Journal name
000(00):1–xx

DOI:doi number
http://mms.sagepub.com

Visual Recognition for Humanoid Robots

Sean Ryan Fanello[3] *, Carlo Ciliberto[1,2], Nicoletta Noceti[2], Giorgio Metta[1] and Francesca Odone[2]

[1] *iCub Facility, Istituto Italiano di Tecnologia, Genova, Italy,*

[2] *DIBRIS, Università degli Studi di Genova, Genova, Italy*

[3] *Microsoft Research Labs, Redmond, WA 98051, USA.*

Abstract

Visual perception is a fundamental component for most robotics systems operating in human environments. Specifically, visual recognition is a prerequisite to a large variety of tasks such as tracking, manipulation, human-robot interaction. As a consequence, the lack of successful recognition often becomes a bottleneck for the application of robotics system to real-world situations. In this paper we aim at improving the robot visual perception capabilities in a natural, human-like fashion, with a very limited amount of constraints to the acquisition scenario. In particular our goal is to build and analyze a learning system that can rapidly be re-trained in order to incorporate new evidence if available. To this purpose, we review the state-of-the-art coding-pooling pipelines for visual recognition and propose two modifications which allow us to improve the quality of the representation, while maintaining real-time performances: a coding scheme, Best Code Entries (BCE), and a new pooling operator, Mid-Level Classification Weights (MLCW). The former focuses entirely on sparsity and improves the stability and computational efficiency of the coding phase, the latter increases the discriminability of the visual representation, and therefore the overall recognition accuracy of the system, by exploiting data supervision. The proposed pipeline is assessed from a qualitative perspective on a Human-Robot Interaction (HRI) application on the iCub platform. Quantitative evaluation of the proposed system is performed both on in-house robotics data-sets (iCubWorld) and on established computer vision benchmarks (Caltech-256, PASCAL VOC 2007). As a byproduct of this work, we provide for the robotics community an implementation of the proposed visual recognition pipeline which can be used as perceptual layer for more complex robotics applications.

Keywords

Human-Robot Interaction, Learning and Interaction, Visual Recognition, Sparse Representations, iCub

* Corresponding author; e-mail: seanfa@microsoft.com

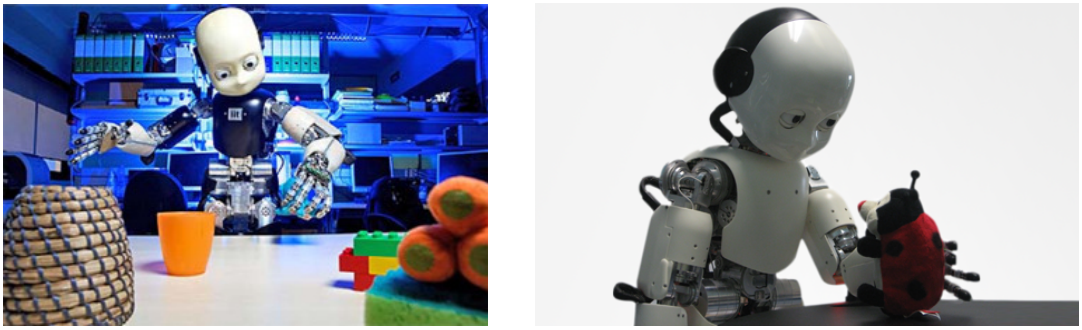


Fig. 1. Typical scenario with the iCub humanoid robot: objects are placed on a table, it has to learn their appearance and perform manipulation actions.

1. Introduction

Understanding the semantics of a scene is probably one of the most challenging tasks in artificial intelligence. The possible application domains are countless and include industry, communications, entertainment, robotics, just to name a few.

In particular, in *automation and cognitive robotics* visual recognition is a building block of very complex systems that include many other components — pose estimation, grasp, manipulation [16, 23, 39, 70]. This is possibly the reason why, so far, the task itself did not receive too much attention, and it was often addressed through methods which require a strict supervision in the training phase, including uncluttered views of the objects [50] or meta-data about object position or its orientation with respect to the camera [17].

Instead, in *computer vision*, visual recognition is a challenge *per sé*, at the essence of image understanding, and in recent years it stimulated important contributions in terms of appearance models learned by large dimensional datasets [20, 26, 34, 80] and possibly a limited amount of supervision [1, 13, 35, 45, 48, 49, 51, 64]. Such large data-sets are very ambitious in their scope, in that they aspire to represent the whole (or a large portion of the) visual world. In practice, since they are composed by an unordered set of images, albeit large, they mostly adhere to a content-based image retrieval scenario, rather than the actual image understanding.

