Incremental Learning of Object Models from Natural Human-Robot Interactions

Pablo Azagra, Javier Civera, Ana C. Murillo
DIIS-I3A. University of Zaragoza, Spain.

***Abstract*—In order to perform complex tasks in realistic human environments, robots need to be able to learn new concepts *in the wild*, incrementally, and through their interactions with humans. This paper presents an end-to-end pipeline to learn object models incrementally during the human-robot interaction.**

The pipeline we propose consists of three parts: (a) recognizing the interaction type, (b) detecting the object that the interaction is targeting, and (c) learning incrementally the models from data recorded by the robot sensors. Our main contributions lie in the target object detection, guided by the recognized interaction, and in the incremental object learning. The novelty of our approach is the focus on natural, heterogeneous and multimodal human-robot interactions to incrementally learn new object models. Throughout the paper we highlight the main challenges associated with this problem, such as high degree of occlusion and clutter, domain change, low resolution data and interaction ambiguity. Our work shows the benefits of using multi-view approaches and combining visual and language features, and our experimental results outperform standard baselines.

***Note to Practitioners*—This work was motivated by challenges in recognition tasks for dynamic and varying scenarios. Our approach learns to recognize new user interactions and objects. To do so, we use multimodal data from the user-robot interaction: visual data is used to learn the objects and speech is used to learn the label and help with the interaction type recognition. We use state-of-the-art deep learning models to segment the user and the objects in the scene. Our algorithm for incremental learning is based on a classic incremental clustering approach.**

The pipeline we propose works with all sensors mounted on the robot, so it allows mobility on the system. Our work uses data recorded from a Baxter robot, which enables the use of the manipulation arms in future steps, but it would work with any robot able to have the same sensors mounted. The sensors used are two RGB-D cameras and a microphone. The pipeline currently has high computational requirements to run the two deep learning based steps. We have tested it with a desktop computer including a GTX 1060 and 32GB of RAM.

***Index Terms*—Object recognition, incremental learning, multimodal data, human-robot interaction.**

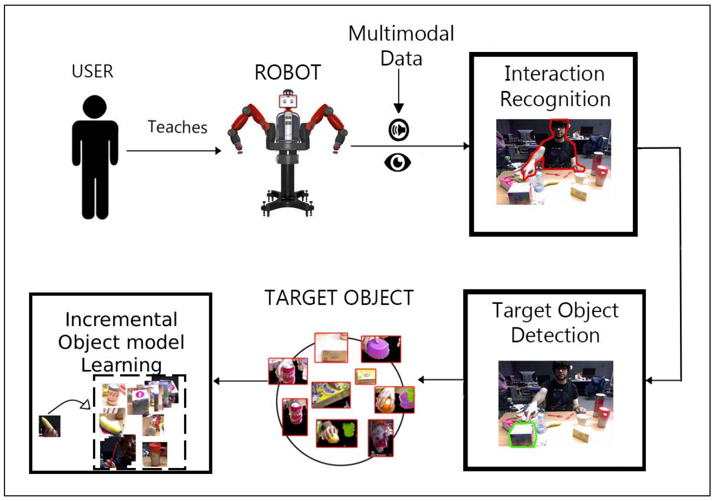


Fig. 1: Overview of our approach. A human user teaches a robot new objects through natural interactions (e.g., pointing to it). The robot recognizes the type of interaction from the multimodal recordings, finds the target object region on its camera views and updates the object model incrementally.

I. Introduction

We are witnessing a widespread adoption of service robotics in multiple aspects of our daily life and activities, such as household assistance devices or semi-autonomous cars. Visual perception is an essential component for these systems. In the last years, visual object recognition has been studied the most, typically with data-driven models trained offline, following the great success of deep learning based approaches for numerous recognition tasks. This paper addresses a particular case of object recognition, incremental learning of object models from natural human-robot interaction.

The need for incremental learning system comes from the fact that models trained offline on large generic datasets cannot, in general, address some challenges of using home environment real data. One example is the long-tail distribution, i.e., objects that appear rarely and for which few or none training samples exist in generic datasets. Another example 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 cases, robotic perception should have lifelong learning components.

One essential aspect to learn or update world models, affordances, and capabilities is a comfortable and intuitive human-robot interaction. This interaction should happen in a natural manner requiring the minimum effort for the human user. In our believe, the best interaction would be natural language and gestures, similarly to how the user would teach something to another person.

Learning from this type of natural human-robot interactions poses many significant research challenges. Although offline learning (deep learning in particular) has shown an impressive performance, it typically requires copious amounts of data.

P. Azagra is a PhD student at the University of Zaragoza
J. Civera and A.C. Murillo are faculty at the University of Zaragoza

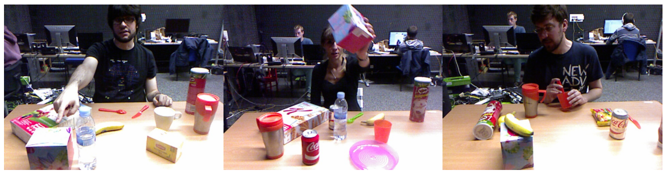

(a) *Point* (b) *Show* (c) *Speak*

Fig. 2: Interaction types from the *MHRI* dataset [1].

This amount of data is not available for all the relevant scenarios of human-robot interaction and other strategies are required. In addition to the object model that the robot has to learn or update, the correct identification of the interaction is a challenge on its own. The data recorded by the robot is typically multimodal, so a proper fusion has to be designed. Besides, images captured by robot cameras are usually very different from pictures acquired by humans: they typically contain objects of interest at low resolution, occluded by the human, only partially visible, or presenting motion blur. Such domain shift makes the use of standard datasets impractical, and might cause a bad performance of standard methods. Finally, online learning from such noisy data is also an open research problem.

To illustrate our goals, let us use the specific case of a robot helping a user in a kitchen. The robot needs to know the specific objects that are in the kitchen it operates. Some may have common shapes, and may be easy to recognize with models pre-trained offline (e.g., apple or mug), while others may have a particular appearance (e.g., a new wine brand or design cooking utensils). The user should be able to teach these latest unknown objects to the robot, in a natural manner, so the robot can identify them from then on. In order to learn interactively from a human user, as summarized in Fig. 1, the robot has to: 1) recognize the type of interaction the user is attempting; 2) segment image regions relevant to the object of interest according to this interaction; 3) create or update a model for the object of interest.

The main contribution of our work is a specific pipeline for learning new objects that is interaction-driven. As humans do, our pipeline ranks the relevance of scene parts depending on the gestures/speech of the person that is teaching. We focus on the three most natural interactions for humans (Fig. 2 shows one example of each of these interactions):

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

This paper builds on our previous works in [1] and [2]. The new contributions here are:

- An incremental learning algorithm, which is based on incremental clustering and k-Nearest Neighbour (k-NN).
- An improved interaction recognition module, which now uses a state-of-the-art CNN-based skeleton extractor [3] and our incremental learning algorithm supervised with human speech.
- A new strategy for the detection of object candidates, combining the segmentation from MaskRCNN [4] with the superpixel-based approach previously used in [1].
- A novel object recognition module that uses our incremental learning algorithm.
- A complete evaluation and thorough analysis of each module separately and then running the whole pipeline.

We evaluate our algorithms in two recent public datasets: *Core50* [5] and *MHRI* [1]. Our experimental validation shows that our approach outperforms standard baselines, and it also highlights the challenges associated with online model learning from natural Human Robot Interaction (HRI).

