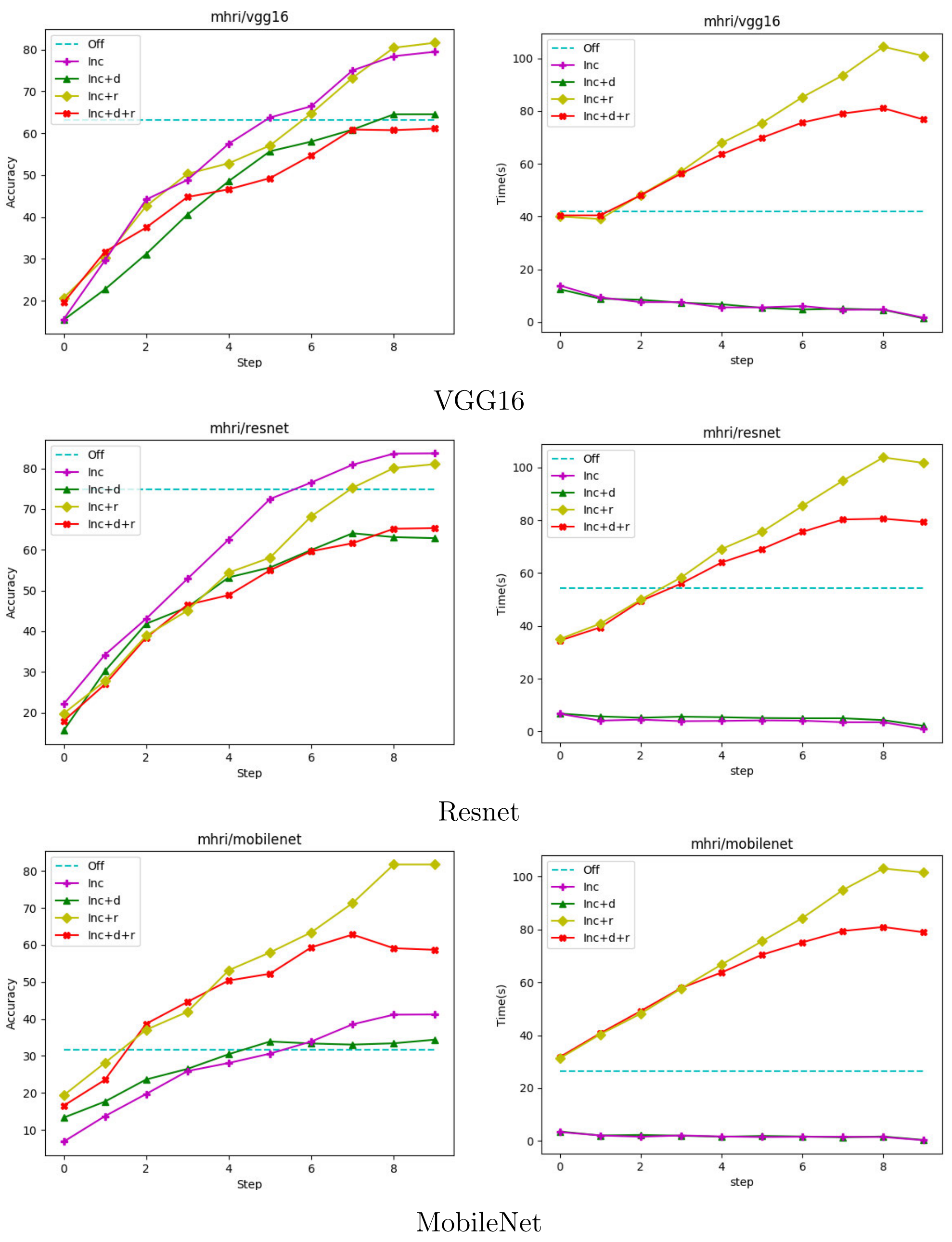


(a) Accuracy (b) Total time execution (Train+Test)

Figure 8.2: Accuracy (first column) and training time (second column) at different steps of the incremental process using MHRI data. (Average of 10-fold cross validation)

Table 8.3: Accuracy (*acc*) and training time (*T*) in seconds using Core50.