In spite of the complementary challenges, cognitive robotics and computer vision mainly proceeded on independent tracks. It appears today that the mutual benefit one community can bring to the other has not been fully exploited. Indeed, visual recognition pipelines would provide flexibility and adaptivity to robotics systems, while robotics can be an ideal test bed for such models. As for the latter, while computer vision addresses the general question of "modeling the world" which is large and complex, far more complex than what a billion image unordered dataset could describe, in robotics the considered application, e.g. navigation, grasping, interaction, sets the boundaries of the world itself. In particular, the use of a humanoid robot represents the perfect scenario for designing and developing natural human-like visual recognition algorithms.

In this context, the main goal of the paper is to enhance the iCub human robot (Figure 1) with visual perception capabilities, enabling the implementation of complex behaviors, such as domestic assistance. A further aim of this work, is to evaluate the use of the iCub robot as a benchmark for computer vision tasks, considering visual recognition as a prototype of a computer vision problem.

Since visual recognition is a rather broad task, we better formulate our objectives by taking into account the main requirements of the application scenario we consider. Ideally, visual recognition should be based on a light and flexible training phase: to this purpose we consider a Human-Robot Interaction (HRI) framework, where the robot starts with no knowledge of the surrounding world and it tries to learn it one object at a time. Interestingly, within a HRI setting, human labeling and manual data acquisition are easily replaced by vocal and gesture interaction of the human supervisor with

the robot and are thus less costly. Therefore in this scenario, supervised learning is appropriate and acceptable. Also, we exploit the iCub attentive capabilities to obtain a localization of the object of interest (see Figure 2). A more controlled environment typical of robotics allows us to reduce biases and tune the amount of context variability [33].

Conversely, a prompt feedback is an important and demanding requirement of HRI: computational cost needs to be taken into account in the system design. Finally, the idea of having an artificial system that learns incrementally as a human does, speaks in favour of methods that can learn from a reasonably small set of examples and do not require a heavy training phase. For these reasons, with the current state of research, one-layer coding-pooling pipelines [48] appear today more appropriate than deep architectures [49].

Therefore we adopt a coding-pooling framework, as the one summarized in Figure 3, where we propose two main modifications to the standard pipelines increasing the performances still maintaining low computational costs: in the *coding stage*, we propose a new coding operator, Best Code Entries (BCE) which is based on the evidence that sparse representations increases the discriminative power of the data [30, 81], and allows us to obtain more stable solutions. In the *pooling stage* we exploit the data supervision which is easily obtained in a HRI environment, and we propose the Mid-Level Classification Weights (MLCW) [32] to guide the pooling operator.

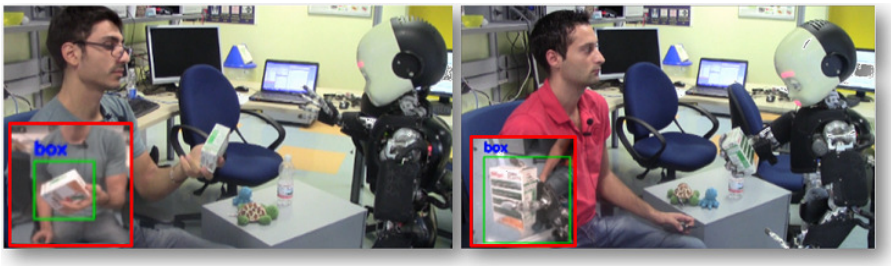
The pipeline is implemented and currently in use on the iCub platform, and demonstrated by two applications, which implement two specific use-cases: the first one (Fig. 2(a)) aiming at on-the-fly learning of new objects followed by recognition, with the help of the iCub attentive system, the second one (Fig. 2(b)) at learning and interacting with a specific type of objects lying on a desk.

Also we produce a thorough quantitative analysis on pre-recorded data. Lacking a benchmark reference for the robotics community, we rely on two standard computer vision benchmarks (Caltech-256 [34] and Pascal VOC 2007 [26]), for a comparative analysis of our coding and pooling methods with respect to the state of the art. Moreover, to provide a quantitative analysis of the results on our reference setting, we acquired a dataset for classification and categorization in a HRI scenario. The iCubWorld is available for download [33] and will hopefully facilitate the reproducibility of visual recognition results in a robotics setting.

In the remainder of this section we discuss the related works, while the rest of paper is organized as follows. Section 2 describes the two applications implemented on iCub that are based on our visual recognition method, which is thoroughly presented in Section 3. In Section 4 we discuss the experimental analysis we performed, proposing both qualitative and quantitative evaluations, while in Section 5 we summarize some common practices that may help robotics researchers interested in the visual perception field. We conclude with a final discussion in Section 6.

Related Work

In robotics, visual recognition is often addressed by exploiting 3D information to obtain invariant models of the observed scene. A large portion of methods build global topological representations of the objects that encode local geometric relations [24] or perform clustering to directly derive tridimensional templates from point clouds [2]. Systems that perform recognition based only on visual cues are typically employed to solve pose estimation problems [16, 23, 39, 70]. They often share the following core strategy: local features (such as SIFT [53] or SURF [6] — see also [57]) are first extracted from raw images and then matched with a learned object template via robust outliers rejection schemes (RANSAC) [38]. Sparse local SIFTs have been used to build robust 3D models of the object which is used to perform real-time and multiple instances detection even in severely cluttered environments [16]. Recognizing instances of objects can be done efficiently end effectively, even though most of the current methods require a costly off-line training phase where all the viewpoints of the objects have to be shown, typically with no clutter and known poses [16, 41].