The rest of the paper is structured as follows. Sec. II details the related work. Sec. III provides a general overview of our approach. Sec. IV details the algorithm proposed for incremental learning. Sections V, VI and VII present, respectively, the details of the three modules of our pipeline: multimodal interaction recognition, target object detection and incremental learning of object models. Sec. VIII presents a separate analysis and evaluation for each module, while Sec. IX presents results running the whole pipeline. Sec. X concludes the work and discusses lines for future work.

II. Related work

Our proposal spans over several areas of interest in robotics, and we organize this section into such main areas.

A. *Robot Interaction*

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. [6] presents 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. [7], presents a robot that interacts with users to perform daily routines, and Reiser et al. [8], presents 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. [9] aims to decide if it is the robot or the human who should take the initiative in collaborative work, and Dumora et al. [10] presents 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. [11] and Krause et al. [12]. 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. [13] presents a work where the user can correct the robot when an object is not found using voice commands and pointing. In Siam et al. [14], HRI helps to improve the segmentation of a target object and to learn a better model. Other works focus on teaching actions like Aksoy et al. [15], which is able to incrementally learn semantic event chains (SECs) extracted from actions using human demonstration.

Very related to our work, Pascuale et al. [16] uses 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. [17] 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. [18] uses 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. [19] learns 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 [20] presents 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.

B. Object Recognition

Object recognition is a traditional research area in computer vision and robotics. Deep learning has made a breakthrough in this area reaching human-level performance [21] in some task. With well known models like ResNet [22] or Inception [23], deep learning approaches have provided a significant boost on related problems, such as object recognition, object segmentation [4] or object detection [24]. Also, their efficiency has been improved and networks like [25] or [26] are able to run on platforms with limited computational resources. All of these models, however, are trained offline and typically require a large amounts of labeled data.

Of particular relevance for our work are strategies that approach object model learning through incremental methods. Yao et al. [3] proposes an incremental learning method that continually updates an object detector and detection threshold, as the user interactively corrects annotations proposed by the system. Kuznetsova et al. [27] investigates incremental learning for object recognition in videos. Lee et al. [28] presents a SIFT-vocabulary that builds an incremental graph from a single image.

Recent recognition approaches focus on 3D object model, like Furrer et al. [29]. They use multiple observations of a 3D scene to incrementally create 3D models. We do not consider 3D object models, even though we use the depth channel, because our data contains a high level of clutter and small objects, which make such 3D modelling fail. Besides, in our data, object 3D shapes are not discriminative enough to be useful.

To study and evaluate different approaches for object recognition, the research community has released plenty of public datasets. For example [30], [31] are two well-known datasets targeting object recognition from *RGB-D* images. Also Temel et al. [32] shows objects in both real and unreal environments and different challenging conditions like underexposure or noise. However, most of the existing datasets focus on offline visual learning and RGB images, like COCO dataset [33] and Imagenet [34] . 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 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. Some works capture HRI but from another POV very different. Datasets like [35] or [36] capture human-robot interaction from a third-person POV. Other works capture the POV but the interaction is not with objects, like Vatakis et al. [37] which has an experiment with a first-person POV but with a screen interaction and a third-person POV experiment with objects.

Besides, the majority of the datasets lack other sensor data besides images like speech from the user, common in human-robot interactive scenarios. As detailed in the next section, the main dataset we use[1] is focused not only on capturing the user expression, but also on capturing the scene information (hands, arms, desk, objects, etc.) related to the object classes being taught to the robot.

C. Incremental Learning

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. [38], 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. [39] 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. [40] 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. [41]. 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 [42] use an offline learning algorithm as a base and incrementally modify their parameters. [43] 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. [44] presents an approach that combines the k-means algorithm with multilevel representation of the clusters. Likas et al. [45] 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. [46] 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. [47] presents an online unsupervised system with an incremental update of a Neural Network based on SOM (SOINN). Xing et al. [48] 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. [49] 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 [50] or inverse dynamics incremental learning in [51]. The same way, Angeli et al. [52] 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 [53], [54]. 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 [55], [56], [57]. 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.

III. Overview

This section presents an overview of our incremental object learning pipeline and the data we use to validate it.

A. Pipeline

The three main modules of our pipeline, illustrated in Fig. 3, are summarized next and detailed in the following sections. All the code to run our pipeline is available online[1].

Multimodal Incremental Interaction Recognition: An accurate identification of the human-robot interaction is a key aspect, as the extraction of object patches to train the object models depends on it. One of the main challenges here is that each user interacts differently, so we need to learn or adjust our models incrementally. This module combines skeleton detection [58] to guide the hand search and our incremental learning algorithm guided by natural user feedback.

Target Object Detection: For each interaction type we propose a specific algorithm to select the candidate image patches likely to contain the target object. We use two different camera views, a top (head-mounted) one that facilitates the segmentation, and a frontal one that provides closer object views. We also consider the user motion to select target object patches that minimize clutter and occlusion.

Incremental Object Model Learning: The candidate patches obtained from previous steps are used as training samples by our incremental learning algorithm. The category labels are obtained from the user speech data.

B. Data

Our approach is built to learn from natural human-robot interactions. In our scenario, the robot and the user share a workspace and the objects to learn are visible to both. We use the *Multi-modal Human-Robot Interaction dataset*, that contains data from the robot POV capturing the user teaching objects. We also use another public dataset, *Core50*, focused on incremental learning of objects.

TABLE I: Summary of *MHRI* 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*

Multi-modal Human-Robot Interaction (*MHRI*) dataset [1]: This is the main dataset used in our experimentation.

The *MHRI* dataset captures the most common natural interactions to teach object classes to a robot, namely: *Point*, *Show* and *Speak*. Table I summarizes its content: Recordings from 10 users; each user interacting with 10 of the object classes for each of the 3 types of interactions, for a total of 300 multimedia short clips. The aforementioned 10 objects per user were picked randomly, out of a pool of 22 available objects, and were used by such user in all his/her recordings.

Fig. 4 shows the cameras placement in the Baxter robot used for the acquisition. The *Frontal* RGB-D camera is mounted on