	MobileNet		**ResNet**		**VGG16**	
	acc	*T*	*acc*	*T*	*acc*	*T*
Off	13.57	76	**65.14**	151	60.92	128
Inc	13.88	1212	**63.95**	2000	56.80	2052
Inc+d	14.73	1171	**62.65**	1918	57.72	2004
Inc+r	**63.84**	40643	63.03	40339	63.47	41025
Inc+d+r	**64.73**	38591	64.69	39433	63.27	38711



(a) Accuracy (b) Total time execution (Train+Test)

Figure 8.3: Accuracy (first column) and training time (second column) at different steps of the incremental process using Core50.

8.4.2 Incremental learning baselines and challenges

This section shows our results compared to other incremental learning baselines, and analyze the good properties of our approach with respect to the main challenges of continuous learning.

Comparison against baselines

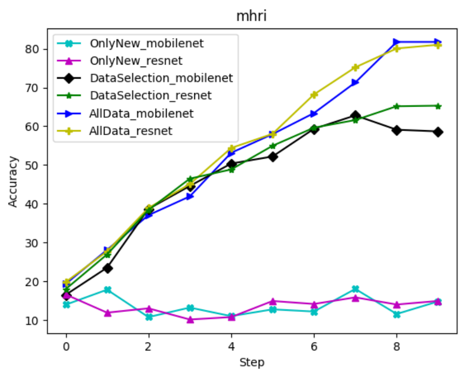
When comparing to recent baselines obtained in the available benchmarks, our method outperforms the baselines for the MHRI dataset presented in Chapter 7 by a large margin (we obtain an average accuracy of 65.3% against 18.2% and 35.5%). Our method obtains slightly better accuracy (64.7%) than one of the recent baselines run on Core50 (64.1%) obtained in [34], even though this baseline is an accumulative approach that stores all the training data. Our results are below those from a more recent but complex approach presented in [104]. They obtain 87.1% with a novel memory system focused on how to exploit the temporal relations of the inputs, after using a fixed encoder to obtain each image embeddings. Our approach demonstrates insights that are complementary to this last approach, about how to run an efficient update of the encoder to provide a more adequate and adapted embedding.

Analysis of incremental learning challenges

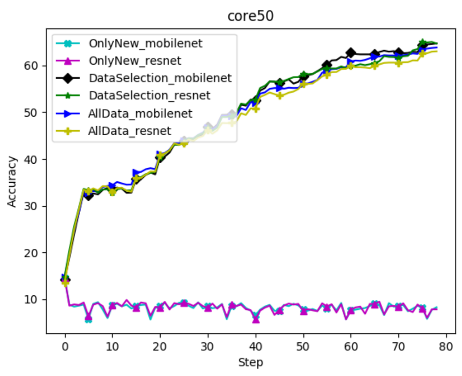
Finally we analyze the behaviour of our system with respect to the main challenges we want to deal with.

Catastrophic forgetting. To evaluate the effects of catastrophic forgetting on our approach, we compare our approach (using data selection) to a variation that only re-trains using the new data, and to another variation that keeps and uses all data to retrain. Figure 8.4 shows the comparison of these three alternatives, using ResNet and MobileNet on both MHRI and Core50. In both datasets, models using only new data stall at around 10%, showing the catastrophic forgetting effect. As expected, the accuracy of the other strategies grows as more batches are used for training. Our data selection scheme reaches an accuracy that is similar to that of keeping all data, but requiring a significantly smaller amount of memory as detailed next.

Memory use. A relevant challenge in continual learning in real applications is the use of limited resources. Table 8.4 compares the memory requirements of our approach with and without the data selection, which indeed brings significant reductions. However, the base model network size is the most critical factor, both in execution time and GPU memory requirements. We should remark again that our incremental algorithm facilitates the use of MobileNet, which is the smallest network, and allows it to match the performance of other larger networks. It is also remarkable that our data selection can reduce the stored data size significantly both in MHRI and Core50, although such reduction pales in comparison with the one related to the network.



MHRI Dataset



Core50 Dataset

Figure 8.4: Catastrophic forgetting. Comparison between only new data, w/ our data selection and w/ all data stored.