(a)

(b)

Fig. 2. Examples of applications with iCub. Fig. 2(a), *On The Fly Learning and Recognition* application. The iCub's motion detector (left) and forward kinematics (right) both provide a reasonably good estimation of the object position within the image. The former is used in **human mode**, when a demonstrator moves an object in front of the iCub, the latter in **robot mode**, when the iCub holds the object in its hands. Fig. 2(b), *Learning and Interacting* application: a typical situation. The demonstrator asks the robot for the position of the car. The iCub first recognizes the object and then points at it.

Clutter and view-point variation have been successfully addressed in computer vision starting from the Bag of Words (BoWs) paradigm [18]. In its original formulation, images are seen as unordered collection of descriptors quantized into visual words (coding stage). These quantizations are then mapped into a histogram representation (pooling stage) that is used as input for a classifier. This basic approach has been extended to incorporate spatial information, with the so-called *Spatial Pyramid Representation* (SPR) [51], which is one of the first examples of coding-pooling pipelines. After this initial work, many improvements have been proposed in the literature. As for coding, the concept of sparsity has been widely addressed. Besides its biological motivations [59], sparse coding has been shown very effective to address classification tasks – see for instance [21, 30, 42, 76] and references therein. Sparse coding has been inserted in the classical SPR in [81]. Other improvements which is worth mentioning are adaptive data representations [47], mid-level features [10, 48], coding employing locality constraints [78], models to enhance spatial information [66]. All the above models look for the most appropriate encoding, by minimizing the reconstruction error. In this setting, each atom of the dictionary contributes for one single value of the code, i.e. its contribution in the linear combination. This means that, in order to achieve high accuracy, they require large dictionaries [12]; consequently the coding stage becomes slow and the problem becomes unfeasible for large-scale settings, such as content-based image retrieval on large databases [61]. In this context, alternative models based on the use of first (Super Vectors (SV) [84] or aggregation of local descriptors (VLAD) [43]) or second order statistics (Fisher Vector (FV) [65]) have been proposed. These methods ensure informative representations even with small dictionaries, but for their nature they appear to be more appropriate in content-based retrieval settings, where the spatial configuration of the image can be disregarded. Also, their computational cost makes them inappropriate in real-time robotics (see [65] for an analysis). As for pooling, it has been observed how common operators, such as max or average pooling, may produce an unrecoverable loss of spatial information if the image regions support is not designed appropriately. Other improvements have been made in the pooling stage: combining the standard SPR with a configuration space description in

order to capture similarity neighborhoods [11]; weighing the visual word locations in order to understand which positions of the images are relevant in the classification stage [36]; learning the regions of the pyramid directly from the data [45]; using object detectors to guide the pooling stage [64]. In [31] instead, each class scores different parts of the image, the pooling stage is performed based on these scores. VOC 2012 winners [13] exploit both object detectors and saliency maps to better refine the pooling.

Complementary visual recognition pipelines try to learn also the lowest level of the pipeline (i.e. local descriptors) making use of deep architectures. One example is the work of [49] where multiple sequences of coding-pooling-normalization steps are carried out for a fixed number of levels. The final image classification is performed through a fully connected network. Another method of this trend is the HMAX framework [68], which is an algorithmic model of the recognition process in humans. HMAX retraces the human's ventral stream structure of simple and complex cells forming a hierarchy of alternating layers that can be interpreted as a sequence of coding and pooling stages. The main difference between the one-layer coding-pooling pipeline and deep architectures is the initial data processing: whereas deep architectures learn local features directly from the data, in BOW-like systems they are assumed to be fixed, usually hand crafted (e.g. SIFTs). In spite of this, coding-pooling pipelines obtain accurate results. More importantly for the applications, they do it with low computational cost overall, whereas deep learning algorithms, are fast in test phase but require extremely large amounts of training data and are very demanding in terms of computational resources during training, even when GPU implementations are used.

Regarding the learning stage, the described methods rely on linear classifiers (typically SVM [72]), with a one-vs-all paradigm. In [1] authors make a fair and exhaustive comparison among different learning strategies. They showed the superiority of one-vs-all paradigm also in large scale settings. Linear models are to be preferred when real-time computation is required. In this case it is possible to explicitly map descriptors into a new feature space that approximate the use of non-linear kernels [75].

There appear to be very few works that investigated the use of visual recognition pipelines in robotics [9, 37]. To this regard, in a recent study [14] we performed an exhaustive evaluation of modern hierarchical image representation methods in a real robotic application. Our work highlighted several positive aspects of visual recognition pipelines in the setting considered, such as real-time efficient training and testing, robustness to clutter and noise, high accuracy for classification.