[1]https://sites.google.com/a/unizar.es/iglu_mhri/

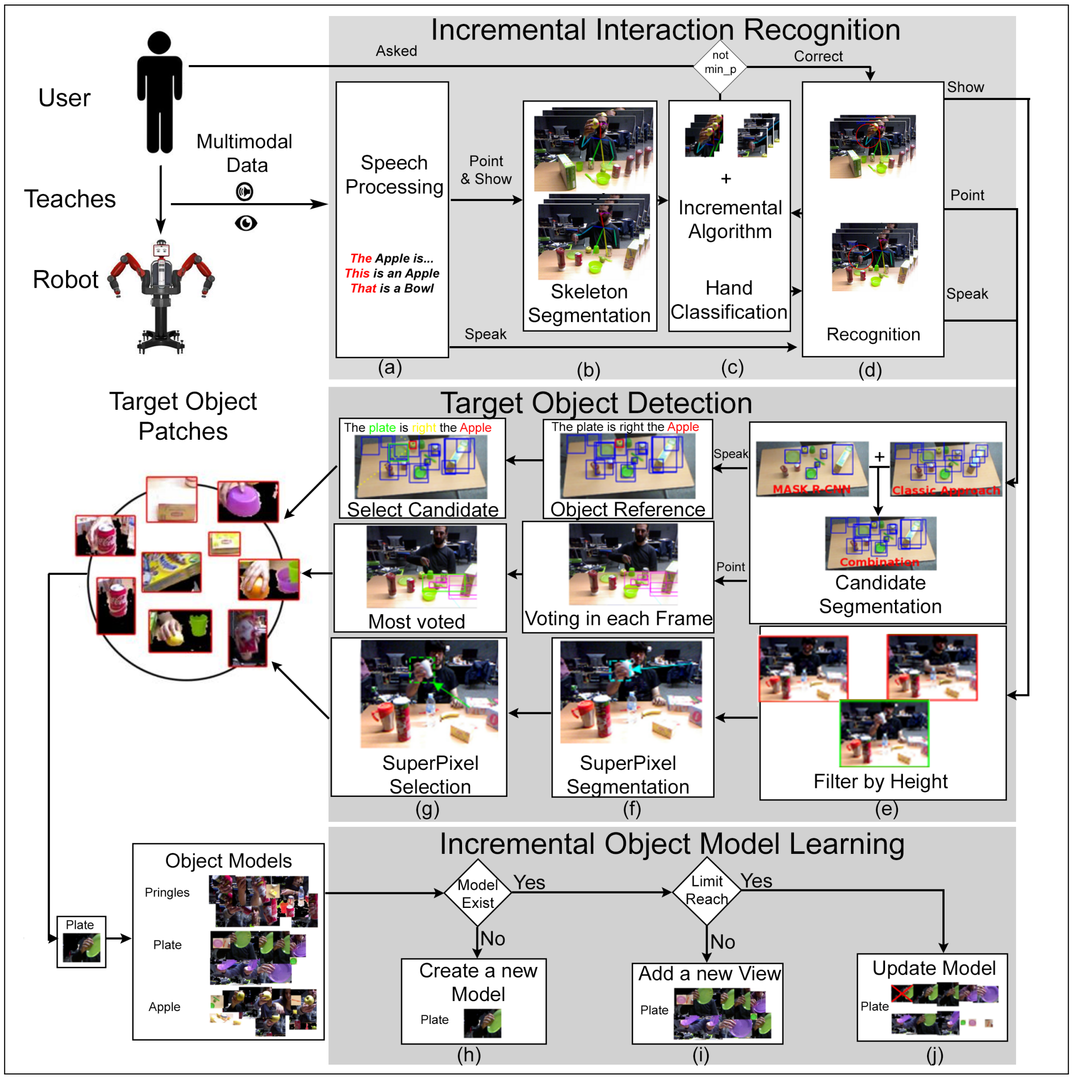


Fig. 3: Proposed interactive and incremental learning pipeline. A user teaches an object to a robot using one of the considered interactions: *Show*, *Point* or *Speak*. The multimodal data recorded by the robot is the input to our pipeline. (a) to (d) are the interaction recognition process steps. (e) to (g) summarize the target object extraction algorithm, that is dependent on the interaction type recognized. Finally, (h) to (j) are the stages of the incremental object model learning, using object patches obtained in the previous steps.

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. The annotations include 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. Each frame is labeled with the type of interaction.

Core50 dataset [5]: To further evaluate our incremental learning algorithm, we also use the *Core50* dataset. It focuses on the incremental learning of both instances and classes, and consists of 50 instances (10 categories) in 11 different scenarios (indoor and outdoor). Each video shows an object in a hand. The hand rotates to show different views of the object.

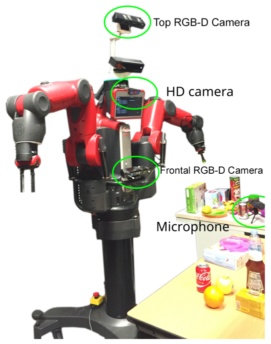


Fig. 4: Baxter robot setup used to acquire the dataset. The three cameras and the microphone locations are highlighted.

IV. Incremental Learning

Our online learning algorithm 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 $\mathcal{C} = \{C_1, \ldots, C_l, \ldots, C_N\}$, where C_l represents the set of clusters class l, $C_l = \{C_l^1, \ldots, C_l^j, \ldots, C_l^s\}$.

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. Figs. 3(h) to (j) show an example of the possibilities when a new training sample is given to the incremental learning. 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.

A. 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 Fig 3(h), 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 Fig 3(j), two things 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.

$$\begin{gathered} \{\hat{x}, \hat{y}\} = argmin_{x,y}\{D_B(C_l^x, C_l^y)\} \ni x \neq y \\ \{\hat{z}, \hat{w}\} = argmax_{w,z}\{D_B(C_l^w, C_l^z)\} \ni z \neq w \end{gathered} \quad (1)$$

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.

$$\hat{w} = argmin_w \tau_l^w \quad (2)$$

In addition to the approach to update the model described above, we tested several simple baselines (*random* and *always similarity*, where the merging is always with the closest clusters) as alternative criteria for the cluster reorganization. The proposed method offered the best performance, with an object recognition accuracy of 13% against 11% and 10% of *random* and *always similarity* respectively. Our 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). Since our application data is likely to contain significant noise and have a different distribution than typical public datasets, we do not find benefits pre-training our models on such data.

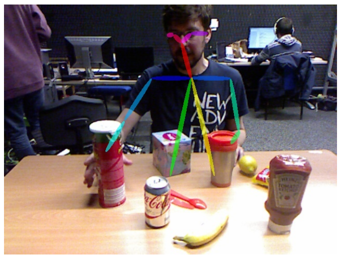
Algorithm 1 Incremental learning algorithm

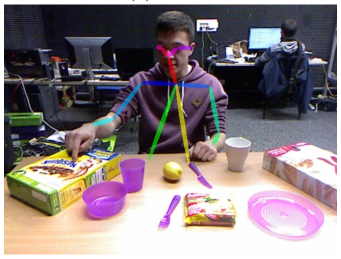
```
1:  procedure INC.TRAIN($e$,$l$)
2:      $C_l^e$ = Create_Cluster($e$,$l$)
3:      $C_l$.add($C_l^e$)
4:      if $l$ not in $system$.Labels then
5:          $system$.add_label($l$)
6:      else
7:          if $C_l$.is_full($k$) then
8:              if $system$.updates % $n_{updates} \neq 0$ then
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
```

B. Classification of new samples

A new sample is assigned to the existing classes following a k-Nearest Neighbor (k-NN) approach, following eq. 4. We compute the distance from the descriptor e of the new sample to all the cluster centroids in our model (C), sort them and obtain the k-top clusters ($x_{0:k}$ in the eq. 4). Each existing cluster, x, has an object 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.

(a) *Show*


(b) *Show*


(c) *Point*

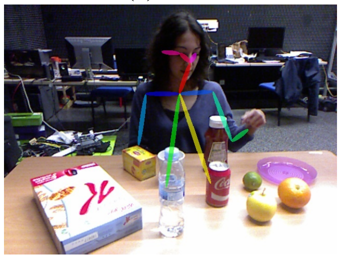

(d) *Point*

Fig. 5: Examples of skeleton detections. Notice that, in 5b, the hand joint is predicted even when it is occluded.

$$x_{0:k} = \text{sort}(D_B(e,\ C))[0:k] \tag{3}$$

$$l^{\hat{x}} = Mode(l^{x_{0:k}}) \tag{4}$$

V. Multimodal Incremental Interaction Recognition Module

This section details our interaction recognition module. It uses visual and language features related to the analysis of the user hand and the keywords in the user speech, respectively.

Classifying a person's activity only from visual data is challenging. Several works, e.g. [59], show that the combination of language and vision leads to substantial improvements. In our previous work [60] we demonstrated that including *Speak* interactions to train the models obtains better accuracy than using only *Point* and *Show* ones.