Table 8.4: Memory requirements from data and networks used in different incremental experiments.

	Main memory (MB)	**GPU memory (GB)**
Core50 (Inc+r)	16.1	–
Core50 (Inc+d+r)	12.3	–
MHRI (Inc+r)	6.2	–
MHRI (Inc+d+r)	2.0	–
MobileNet	14.0	4.4
ResNet	98.0	27.1
VGG16	528.0	26.7

Chapter 9

Conclusion and Future Work

9.1 Conclusions

The main objective of this Thesis was ***investigating new methods for a robot to learn incrementally from multimodal user interaction***. As the main contribution of this research, an end-to-end pipeline for incremental learning of objects using human-robot interaction was developed and evaluated. This pipeline contains several novel and promising components in comparison with previous works, and it outperforms existing baselines. The following conclusions can be extracted from each of the modules that were proposed:

- A first step for a robot to learn from humans is identifying the type of human interaction that is occurring. Three natural interactions were explored in this thesis, namely *Point*, *Show* and *Speak*. For the three, two different approaches were designed: an offline approach pre-trained in hand patches, and an online approach that interactively adapts to each user by using questions when the interaction type is uncertain. Both approaches use visual and speech data. Our evaluation demonstrates that the online method is able to learn better interaction models than the offline one. *Point* and *Show* obtain an accuracy of 90% while *Speak* is completely separated using the speech data. A more accurate recognition of the interaction type helps the whole pipeline to obtain better regions of interest for the next steps. Besides, the proposed incremental learning of user interactions is able to adapt to new users and, potentially, to new interactions.

- Once the interaction type is recognized, the region of interest detected needs to be segmented. A different segmentation strategy was proposed for each type of interaction. Among them, the strategy for *Show* interaction obtains over 80% of correctly segmented patches, while *Point* and *Speak* strategies obtain around 50% each. Most of the errors in *Show* happen because the hand holding the object occludes more than half of the object. In the case of *Point*, most errors come from incorrect segmentation of the hand to obtain the correct direction. For *Speak*, errors in the recognition module for the reference object are the most frequent. Both *Speak* and *Point* use the same candidate segmentation approach: a combination based on deep learning and superpixels, which obtains better performance than using any of them separately.

- An incremental algorithm was developed to learn object models in a fully incremental scenario, where there is not a predefined number of classes and new data appear sequentially. The evaluation shows that the proposed approach outperforms other incremental approaches in the literature. From the descriptors evaluated in both datasets (*Core50* and *MHRI* dataset), deep learning descriptors pre-trained in large dataset (*ImageNet*) obtain the best results. These descriptors are the output of an intermediate layer from a CNN Network. Traditional hand-crafted descriptors obtain worse accuracy, since they are less robust to noise in the images.

- In our incremental learning setup, the robot needs to adapt as data change over time. Besides an incremental learning approach, a novel online retraining of the model used to compute the descriptor was developed. As previously said, this descriptor comes from the output of the final layers of an object classification CNN. In this thesis, it is proposed to update the descriptor over time. The evaluation of this novel approach shows promising results. The proposed approach outperforms descriptors extracted from models trained offline and obtains an accuracy close to offline baselines. In the presented experiments, thanks to the proposed update, a small network with focus on efficiency like Mobilenet, is able to obtain a similar performance in object recognition than a more complex network like ResNet. This is very useful to be able to achieve good incrementally learned recognition even in less resourceful systems.

For the evaluation of the complete pipeline and each of the contributions of this thesis, a new dataset (*MHRI* dataset) was recorded involving teaching human robot interaction and object recognition. The experiments in this thesis demonstrate that this type of realistic scenario is very challenging and the performance of state-of-the-art algorithms for object recognition is low. To our knowledge, the presented pipeline is the first to address object learning guided by natural HRI. The results show that the proposed approach outperforms well known baselines in the literature, while reducing the resources that are needed.