$2D$ **Vision and Depth.** In the last few years, affordable range sensors such as the MS Kinect [40] have become remarkably popular in robotics. As a consequence, the idea of combining $2D$ vision with depth information has attracted the attention and interest of both robotics and computer vision communities [8, 22, 46, 50]. Indeed, a natural intuition is that while standard RGB images allow to learn and recognize the visual appearance of an object (e.g. texture, colours etc.), depth information is more suited to capture shape features.

The basic approach in these settings is to obtain separate data representations for the two modality and then train a joint machine learning system over the combined feature spaces [50]. However, methods that account for the fusion of $2D$ and depth information during the data representation phase have been proposed [7, 8], achieving significant improvements in recognition accuracy over methods that learn separate data representations for the two modalities. Most of these approaches adopt coding and pooling strategies from the visual recognition literature as the one describes above, but apply them to the 4-channels RGB+D images rather than the standard 3-channels RGB [8, 50]. More recently, sophisticated RGB+D combination strategies have been proposed using convolutional neural networks [22, 67], however they have shown marginal improvement over previous methods while imposing heavier requirements on computational resources.

Depth information, although being extremely helpful for visual recognition, is not always available. For instance, in outdoor scenes, ambient light disrupts the signal acquired by sensors such as the Kinect. Moreover, when an object is too close to the camera (e.g. when the robot is holding it in its hand as in the application described in Sec. 2) both range sensors

and stereo matching techniques can often time fail to capture the correct depth; indeed, the Kinect has an 80cm minimum working range while the of stereo matching strongly depends on the distance from the object and the stereo baseline. Since we are interested in a system capable to perform object recognition robustly regardless of the environmental nuances, in this work we focus exclusively on visual recognition methods that can be used also in absence of depth information. However note that all the methods reviewed and proposed in this work are independent on the number of image channels and therefore can be naturally applied to RGB+D images (interpreted 4-channels arrays) leading to similar strategies to the one described in [8].

2. Human-iCub Interaction

Human Robot Interaction (HRI) applications present challenging tasks for autonomous systems. Indeed, in these settings the robot is typically required to rapidly interpret the current state of the scene in order to promptly react to the actions/requests of the human actor(s). In particular, the ability to process and semantically parse the visual information plays a fundamental role for most HRI scenarios, since visual communications is one of the most natural channels commonly adopted by humans in their everyday life. In this paper we concentrate on a specific aspect of the problem of understanding the visual scene, namely object recognition, and we will discuss practical applications on the iCub robot platform.

2.1. iCub in a few words

The iCub is a full humanoid robot developed within the FP-6 European Project RobotCub [1]. Here we provide minimal information on hardware and software, referring the interested reader to [55, 58] . The robot is equipped with 53 degrees of freedom (DoF): 6 DoFs for each leg, 7 DoFs for each arm and 9 DoFs for the hands. The head comprises a three DOF neck (tilt, pan and roll) and cameras that can rotate around a common tilt and independently around pan axes to control the gaze direction and the angle of vergence.

The robot is endowed with a complete sensory system [58], in this work we mainly consider the vision system, formed by two dragon fly cameras, each mounted on the end effector of the kinematic chain of each eye. They constitute a human-like stereo pair that can be used for depth estimation, which is not considered in this work.

Images are acquired by the PC104 computer mounted on the head and streamed to the network at the maximum rate of 60 Hz and resolution up to 640×480 pixels.

The iCub software architecture was built on top of YARP , an open source middleware that supports code development for iCub. The robot offers a YARP interface to communicate with its motors and sensory system. All the users need to know is how to use the interface to send commands to the robot and access the sensory information. The complexity of the networking, low-level hardware and device drivers are completely hidden to users. YARP software is organized in *modules*; a collection of modules forms an *application* which is designed to achieve a particular functionality. We now discuss two different applications which have been implemented based on two visual recognition use cases.

2.2. Visual recognition use cases

In this section we describe two HRI applications in which vision plays a central role, and that are currently implemented on the iCub and available to the community. These scenarios serve both as motivation for our investigation on the impact of visual recognition methods in robotics and as a test-bed where new vision algorithms can be tested. The challenge of HRI could in fact be of interest to the computer vision community, as it imposes several constraint on the algorithms which are often overlooked in standard computer vision, such as the requirement for the system to perform in real-time and to be stable with respect to small view-point changes. The C++ code developed for the experiments described in this work is

[1] `http://www.robotcub.org/`

open source and can be found on the iCub Repository[2]. Considering the large diffusion of the iCub platform among several research labs in the world, this public engineered HRI architectures represent an other important contribution of this work.

In the following we describe the two use-causes we demonstrated.