A. Visual analysis of the user

To recognize the user interaction, we first analyze the possible users in the robot field of view, and focus on the main user hand.

We use a CNN-based approach by Cao et al [58] that identifies all skeleton joints for all the people in the image.

It could be replaced by any other skeleton segmentation strategy, but we chose this particular approach for several reasons. It shows good performance in our images, even with considerable occlusions of the user, and its computational load is reasonably low. Fig. 5 shows several examples of estimated skeletons where its accuracy can be appreciated.

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

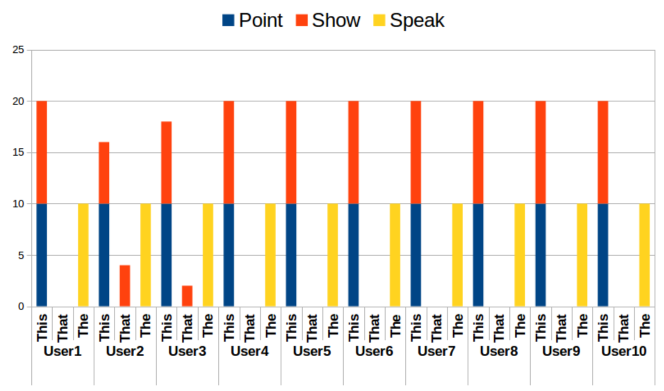


Fig. 6: Language feature occurrences in all recordings, per type of interaction and per user.

B. Multimodal features

Language Features: From the user speech we extract the label of the object that the user is referring to, as well as information to guide the interaction recognition. We use a simple language feature consisting of the first word of the user speech. In our dataset, by construction, this word is either *this* or *that* for *Point* and *Show* interactions and any other word for the more heterogeneous *Speak* interaction. This feature is not discriminative enough to separate the three interaction classes, as it can be seen in Fig. 6. It can be used to separate *Speak* interactions, but it is not helpful to differentiate between *Point* and *Show*. However, separating *Speak* is particularly valuable, as there are no specific visual patterns associated with this interaction.

Visual Features: We compute *Histogram of Gradients* (HOG) [61] in the depth channel of the hand patch.

C. Incremental 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, a general gesture classifier trained offline does not reach a reasonable performance.

In our experiments we have trained several baselines, specifically common SVM classifiers, and well-known classification CNNs such as Inception [23]. Probably 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 accuracies. Instead, we use the incremental learning algorithm described in section IV . It uses interactive supervision from the user if the classifier output is not conclusive.

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.

Algorithm 2 Incremental Interaction Recognition.

```
 1: $min\_prob = 85\%$
 2: $videos\_processed = 0$
 3: $n\_videos = 4$
 4: $min\_dist = 0.56$
 5: function EXTRACT_DESCRIPTOR(Frame_RGB)
 6:     Skeleton = obtain_skeleton(Frame_RGB.Front)
 7:     Hand_patch = Crop_Hand(Frame_RGB.Front,Skeleton.wrist)
 8:     Descriptor = Calculate_HOG(Hand_patch)
 9:     return Descriptor
10: end function
11: function TRAIN_INCREMENTAL(Video_RGBD)
12:     Interaction = Ask_User()
13:     for each Frame_RGB in Video_RGBD do
14:         Descriptor = Extract_Descriptor(Frame_RGB)
15:         Inc.Train(Descriptor, Interaction)                ▷ See Alg. 1
16:     end for
17:     return Interaction
18: end function
19: function INTERACTION RECOGNITION(Video_RGBD,speech)
20:     if speech.get_first_word() $\neq$ ("This" | "That") then
21:         return "Speak"
22:     else
23:         if videos_processed < $n\_vids$ then
24:             videos_processed += 1
25:             Interaction = Train_Incremental(Video_RGBD)
26:             return Interaction
27:         else
28:             videos_processed += 1
29:             Votes = []
30:             for each Frame_RGB in Video_RGBD do
31:                 Descriptor = Extract_Descriptor(Frame_RGB)
32:                 Class, Distance = Inc.Test(Descriptor)
33:                 if Distance < min_distance then
34:                     Votes.add_vote(Class)
35:                 end if
36:             end for
37:             Interaction,Confidence = Process_Results(Votes)
38:             if Confidence > min_prob then
39:                 return Interaction
40:             else
41:                 return Train_Incremental(Video_RGBD)
42:             end if
43:         end if
44:     end if
45: end function
```

VI. TARGET OBJECT DETECTION MODULE

This module extracts image patches that are likely to contain the target object we want to learn from the user interaction. Depending on the type of interaction recognized with the previous module (*Show*, *Point*, *Speak*), we propose three different strategies to segment the target object patches.

A. 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 [62] superpixels. Our algorithm selects superpixels likely to contain relevant information, i.e., large overlap with the *hand* patch and small distance between the superpixel center and the *hand* patch center.

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
```

B. 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 4.

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 [4] 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 Fig. 12), it misses important candidates. In the second approach, a superpixel segmentation [63] is used to remove table pixels. Then we apply Otsu's thresholding with the Watershed algorithm (as described in Meyer et al. [64]), 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 5.

Hand pointing direction estimation. Fig. 7 shows several examples of the output of our pointing direction estimation algorithm. Hand contours are extracted using a Canny edge

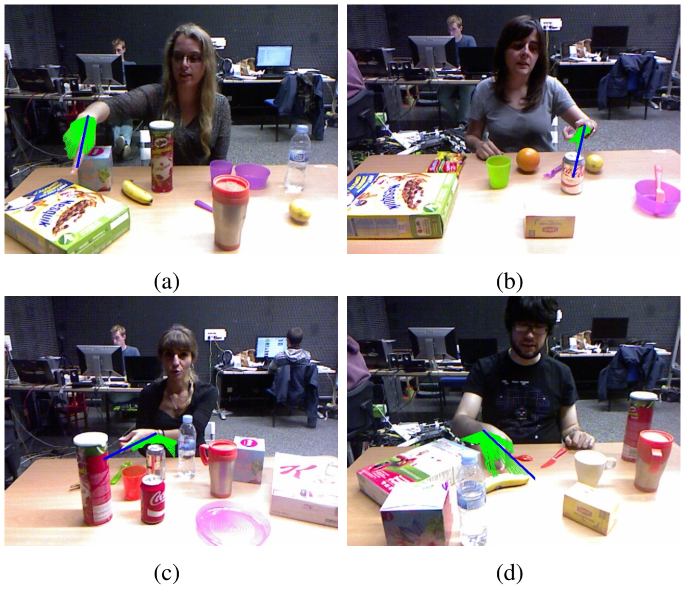


Fig. 7: Examples for hand detection and pointing direction estimation. The green regions show equally-distributed possible directions. The blue line is the estimated direction. (a) to (c) show common correct cases; (b) shows an example where the direction is correct even for an uncommon hand pose. (d) shows a failure case, the direction is incorrect due to similar depths in the hand and the table. (best viewed in color)

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 Fig. 3(f). To evaluate which candidates are more promising, our algorithm computes the following score:

$$Score = \sum_{Frames} \frac{Intersect(Hand\_direction, Candidate)}{Diagonal(Candidate)}, \quad (5)$$

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 normalize by the size of each candidate. The candidate with 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.

C. 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 [65] can extract the target object name, reference objects and their relative positions. Table II shows two examples of the speech processed. Algorithm 6 describes our strategy for this interaction type and the main ideas are discussed next.

Algorithm 4 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. 5
 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 5 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
```

TABLE II: 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

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 using the corners of the reference patch, the corner of the images and the search direction. The selected