Robots might have very different hardware configurations and focus on very different tasks. In order to explore the generalization of HRI, a different use case was explored in this thesis. The selected scenario was drone human interaction, and a new benchmark for point direction recognition from drone point of view (*DDIR* dataset) was recorded. A novel approach that combines deep learning segmentation with deep learning classification was developed as baseline in the presented dataset. The proposed approach obtains good results in accuracy and show robustness to variations in the distance to the camera, user and scenario. The complete system can be run in an Jetson AGX Xavier onboard at roughly 1fps. However, when distances to humans are larger than 10 meters, the accuracy drops considerably.

9.2 Future work

While the main goal of this thesis has been fulfilled with a promising full pipeline, the results prove that there are still many challenges and more work needs to be done in this area. Specifically, the contributions of this thesis made us spot several interesting research lines and new challenges that could be investigated in the future, such as improvement in the efficiency of incremental methods, better segmentation techniques, more human robot interaction data and continuation in the drone human interaction scenario.

Incremental learning algorithms have improved substantially in the last years, and the approach developed in this thesis can be taken as the starting point for research in this direction. A more efficient execution while maintaining similar performance would help greatly to integrate these learning approaches in robot on-board computers, that typically have less resources that a common computer.

Deep learning-based techniques are making many areas progress, including some of the problems studied in this thesis. The modular pipeline proposed in this thesis allows an easy replacement of each individual module, that could lead to an overall improvement. In particular, integration with better or more recent segmentation methods would be a significant boost for the performance of this pipeline, since there are many steps that need to segment specific image regions.

The general HRI scenario considered in this thesis has proved to be challenging. One of the main problems of the interaction is that each user is and behaves differently, and sufficiently varied training data is costly to obtain. Works that focus on simulating or extending HRI data could lead to more accurate results in the interaction recognition and higher robustness to user variations.

This thesis work has also studied a different scenario, drone human interaction, that leads to a new line of work. As explained before, UAV points of view are completely different from service robot ones and need to be further studied. Controlling an UAV becomes harder as the complexity of the vehicle increase, so human interaction with autonomous driving is an interesting topic of study to free the user hands and improve the navigation of the UAV.

9.3 Publications and dissemination

Significant part of the work presented in this Thesis was published in the following articles:

- Pablo Azagra, Florian Golemo, Yoan Mollard, Manuel Lopes, Javier Civera and

Ana C Murillo. A multimodal dataset for object model learning from natural human-robot interaction. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6134–6141. IEEE, 2017. [H-index: 99 , CORE A]

- P. Azagra, J. Civera and A. C. Murillo, Incremental Learning of Object Models From Natural Human-Robot Interactions, in IEEE Transactions on Automation Science and Engineering, 2020.[JCR FI: 5.224, Q1]

Besides, part of the Thesis work was presented in the following international venues and published in the corresponding peer-reviewed workshop proceedings:

- Pablo Azagra, Yoan Mollard, Florian Golemo, Ana Cristina Murillo, Manuel Lopes, and Javier Civera. A multimodal human-robot interaction dataset. In Future of Interactive Learning Machine (FILM) Workshop in Conference on Neural Information Processing Systems (NIPS), 2016.

- Pablo Azagra, Javier Civera, and A.C. Murillo. Finding Regions of Interest from Multimodal Human-Robot Interactions. In Proc. on 2017 International Workshop on Grounding Language Understanding in the Interspeech conference (pp. 73-77).

- Pablo Azagra, Ana Cristina Murillo, Manuel Lopes, and Javier Civera. Incremental object model learning from multimodal human-robot interactions. In Workshop on Visually Grounded Interaction and Language (ViGIL) on NeurIPS 2018.

- Leon Barbed, Pablo Azagra, Lucas Teixeira, Margarita Chli, Javier Civera, Ana C. Murillo. Fine grained pointing recognition for natural drone guidance.In Workshop on Towards Human-Centric Image/Video Synthesis, Computer Vision and Pattern Recognition (CVPR) 2020.

- Pablo Azagra, Javier Civera, and A.C. Murillo. Incrementally Learned Embeddings for Continual Object Recognition. Submitted to British Machine Vision Conference (BMVC) 2020. (Under review)

The work from this Thesis was also presented in CHISTERA HLU Master Class that took place in Paris on 10th-11th of September 2018.