On The Fly Learning and Recognition In this scenario, a human teacher shows different objects to the iCub which is required to learn their visual appearance and to be able to recognize them in the future. During each training session, the human provides a single verbal annotation to each object and the robot observes it from multiple points of view by means of physical exploration. Such process can take place in two different ways, as reported in the following:

- **Human Mode (Fig. 2(a) - left)**. The demonstrator moves the object in front of the robot, so that the robot can observe the object from different viewpoints. An independent motion detector [15, 29] is employed to identify a bounding box around the moving target and to have the robot actively track it with its gaze.
- **Robot Mode (Fig. 2(a) - right)** In this modality, the demonstrator gives the object to the robot. Once the object has been grasped, the iCub starts performing random exploration of its workspace with its own hand while visually tracking the hand-held object. In this way the target is observed from several viewpoints, although the hand of the robot is always present in images and often causes occlusions of the object of interest.

In the training modality the image regions within the bounding box are saved as training examples, associated with a label provided by the human teacher. At the end of the data acquisition session, a *learning* stage is performed, updating the models of the objects of interest. At run time, the content of each bounding box is associated with one or more object labels, and the most likely estimate is associated with the object. The iCub, upon request, is able to provide an audio feedback (the object's label). In this application the detection of the object of interest is helped by motion information or robot kinematics. During data acquisition the object is shown from different points of view, thus the learning module is rich of complementary visual information. The challenges here are mainly due to background and illumination changes, which are not controlled, and by occlusions caused by the hand holding the object.

A video demonstrating an implementation of this use case is available at `http://www.youtube.com/watch?v=vhPLUNg9r5k`. For more details we refer the reader to[29, 33].

Learning and Interacting In this use-case we consider a more complex scenario where the iCub is standing in front of a table and a human agent communicates verbally with it by requiring the robot to carry out interaction tasks with specific classes of objects lying on the table. In its initial state, the system does not possess any knowledge regarding the visual appearance of the objects it will have to interact with. Therefore, the human supervisor will provide also verbal annotation for the objects in order for the system to build visual models of the elements in the scene [83]. The system is required to detect and recognize one or more objects in the scene based on appearance properties only and using observations from a single viewpoint — differently from the first application we discussed. Then, it can also perform interaction tasks as required by the human agent. At the moment, the possible actions we have implemented on the iCub are *point*, *touch*, *push* and *grasp* which are self-explanatory.

The application therefore alternates between phases of *learning* of new entities (or updating of known visual models) and *acting*, during which the robot exploits its visual experience to detect known objects and interact with them. More complex behaviours developed on the iCub, featuring a broader set of interaction capabilities can be found in [71].

A video demonstrating this application is available at `http://www.youtube.com/watch?v=ZIpVrD6e-kA`.

[2] `https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub/contrib`

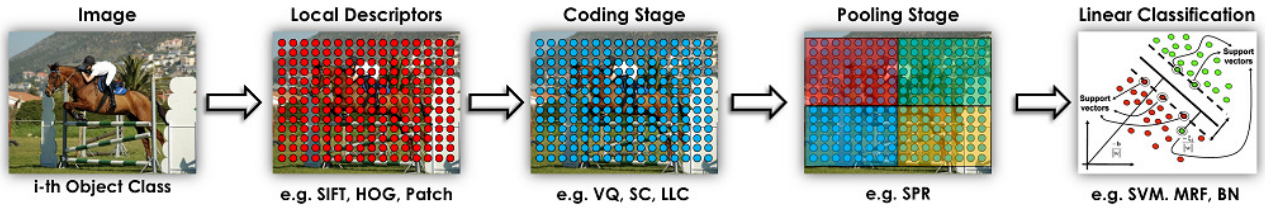


Fig. 3. Visual recognition coding-pooling pipeline . A multi-level architecture builds hierarchical image representation that are fed to a linear classifier. See Sec. 3 for details.

3. The Visual Recognition module

In this section we describe in details the standard coding-pooling pipeline we refer to, and then describe the innovative elements we propose for the coding and pooling stages.

3.1. The pipeline

Fig. 3 reports the typical pipeline organized in subsequent layers. In the first layer local features are extracted to capture the low level statistics of the image. A dictionary learning step typically accompanies the first level, where a set of relevant descriptors (or atoms) is learnt from data. To follow, the pipeline considers a coding stage, where local features are represented in terms of the learnt dictionary, and a pooling stage which summarizes the description over the whole image or a set of image regions. During coding-pooling, high level image statistics are learned directly from the data. Finally a learning stage involving a linear classifier is employed.

Low-Level Descriptors A set of local descriptors $\mathbf{x}_1, \ldots, \mathbf{x}_M$, with $\mathbf{x}_i \in \mathbb{R}^d$, is extracted from an image I. Examples of local descriptors are image patches, SIFT [53], or SURF [5] (either sparse or dense). SIFTs are considered so far, the most effective local descriptor for visual recognition [56]. In image retrieval tasks, where the goal is to retrieve the most similar images to a query image, sparse sets of descriptors are often employed [43, 65]: first a corner detector is run and then a SIFT descriptor is extracted for each retrieved keypoint. In categorization problems, instead, a dense regular grid is to be preferred, according to the evaluation reported in [35]. This is due to the fact that keypoint detectors usually focus on image regions where large variations of the gradient occur, and as a consequence the retrieved points will be placed in strong textured image patches. In categorization problems, however, it is likely that texture strongly changes between two different instances of the same class, whereas their shape is preserved. A regular grid of local descriptors instead seems to be suitable to the goal of capturing image statistics.