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. 5
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
```

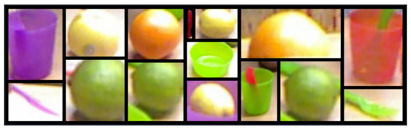


Fig. 8: Sample objects patches from human-robot interaction. The object views are typically low-resolution patches where standard keypoints/descriptors give low performance.

candidate is the closest to the reference patch within the target area, similar to the *Point* interaction.

VII. Incremental Object Model Learning Module

We apply our incremental learning approach, Sec. IV, to learn object models from image patches. For an illustration of the typical patches we have available in the considered robotic settings, Fig. 8 shows a few examples of [1]. 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 [66], as they provide a good compromise between accuracy, efficiency and number of keypoints. The BoW 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 [30] 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:

$$BoW_{ORB} = [t_1, ..., t_w, ...t_{1000}] \quad ; \quad t_w = \frac{n_{\mathrm{wp}}}{n_k} \ , \qquad (6)$$

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:

$$HC_{RGB} = [H_r \ H_g \ H_b]. \qquad (7)$$

SIFT keypoints: We obtain SIFT keypoints and their associated descriptors [67] as

$$SIFT = \{s_1, s_2, s_3, ..., s_n\}, \qquad (8)$$

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.

CNN features: We use the flattened output of the last GAP (Global average pool) layer from ResNet50 [22].

$$ResNet50 = ResNet50(patch).GAP \qquad (9)$$

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. VIII-C), but its performance decreases when evaluating the whole pipeline (see Sec. IX).

VIII. Module Validation

This section evaluates the performance of each module of the proposed pipeline independently.

A. Multimodal Incremental Interaction Recognition

This experiment evaluates the accuracy of the incremental interaction recognition module, described in Sec. V. We run Algorithm 2 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 parameters we 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 9 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 correct interaction. *Question* means the confidence of the classification

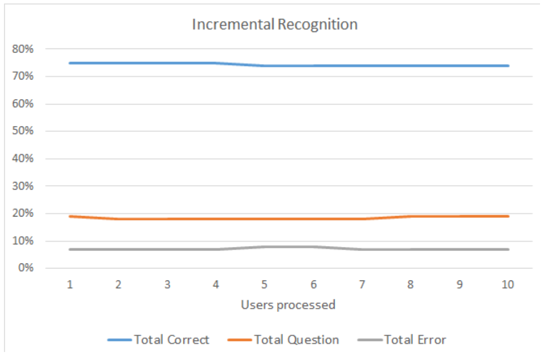
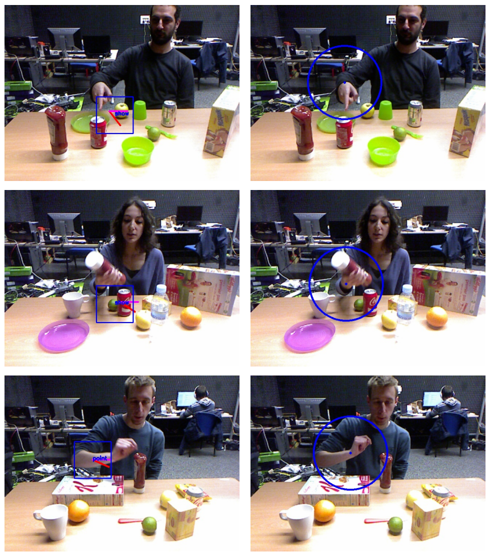


Fig. 9: Interaction recognition results.

TABLE III: 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



(a) Baseline [1] (b) Skeleton-guided (ours)

Fig. 10: Interaction recognition sample results.

output is below min_prob and the algorithm needs to ask the user for clarification.

Looking into the results per class, Table III shows that *Point* and *Show* achieve similar accuracy. Compared to the baseline version of this module in [1], we observe an improvement (13% error rate in [1] versus 7% error rate here) and more balanced results (14% and 22% error rate for *Point* and *Show* in [1] vs 11% and 10% error rates here). The key difference between [1] and the algorithm in this paper is the skeleton detection. Thanks to this, we improve from an average of 33 valid frames found per video to an average of 47. Besides, the hand patches extracted now present higher quality, as shown in the examples in Fig. 10. This also benefits the general performance of the pipeline because the hand patch is used in following steps.

B. Target Object Detection

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. Fig. 11 shows several correct and incorrect examples of the target object segmentation and Table IV 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 approach and a superpixel (SPX)-based one, both explained in Sec VI-B. Fig. 12 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 Fig. 13(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 Fig. 14(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

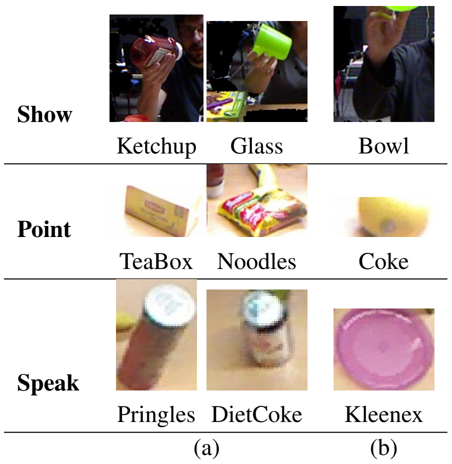


Fig. 11: Target Object Detection sample results: (a) Correct segmentation; (b) Incorrect segmentation

TABLE IV: 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

kept, 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 Fig. 13(b)), providing very noisy training data.

C. Incremental Object Learning

This section analyzes the performance of the incremental learning module described in Sec. VII. 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 [5]. 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 Fig. 14(a) for examples of such patches).

Object Patch Descriptors: We evaluate several **patch descriptors**, as detailed in Sec. VII: Color Histogram (HC_{RGB}), Bag of Words using ORB descriptors (BoW_{ORB}), the output of layer GAP of pre-trained ResNet [22] ($ResNet50$), 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 Fig. 15 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% [2] reported in [5] (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 Fig. 16, 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 [68], left-right consistency and Lowe's nearest neighbour ratio test [67]. Since *MHRI* is around five times smaller than *Core50*, we configured the size-limit to $10, 20, 30$, and all data. Table V(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 Fig. 14. Notice

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

TABLE V: 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
$SVM + HC_{RGB}$ [1]	34,8
Inception-based [69]	**59,3**

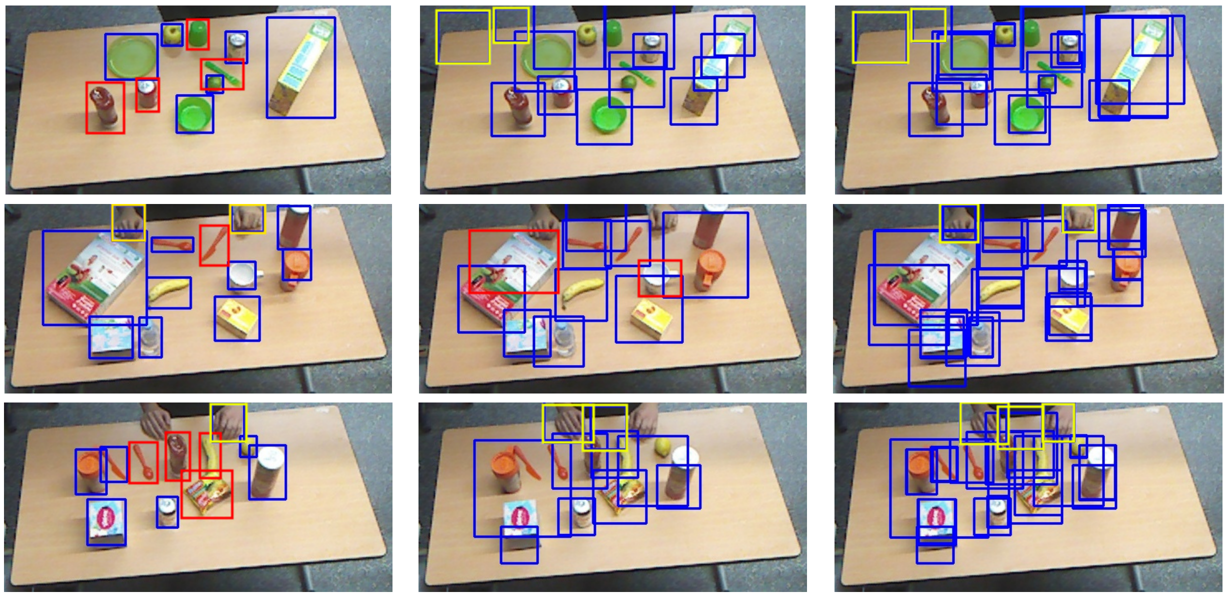
that our objects have distinctive colors and poor texture, and hence descriptors based on keypoints will perform poorly.

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. Fig. 17 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.

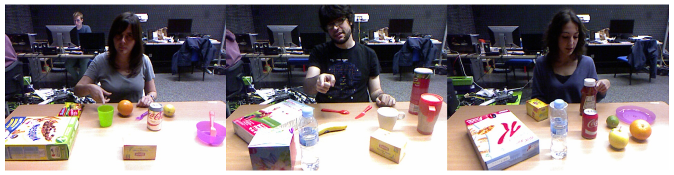
*Discussion of related **offline** and **online** baselines:* The object recognition performance of our incremental model is compared with previous work on the target scenario [1], which used an offline recognition pipeline, and with two standard offline strategies for object recognition:

- $SVM+HC_{RGB}$ [1]: we consider the best result obtained on our earlier work. It uses HC_{RGB} as a descriptor and

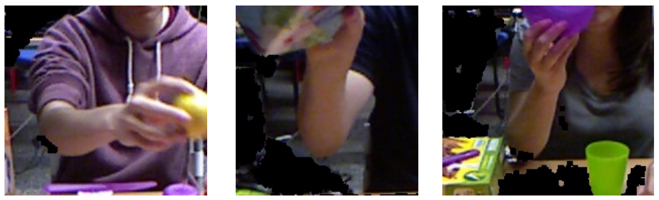