The code for the main framework presented and the two datasets recorded and labeled in this Thesis are public and can be downloaded. [1] [2]

[1]Code and MHRI dataset `https://sites.google.com/a/unizar.es/iglu_mhri/`

[2]DDIR dataset `https://sites.google.com/a/unizar.es/hri-drones/`

Chapter 10

Bibliography

[1] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1975–1981. IEEE, 2010.

[2] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[3] Da-Wen Sun. Inspecting pizza topping percentage and distribution by a computer vision method. *Journal of food engineering*, 44(4):245–249, 2000.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Fotios Dimeas, Filippos Fotiadis, Dimitrios Papageorgiou, Antonis Sidiropoulos, and Zoe Doulgeri. Towards progressive automation of repetitive tasks through physical human-robot interaction. In *Human Friendly Robotics*, pages 151–163. Springer, 2019.

[6] D. Meike, M. Pellicciari, G. Berselli, A. Vergnano, and L. Ribickis. Increasing the energy efficiency of multi-robot production lines in the automotive industry. In *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 700–705, Aug 2012.

[7] Y. Huang, Y. Zhang, and H. Xiao. Multi-robot system task allocation mechanism for smart factory. In *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pages 587–591, May 2019.

[8] Jacques Penders, Lyuba Alboul, Ulf Witkowski, Amir Naghsh, Joan Saez-Pons, Stefan Herbrechtsmeier, and Mohamed El-Habbal. A robot swarm assisting a human fire-fighter. *Advanced Robotics*, 25(1-2):93–117, 2011.

[9] Markus Bajones, David Fischinger, Astrid Weiss, Daniel Wolf, Markus Vincze, Paloma de la Puente, Tobias Körtner, Markus Weninger, Konstantinos Papoutsakis, Damien Michel, et al. Hobbit: Providing fall detection and prevention for the elderly in the real world. *Journal of Robotics*, 2018, 2018.

[10] A. Stewart, M. Cao, A. Nedic, D. Tomlin, and N. Leonard. Towards human–robot teams: Model-based analysis of human decision making in two-alternative choice tasks with social feedback. *Proceedings of the IEEE*, 100(3):751–775, March 2012.

[11] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[12] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2019.

[13] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[14] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687, Sep. 2015.

[15] A. Wang, J. Lu, J. Cai, T. Cham, and G. Wang. Large-margin multi-modal deep learning for rgb-d object recognition. *IEEE Transactions on Multimedia*, 17(11):1887–1898, Nov 2015.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[17] Evan A Krause, Michael Zillich, Thomas Williams, and Matthias Scheutz. Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[18] Danijel Skočaj, Matej Kristan, Alen Vrečko, Marko Mahnič, Miroslav Janíček, Geert-Jan M Kruijff, Marc Hanheide, Nick Hawes, Thomas Keller, Michael Zillich, et al. A system for interactive learning in dialogue with a tutor. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3387–3394. IEEE, 2011.

[19] V. Berenz and S. Schaal. The playful software platform: Reactive programming for orchestrating robotic behavior. *IEEE Robotics Automation Magazine*, 25(3):49–60, 2018.

[20] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, Lorenzo Natale, and Ingegneria dei Sistemi. Teaching iCub to recognize objects using deep convolutional neural networks. *Proc. Work. Mach. Learning Interactive Syst*, pages 21–25, 2015.

[21] Mennatullah Siam, Chen Jiang, Steven Lu, Laura Petrich, Mahmoud Gamal, Mohamed Elhoseiny, and Martin Jägersand. Video segmentation using teacher-student adaptation in a human robot interaction (hri) setting. *2019 IEEE International Conference on Robotics and Automation (ICRA) 2019.*, abs/1810.07733, 2019.

[22] S Hamidreza Kasaei, Miguel Oliveira, Gi Hyun Lim, Luís Seabra Lopes, and Ana Maria Tomé. Interactive open-ended learning for 3d object recognition: An approach and experiments. *Journal of Intelligent & Robotic Systems*, 80(3-4):537–553, 2015.