Dictionary Learning The goal of dictionary learning is to learn a basis (or set of atoms) relevant for the considered dataset. These atoms should be able to capture the essence of any datum, such that $\mathbf{x} \simeq \mathbf{Du}$. A dictionary $\mathbf{D}$ is a $d \times K$ matrix, where K is the number of atoms and d the dimensionality of the local descriptors. Typically $K > d$, therefore D corresponds to an over-complete space. Dictionaries allow us to map local descriptors into a common reference frame where the comparison among images is more effective and robust. Dictionary learning should be performed selecting a subset of all the SIFTs descriptors extracted from an image dataset, in order to contrast over-fitting. Typically a set of $1M$ descriptors is used in standard computer vision benchmarks (see [81]), although this choice depends on the size of the data-set.
In this work we consider dictionary learning by K-Means [18], which minimizes the following reconstruction error:

$$\begin{aligned} &\min_{\mathbf{D},\mathbf{U}} \|\mathbf{X} - \mathbf{DU}\|_F^2 \\ &\text{s.t.} \quad \text{Card}(\mathbf{u}_i) = 1, |\mathbf{u}_i| = 1, \mathbf{u}_i \succeq 0, \forall i = 1, \ldots, T \end{aligned} \tag{1}$$

with $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ is the set of local descriptors used for the dictionary learning, $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_T]$ are the cluster membership codes, with each $\mathbf{u}_i \in \mathbb{R}^K$, K dictionary size. The constraint $\mathrm{Card}(\mathbf{u}_i) = 1$ means that only one element of $\mathbf{u}_i$ is non-zero, and $\mathbf{u}_i \succeq 0$ means that the element must be greater than zero, i.e. each local descriptor belongs to one cluster.

Coding The coding stage maps the input features $\mathbf{x}_1, \ldots, \mathbf{x}_M \in \mathbb{R}^d$ into a new, possibly over-complete, space of codes $\mathbf{u}_1, \ldots, \mathbf{u}_M \in \mathbb{R}^K$. Coding operators can be categorized according to the output they produce. Among them, BOW-like methods provide a simple statistics such as visual word occurrences or amounts of atom contribution to the linear combination. Typically, these operators minimize the following reconstruction error:

$$\begin{aligned} \mathbf{u}_i = &\arg\min_{\mathbf{u}} \|\mathbf{x} - \mathbf{D}\mathbf{u}\|^2 + \lambda \mathrm{R}(\mathbf{u}) \\ &\text{s.t.} \quad \mathrm{C}(\mathbf{u}) \end{aligned} \tag{2}$$

where different regularization terms $\mathrm{R}(\mathbf{u})$ and constraints $\mathrm{C}(\mathbf{u})$ lead to different algorithms. In the following we summarize common choices from the literature which have been considered in our evaluation phase.

Vector Quantization (VQ) [51] Given a dictionary $\mathbf{D}$, the VQ coding minimizes the following reconstruction error of $\mathbf{x}_i$:

$$\begin{aligned} &\min_{\mathbf{u}_i} \|\mathbf{x}_i - \mathbf{D}\mathbf{u}_i\|^2 \\ &\text{s.t.} \quad \mathrm{Card}(\mathbf{u}_i) = 1, |\mathbf{u}_i| = 1, \mathbf{u}_i \succeq 0 \end{aligned} \tag{3}$$

which corresponds to Equation (1), with a fixed dictionary.

Sparse Coding (SC) [81] SC uses the concept of sparsity to encode each descriptor using a subset of relevant atoms. The idea is to minimize:

$$\min_{\mathbf{u}_i} \|\mathbf{x}_i - \mathbf{D}\mathbf{u}_i\|^2 + \lambda \|\mathbf{u}_i\|_1 \tag{4}$$

with the L_1-norm that induces sparsity and allows the learned representation to capture salient patterns of local descriptors. The parameter λ is a trade-off between the sparsity and the approximation of the signal. A generalization of the functional, with a double minimization with respect to both $\mathbf{u}$ and $\mathbf{D}$ allows to learn the dictionary while sparsifying.