(a) DL [4] (b) SPX (c) DL + SPX (Ours)

Fig. 12: 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.

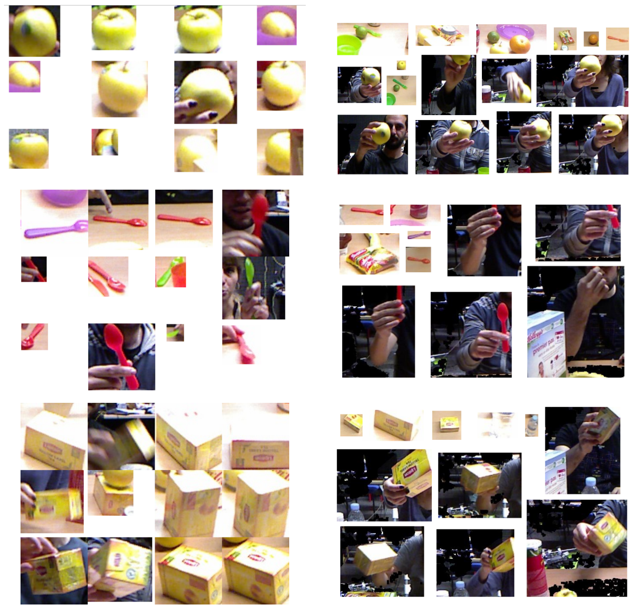
Fig. 13: Examples of two common difficult/failure cases for Target Object Segmentation in *Point* and *Show* videos.

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 [69], 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) [42] applied to SVM[3], which updates the support vectors with each step, and with an incremental SoftMax Regression. We tested the PASVM with

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



(a) (b)

Fig. 14: Examples of (a) *Manually Cropped* and (b) *Automatically segmented* patches from three objects in the *MHRI* data.

HC_{RGB} and ResNet50 descriptors and parameter $C = 2$ and the SoftMax with HC_{RGB}.

Table V(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

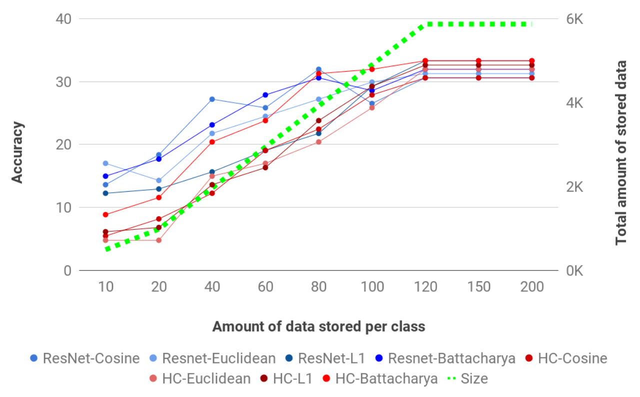


Fig. 15: *Core50* experiments processing all data.

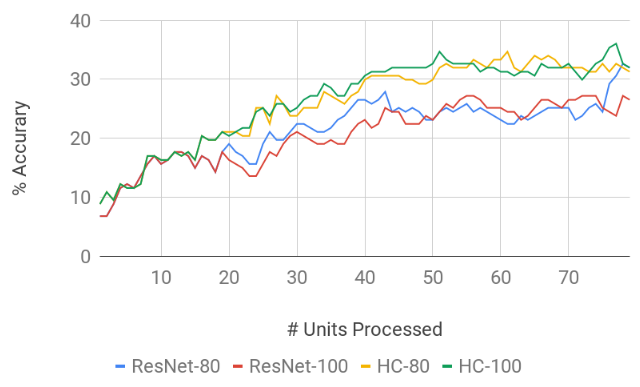


Fig. 16: *Core50* experiments, incremental results.

to limit the cluster size does not harm the performance. Our results are also improve over $SVM + HC_{RGB}$ [1], a remarkable result taking into account that our current approach is incremental, while $SVM + HC_{RGB}$ [1] used offline training. Amont 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.

IX. End-to-End pipeline validation

This section presents an integration experiment to validate the correct behavior and performance of all the modules running together, using the *MHRI* dataset. To our knowledge, this dataset is the only one that contains both interaction and object recognition in the wild. As discussed in more detail in previous work [1], classifiers are sensitive to domain change, so learning models using data from different domains does not necessarily boost the performance.

This experiment uses target object patches, extracted automatically from the interactions, for training and testing. Fig. 14 shows illustrative examples of these automatic patches quality, which is considerably worse than manual patches quality. This

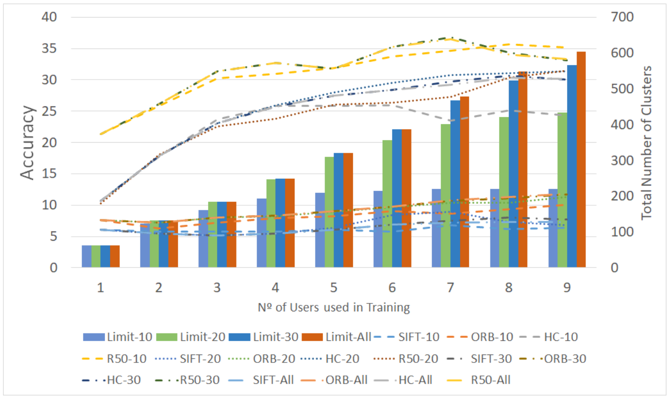


Fig. 17: Accuracy and number of clusters of our incremental learning algorithm as the number of users increases. Descriptors reported: HC_{RGB}, FC7, SIFT, BoW_{ORB}. Model sizes evaluated: 10, 20, 30 and All.

increases significantly the challenge of this experiment but brings it closer to a system running from scratch in the wild.

A. Incremental k-NN

The incremental algorithm we propose is again run with a 10-fold cross-validation, where each fold corresponds to a user, but now uses the more challenging automatically segmented patches. The difficulty 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 section, and the HC_{RGB} descriptor, with a more robust performance in the *Core50* data.

Table VI and Table VII 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 Fig. 18. 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 IX, the size of both models is small compared to other baselines.

Fig. 19 shows several examples of objects recognized after the pipeline has been trained with the 9 users. The algorithm has been able to learn models from interactions and identify them correctly on the table. For the class *cereal*, the descriptor is not invariant to rotation, it needs examples in different positions. We can see from the examples that our algorithm works even if the object has instances from different colors (like *knife*, *apple* or *lime*). For a quantitative evaluation of the results, Fig. 20 shows the evolution of the F_1 score (averaged over 10 folds) with the amount of training data. It can be observed that the F_1 score remains quite stable after the

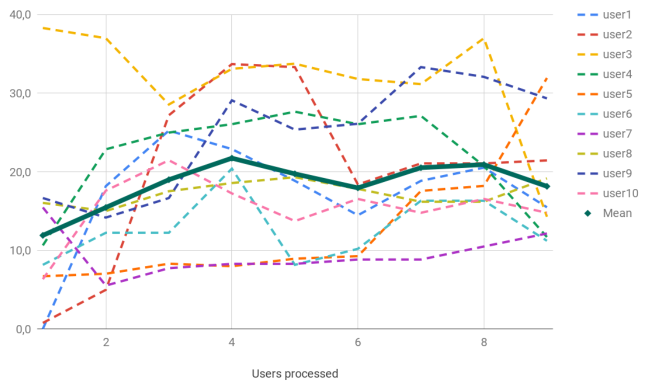


Fig. 18: Incremental learning accuracy as more data (from more users) is used for training. Dashed lines are per-fold results, solid line is the average. Table VII contains the numerical results for this graph. (best viewed in color)

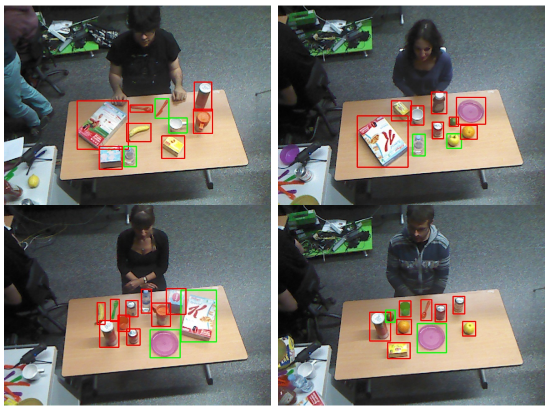


Fig. 19: Examples of the recognition results after the incremental learning was run. The objects in green are correctly labeled. Best viewed in color.

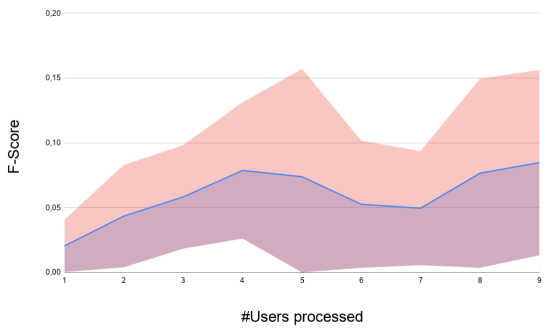


Fig. 20: Evolution of the F_1 score with the amount of training data over 10 folds. The blue line stands for its average, and the colored area for its standard deviation.

fourth user, which is consistent with previous experiments. We can also see a high deviation between folds, due to certain particularly challenging object classes, as it was discussed before in more detail.

TABLE VI: 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

TABLE VII: Object Recognition Accuracy ($ResNet50$, 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 VIII: 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

B. Comparison with incremental and offline baselines

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

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

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

	Accuracy	**STD**	**Size**
Previous Work (offline) [1]:			
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

work presenting the dataset [1]. 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 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 IX 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. Fig. 14 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 [1] 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.

X. Conclusions

This paper presents a complete end-to-end approach for incremental learning of object models from natural and multimodal Human-Robot Interaction.

The pipeline presented in this work is built on our preliminary version of the individual modules. The contributions here with respect to those earlier results include the incremental strategy, the full integration of the pipeline and a more thorough evaluation and discussion on the challenges in the problem considered.