[23] Xiaoyuan He, Ryo Kojima, and Osamu Hasegawa. Developmental word grounding through a growing neural network with a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):451–462, 2007.

[24] Sepehr Valipour, Camilo Perez, and Martin Jagersand. Incremental learning for robot perception through hri. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2772–2777. IEEE, 2017.

[25] Eren Erdal Aksoy, Minija Tamosiunaite, and Florentin Wörgötter. Model-free incremental learning of the semantics of manipulation actions. *Robotics and Autonomous Systems*, 71:118 – 133, 2015. Emerging Spatial Competences: From Machine Perception to Sensorimotor Intelligence.

[26] Marc Toussaint, Thibaut Munzer, Yoan Mollard, Li Yang Wu, Ngo Anh Vien, and Manuel Lopes. Relational activity processes for modeling concurrent cooperation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5505–5511. IEEE, 2016.

[27] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view RGB-D object dataset, May 2011.

[28] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *IEEE Int. Conf. on Robotics and Automation*, pages 509–516, 2014.

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[30] Argiro Vatakis and Katerina Pastra. A multimodal dataset of spontaneous speech and movement production on object affordances. *Scientific Data*, Jan 2016.

[31] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from RGBD images. *plan, activity, and intent recognition*, 64, 2011.

[32] Wenjuan Gong, Jordi Gonzalez, Joao Manuel RS Tavares, and F Xavier Roca. A new image dataset on human interactions. In *International Conference on Articulated Motion and Deformable Objects*, pages 204–209. Springer, 2012.

[33] D. Temel, J. Lee, and G. Alregib. Cure-or: Challenging unreal and real environments for object recognition. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 137–144, Dec 2018.

[34] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26, 2017.

[35] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[36] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017.

[37] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[38] You-Lu Xing, Fu-Rao Shen, Jin-Xi Zhao, Jing-Xin Pan, and Ah-Hwee Tan. Perception coordination network: A framework for online multi-modal concept ac-

quisition and binding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[40] Rong Xiao, Jicheng Wang, and Fayan Zhang. An approach to incremental svm learning algorithm. In *Proceedings 12th IEEE Internationals Conference on Tools with Artificial Intelligence. ICTAI 2000*, pages 268–273, Nov 2000.

[41] M Narasimha Murty and G Krishna. A hybrid clustering procedure for concentric and chain-like clusters. *International Journal of Computer & Information Sciences*, 10(6):397–412, 1981.

[42] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[43] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.

[44] Shen Furao and Osamu Hasegawa. An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, 19(1):90 – 106, 2006.

[45] Youlu Xing, Xiaofeng Shi, Furao Shen, Ke Zhou, and Jinxi Zhao. A self-organizing incremental neural network based on local distribution learning. *Neural Networks*, 84:143 – 160, 2016.

[46] Alexander Gepperth and Cem Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934, Oct 2016.

[47] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 321–328. IEEE, 2000.

[48] Arjan Gijsberts and Giorgio Metta. Real-time model learning using incremental sparse spectrum gaussian process regression. *Neural Networks*, 41:59–69, 2013.

[49] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *Trans. on Robotics*, 24(5):1027–1037, 2008.

[50] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[51] J. Rituerto, A. C. Murillo, and J. Kosecka. Label propagation in videos indoors with an incremental non-parametric model update. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2383–2389, Sept 2011.

[52] Pejman Iravani, Peter Hall, Daniel Beale, Cyril Charron, and Yulia Hicks. Visual object classification by robots, using on-line, self-supervised learning. In *IEEE Int. Conf. on Computer Vision Workshops*, pages 1092–1099, 2011.

[53] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3D object models using next best view manipulation planning. In *IEEE Int. Conf. on Robotics and Automation*, pages 5031–5037, 2011.

[54] Jivko Sinapov, Connor Schenck, and Alexander Stoytchev. Learning relational object categories using behavioral exploration and multimodal perception. In *IEEE Int. Conf. on Robotics and Automation*, pages 5691–5698, 2014.

[55] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *arXiv preprint arXiv:1604.03670*, 2016.