Locality-constrained Linear Coding (LLC) [78] This coding method starts from the assumption that locality is more essential than sparsity, as locality leads to sparsity but not vice versa. In this case the coding operator selects only a subset of the coded vector $\mathbf{u}_i$, whose components are related to the k nearest neighbors of the input $\mathbf{x}_i$ and the atoms in $\mathbf{D}$. We denote with $\overline{\mathbf{u}}_i$ the components of $\mathbf{u}_i$ related to the k nearest neighbors of $\mathbf{x}_i$ in $\mathbf{D}$, which we denote with $\overline{\mathbf{D}}$, hence the codes are obtained minimizing:

$$\begin{aligned} &\min_{\overline{\mathbf{u}}_i} \|\mathbf{x}_i - \overline{\mathbf{D}}\overline{\mathbf{u}}_i\|^2 \\ &\text{s.t.} \quad \mathbf{1}^{\mathrm{T}}\overline{\mathbf{u}}_i = 1 \end{aligned} \tag{5}$$

Codes entries that are not associated with any neighborhood in the dictionary are set to zero.

Pooling The locality of the coded descriptors $\mathbf{u}_1, \ldots, \mathbf{u}_M$ is unable to capture image properties that are significant at higher scales, therefore a pooling stage is usually introduced after the coding phase. The pooling operator g takes a set of

Fig. 4. Example of the spatial pyramid layout with 3 levels The image is partitioned in 1, 4 and 16 spatial cells respectively.

codes $\mathbf{u}_1, \ldots, \mathbf{u}_M$ and outputs a single feature vector $\phi \in \mathbb{R}^K$ whose aim is to summarize the most salient properties of the set. The most popular operators adopted to this purpose are the *average pooling*, which computes the mean vector (in the space of codes) of the set of local measurements $\mathbf{u}_1, \ldots, \mathbf{u}_M$, and the *max pooling* that returns the vector of entry-wise maxima among all codes.

It has to be noted that a single pooling operation performed on the entire image would certainly capture the statistically relevant properties of an image but at the same time would produce a representation that is unable to encode spatial relations among entities in the scene. An effective way of addressing this problem was originally proposed in [51], where the Spatial Pyramid Representation (SPR) was introduced. In this setting, the pooling operator is applied multiple times over codes located at S overlapping regions on the image (see Fig. 4). Following the notation of [11], let us consider S spatial regions and define for each $s \in S$, the set Y_s of indices i of those codes $\mathbf{u_i}$ that lie within region s. The pooling operator g acts on the regions in S, producing each time a feature vector $\phi_s \in \mathbb{R}^K$ encoding the visual properties of the patch,

$$\phi_s = g\left(\{\mathbf{u}_i\}_{i \in Y_s}\right). \tag{6}$$

The final descriptor of the image $\mathbf{z}$ is the concatenation of the descriptors ϕ_s among all the S regions, thus $\mathbf{z} \in \mathbb{R}^{KS}$. This vector is a now a novel representation of the original image but, differently from the raw concatenation of pixel values, it efficiently encodes salient information about the spatial properties of the scene.

Learning Stage At the end of the coding-pooling stage, the data can be represented according to the new feature vector $\mathbf{z} \in \mathbb{R}^{KS}$. From this standpoint, the task of visual recognition amounts to a standard multi-class problem, exhaustively studied by the machine learning community. We consider N image categories, where the number of training examples is n_i for the i-th class. A widely adopted learning strategy is one-vs-all (OVA), where a multi-class classifier is obtained by combining several binary classifiers characterizing one class with respect to all the others. This approach offers several advantages compared to multi-class classifiers, as discussed in [63]. Basically OVA makes the assumption that each class is uncorrelated with the others, that is true when categories differ a lot. In practice, when the number of categories is large, it is likely that classes share common visual features and better learning strategies are needed.

Sparse coding-pooling descriptors are usually trained with simple linear classifiers, such as SVMs [72] in order to make them suitable for large-scale setting. Codes obtained through vector quantization instead usually require ad-hoc kernels to obtain good performances. Instead of using non-linear kernels one could employ the direct feature mapping proposed in [75]. In this work we make use of linear kernels for all the methods: they can be implemented efficiently, they are simple and effective once the data representation is appropriate and they do not add further parameters. It is also worth observing that the adopted data representation is, in a sense, an explicit mapping into a more appropriate feature space.

3.2. Coding with Best Code Entries (BCE)

As mentioned earlier, standard coding approaches consist in minimizing the local reconstruction error with respect to given (or learned) dictionaries and problem-specific constraints (e.g. sparsity). It is important to point out however, that the actual goal of the whole visual pipeline is not reconstruction, but rather to obtain a good representation for the subsequent classification task. Therefore, low reconstruction error could not necessarily be a fundamental aspect for good descriptors.

Following this line of thought, in this section we propose a novel coding method, Best Code Entries (BCE), that strays away from the usual approach to minimize reconstruction error but still leads to competitive or higher classification performances. BCE offers several benefits over competing coding methods, the first of which being its computational efficiency, and has also strong connections with recent results in computational neuroscience. Indeed, the sparsity property can be related to biological evidences [3, 52, 62]: several complex phenomena of V1 neural responses are not well explained by simple linear models (in which the response is a linear function of the input). For instance, many visual neurons display end-stopping, in which the neuron's response to a bar image of optimal orientation and placement is actually suppressed as the bar length exceeds an optimal length. Sparse coding can model the interaction (inhibition) between the bases (neurons) by sparsifying their coefficients (activations). In other words, each phenomenon can be described by a subset of neurons when they form a high overcomplete basis. This also explains why we should consider high informative feature space.