The novelty of our approach relies on the integration of several modules that facilitate the use of natural interaction data. Our main conclusions about each of them are the following. Firstly, recognizing the type of interaction is a critical step to be able to find the relevant regions in the image, as it guides the object patch segmentation used to train the object models. We have attempted different strategies to achieve it and show that it needs to get user feedback to robustly adapt to new users and environments.

Our results also show how the interaction recognition benefits from the use of multimodal data (language and vision). Secondly, the different target object detection strategies proposed successfully extract patches containing the objects targeted. Our performance evaluation highlights significant challenges of the natural interaction settings with respect to other settings, where the data obtained is less noisy. Third, our evaluation of the incremental object learning module also highlights the challenges of our targeted incremental learning in natural or in-the-wild scenarios.

The performance of our incremental algorithm is close to common offline approaches and outperforms the baseline in the *Core50* dataset, while storing significantly smaller amounts of data.

Finally, we have demonstrated the behaviour of the full pipeline with a real use case application and analyzed the main insights and conclusions. As it could be expected, the domain change is critical for the learning step, and it is impractical to use other datasets in a naïve manner. Besides, although our approach shows a reasonable performance, there are still considerable challenges in the target object detection and incremental model learning. Our experiments show that object segmentation strongly affects the learning module performance, justifying the relevance of the topic for future research. We believe the most promising lines for future work are increasing the accuracy of the hand direction estimation in the *Point* videos, or other possibilities to improve the target object segmentation, and the exploration of more sophisticated incremental learning methods, particularly those that are robust to noisy data.

Acknowledgments

The authors would like to thank NVIDIA Corporation for the donation of a Titan Xp GPU used in this work. This research has been partially funded by the European Union (CHIST-ERA IGLU), the Spanish Government, FEDER/Ministerio de Ciencia, Innovación y Universidades - Agencia Estatal de Investigación under grants PGC2018-

096367-B-I00, PGC2018-098817-A-I00 and RTC-2017-6421-7 and the Aragón regional government (DGA T45 17R/FSE).

References

[1] 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)*, Sept 2017 http://robots.unizar.es/IGLUdataset/, pp. 6134–6141.

[2] P. Azagra, J. Civera, and A. C. Murillo, "Finding regions of interest from multimodal human-robot interactions," in *Proc. GLU 2017 International Workshop on Grounding Language Understanding*, 2017, pp. 73–77.

[3] A. Yao, J. Gall, C. Leistner, and L. Van Gool, "Interactive object detection," in *Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3242–3249.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[5] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Conference on Robot Learning*, 2017, pp. 17–26.

[6] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[7] 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*, vol. 5, no. 1, pp. 21–25, Jan 2008.

[8] U. Reiser, C. P. Connette, J. Fischer, J. Kubacki, A. Bubeck, F. Weisshardt, T. Jacobs, C. Parlitz, M. Hägele, and A. Verl, "Care-o-bot® 3-creating a product vision for service robot applications by integrating design and technology." in *Intelligent Robots and Systems (IROS), 2009 IEEE/RSJ International Conference on*, vol. 9, 2009, pp. 1992–1998.

[9] 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)*, March 2016, pp. 67–74.

[10] 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*, Oct 2013, pp. 1828–1833.

[11] D. Skočaj, M. Kristan, A. Vrečko, M. Mahnič, M. Janíček, G.-J. M. Kruijff, M. Hanheide, N. Hawes, T. Keller, M. 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*. IEEE, 2011, pp. 3387–3394.

[12] E. A. Krause, M. Zillich, T. Williams, and M. 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.

[13] S. Valipour, C. Perez, and M. Jagersand, "Incremental learning for robot perception through hri," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 2772–2777.

[14] M. Siam, C. Jiang, S. Lu, L. Petrich, M. Gamal, M. Elhoseiny, and M. Jägersand, "Video segmentation using teacher-student adaptation in a human robot interaction (hri) setting," *2019 IEEE International Conference on Robotics and Automation (ICRA) 2019.*, vol. abs/1810.07733, 2019.

[15] E. E. Aksoy, M. Tamosiunaite, and F. Wörgötter, "Model-free incremental learning of the semantics of manipulation actions," *Robotics and Autonomous Systems*, vol. 71, pp. 118 – 133, 2015, emerging Spatial Competences: From Machine Perception to Sensorimotor Intelligence.

[16] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, L. Natale, and I. dei Sistemi, "Teaching iCub to recognize objects using deep convolutional neural networks," *Proc. Work. Mach. Learning Interactive Syst*, pp. 21–25, 2015.

[17] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta, "Incremental object recognition in robotics with extension to new classes in constant time," *arXiv preprint arXiv:1605.05045*, 2016.

[18] S. H. Kasaei, M. Oliveira, G. H. Lim, L. S. Lopes, and A. M. Tomé, "Interactive open-ended learning for 3d object recognition: An approach and experiments," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 3-4, pp. 537–553, 2015.

[19] N. Lyubova, S. Ivaldi, and D. Filliat, "From passive to interactive object learning and recognition through self-identification on a humanoid robot," *Autonomous Robots*, vol. 40, no. 1, pp. 33–57, 2016.

[20] X. He, R. Kojima, and O. Hasegawa, "Developmental word grounding through a growing neural network with a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 451–462, 2007.