[56] K. Park, H. Lee, Y. Kim, and Z. Z. Bien. A steward robot for human-friendly human-machine interaction in a smart house environment. *IEEE Transactions on Automation Science and Engineering*, 5(1):21–25, Jan 2008.

[57] Ulrich Reiser, Christian Pascal Connette, Jan Fischer, Jens Kubacki, Alexander Bubeck, Florian Weisshardt, Theo Jacobs, Christopher Parlitz, Martin Hägele, and Alexander Verl. Care-o-bot® 3-creating a product vision for service robot applications by integrating design and technology. In *IROS*, volume 9, pages 1992–1998, 2009.

[58] J. Baraglia, M. Cakmak, Y. Nagai, R. Rao, and M. Asada. Initiative in robot assistance during collaborative task execution. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 67–74, March 2016.

[59] J. Dumora, F. Geffard, C. Bidard, N. A. Aspragathos, and P. Fraisse. Robot assistance selection for large object manipulation with a human. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1828–1833, Oct 2013.

[60] Raffaello Camoriano, Giulia Pasquale, Carlo Ciliberto, Lorenzo Natale, Lorenzo Rosasco, and Giorgio Metta. Incremental object recognition in robotics with extension to new classes in constant time. *arXiv preprint arXiv:1605.05045*, 2016.

[61] Natalia Lyubova, Serena Ivaldi, and David Filliat. From passive to interactive object learning and recognition through self-identification on a humanoid robot. *Autonomous Robots*, 40(1):33–57, 2016.

[62] Lili Nurliyana Abdullah and Shahrul Azman Mohd Noah. Integrating audio visual data for human action detection. In *Int. Conf. Computer Graphics, Imaging and Visualisation*, pages 242–246. IEEE, 2008.

[63] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[64] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[65] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[66] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[67] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[68] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[69] Sepehr MohaimenianPour and Richard Vaughan. Hands and faces, fast: Mono-camera user detection robust enough to directly control a uav in flight. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5224–5231, 2018.

[70] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[71] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[72] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[73] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[74] Xiaobo Li, Haohua Zhao, and Liqing Zhang. Recurrent retinanet: A video object detection model based on focal loss. In *International Conference on Neural Information Processing*, pages 499–508. Springer, 2018.

[75] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 840–849, 2019.

[76] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.

[77] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[78] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[79] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[80] Gerard Canal, Sergio Escalera, and Cecilio Angulo. A real-time human-robot interaction system based on gestures for assistive scenarios. *Computer Vision and Image Understanding*, 149:65–77, 2016.

[81] Sadi Rafsan, Safayet Arefin, AHM Mirza Rashedul Hasan, and Mohammed Moshiul Hoque. Design a human-robot interaction framework to detect household objects. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 973–978. IEEE, 2016.

[82] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[83] Joao Carreira and Cristian Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2011.

[84] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.

[85] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012.

[86] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.

[87] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Trans. on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[88] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[89] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[90] Fernand Meyer. Color image segmentation. In *Image Processing and its Applications, 1992., International Conference on*, pages 303–306. IET, 1992.

[91] Steven Bird and Edward Loper. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics, 2004.

[92] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[93] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[94] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[95] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[96] P. Azagra, F. Golemo, Y. Mollard, M. Lopes, J. Civera, and A. C. Murillo. A multimodal dataset for object model learning from natural human-robot interaction. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6134–6141, Sept 2017 `http://robots.unizar.es/IGLUdataset/`.

[97] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: incremental classifier and representation learning. In *CVPR*, 2017.

[98] Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Fine-grained continual learning. *arXiv preprint arXiv:1907.03799*, 2019.

[99] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018.

[100] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[101] Miguel Lagunes-Fortiz, Dima Damen, and Walterio Mayol-Cuevas. Learning discriminative embeddings for object recognition on-the-fly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2932–2938. IEEE, 2019.

[102] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

[103] Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. Toward training recurrent neural networks for lifelong learning. *Neural Computation*, 0(0):1–34, 0.

[104] German I. Parisi, Jun Tani, Cornelius Weber, and Stefan Wermter. Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *Frontiers in Neurorobotics*, 12:78, 2018.

[105] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[106] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

List of Figures

List of Tables