[21] R. Geirhos, D. H. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, "Comparing deep neural networks against humans: object recognition when the signal gets weaker," *arXiv preprint arXiv:1706.06969*, 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[25] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[26] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[27] A. Kuznetsova, S. J. Hwang, B. Rosenhahn, and L. Sigal, "Expanding object detector's horizon: incremental learning framework for object detection in videos," in *Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 28–36.

[28] S. Lee, J. Lim, and I. H. Suh, "Incremental learning from a single seed image for object detection," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1905–1912.

[29] F. Furrer, T. Novkovic, M. Fehr, A. Gawel, M. Grinvald, T. Sattler, R. Siegwart, and J. Nieto, "Incremental object database: Building 3d models from multiple partial observations," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 6835–6842.

[30] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," May 2011.

[31] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 509–516.

[32] 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)*, Dec 2018, pp. 137–144.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[35] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Human activity detection from RGBD images." *plan, activity, and intent recognition*, vol. 64, 2011.

[36] W. Gong, J. Gonzàlez, J. M. R. S. Tavares, and F. X. Roca, *A New Image Dataset on Human Interactions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[37] A. Vatakis and K. Pastra, "A multimodal dataset of spontaneous speech and movement production on object affordances," *Scientific Data*, Jan 2016.

[38] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[39] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1320–1328.

[40] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.
[41] Y.-L. Xing, F.-R. Shen, J.-X. Zhao, J.-X. Pan, and A.-H. Tan, "Perception coordination network: A framework for online multi-modal concept acquisition and binding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[42] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, no. Mar, pp. 551–585, 2006.
[43] 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*, Nov 2000, pp. 268–273.
[44] M. N. Murty and G. Krishna, "A hybrid clustering procedure for concentric and chain-like clusters," *International Journal of Computer & Information Sciences*, vol. 10, no. 6, pp. 397–412, 1981.
[45] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.
[46] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2624–2637, 2013.
[47] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Networks*, vol. 19, no. 1, pp. 90 – 106, 2006.
[48] Y. Xing, X. Shi, F. Shen, K. Zhou, and J. Zhao, "A self-organizing incremental neural network based on local distribution learning," *Neural Networks*, vol. 84, pp. 143 – 160, 2016.
[49] A. Gepperth and C. Karaoguz, "A bio-inspired incremental learning architecture for applied perceptual problems," *Cognitive Computation*, vol. 8, no. 5, pp. 924–934, Oct 2016.
[50] S. Thrun, W. Burgard, and D. 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*, vol. 1. IEEE, 2000, pp. 321–328.
[51] A. Gijsberts and G. Metta, "Real-time model learning using incremental sparse spectrum gaussian process regression," *Neural Networks*, vol. 41, pp. 59–69, 2013.
[52] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *Trans. on Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.
[53] 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.
[54] 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*, Sept 2011, pp. 2383–2389.
[55] P. Iravani, P. Hall, D. Beale, C. Charron, and Y. Hicks, "Visual object classification by robots, using on-line, self-supervised learning," in *IEEE Int. Conf. on Computer Vision Workshops*, 2011, pp. 1092–1099.
[56] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3D object models using next best view manipulation planning," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 5031–5037.
[57] J. Sinapov, C. Schenck, and A. Stoytchev, "Learning relational object categories using behavioral exploration and multimodal perception," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 5691–5698.
[58] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.
[59] L. N. Abdullah and S. A. M. Noah, "Integrating audio visual data for human action detection," in *Int. Conf. Computer Graphics, Imaging and Visualisation*. IEEE, 2008, pp. 242–246.
[60] P. Azagra, J. Civera, and A. C. Murillo, "Finding regions of interest from multimodal human-robot interactions."
[61] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2005, pp. 886–893.
[62] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.
[63] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.
[64] F. Meyer, "Color image segmentation," in *Image Processing and its Applications, 1992., International Conference on*. IET, 1992, pp. 303–306.
[65] S. Bird and E. Loper, "Nltk: the natural language toolkit," in *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, 2004, p. 31.
[66] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.
[67] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov 2004.
[68] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.
[69] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

Pablo Azagra is a Ph.D. student at the Doctoral Program in Systems Engineering and Informatics, University of Zaragoza (Spain), in the Robotics group. His research interests include computer vision, machine learning and human robot interaction.

Javier Civera is an associate professor at the Department of Computer Science and Systems Engineering at Universidad de Zaragoza. His research interests include SLAM and computer vision for robotics and other embedded systems. Dr. Civera received his Ph.D degree in Computer Science from the University of Zaragoza, in Spain.

Ana C. Murillo is an associate professor at the Department of Computer Science and Systems Engineering at Universidad de Zaragoza. Her research interests include scene understanding and visual recognition for robotics and other embedded systems. Dr. Murillo received her Ph.D degree in Computer Science from the University of Zaragoza, in Spain.