Stability of Coding Methods We first discuss a simple experiment that motivated our investigation for a novel coding method. We wanted to assess the stability of the coding methods with respect to small shift in location. Indeed, this scenario is extremely common in robotics applications where small movements in the actuators can cause the scene to slightly shift within the visual stream. It would be therefore a desirable property of the visual representation employed, to be stable to small shifts, since it would be favorable for images acquired close in time to be coded into similar descriptor vectors.

In the experiments reported in Fig. 5, we evaluate the stability of codes obtained using SC and LLC. We consider two dense grids of SIFT descriptors (patch size of 16 pixels) shifted by 2 pixels along the x-axis.

We measure the stability of coding methods according to two criteria: the mean squared difference between two codes extracted from the same grid coordinate (right) and the similarity between their sparsity patterns (left).

It can be noticed that SC provides a very unstable solution: the mean square difference between different codes is large, while the overlap between non-zero components is small. As for LLC, in spite of a large overlap of the non zero components, the difference between the codes is again quite big. Finally, the method we propose, reported later in the section, exhibits instead a very stable behavior.

Best Code Entries The coding method we propose focuses exclusively on sparsity, instead than considering the minimization of the reconstruction error. Consider a dictionary $D = [\mu_1, \ldots, \mu_K]$ to be given. We compute the similarity between each atom μ_k and the current input $\mathbf{x}_i$ by an appropriate similarity measure, or kernel. In order to enforce sparsity we ensure that only the subset of the bases that are really relevant for representing a signal are selected. We do this by choosing the k nearest neighbors (in the dictionary) of the projected local feature, in a similar fashion to that of LLC.

We denote with $\overline{\mathbf{u}}_i$ the k components of the code $\mathbf{u}_i$ related to nearest neighbors of $\mathbf{x}_i$ in the dictionary atoms, and let $\overline{\mathbf{D}} = [\overline{\mu}_1, \ldots, \overline{\mu}_k]$ be the corresponding atoms. Then, the non-zero entries of the code $\mathbf{u}_i$ be:

$$\overline{u}_{ij} = Ker(\mathbf{x}_i, \overline{\mu}_j) \quad \forall j = 1, \ldots, k \tag{7}$$

where $Ker(\mathbf{x}, \mathbf{y})$ is an appropriate kernel function.

In this work we consider a linear kernel, which implements a dot product between dictionary atoms and the input vector. This idea is supported by the recent advances in neuroscience [3] where dot products between an input and stored templates

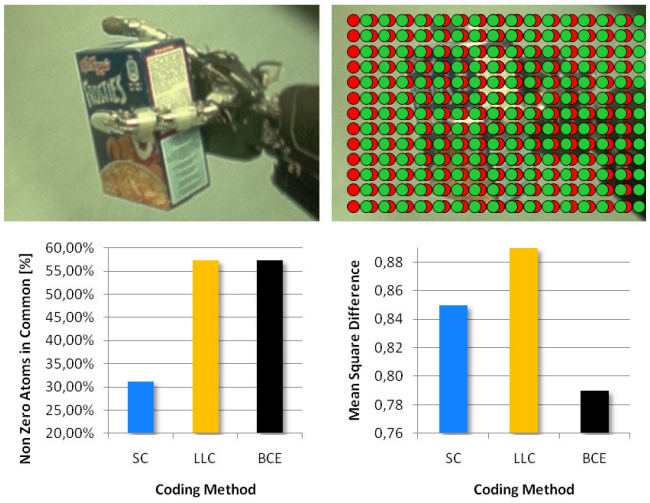


Fig. 5. Top: on the left, an image from the iCubWorld Data-set; on the right, grids of local descriptors with two different horizontal shifts. Bottom: evaluation of the stability of the codes computed from the two grids with different coding strategies (see text).

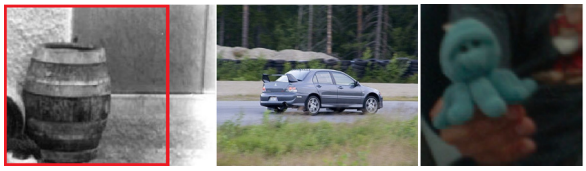


Fig. 6. Spatial bias on different datasets. Left: an image from Caltech-101 is wrongly classified because of an unusual position of the object (too far on a side). Center: a standard configuration for the PASCAL VOC dataset. Right: an example of the iCubWorld 1.0 that does not present any spatial bias.

followed by a max-pooling operator are used to model the complexity of the ventral stream.

3.3. Supervised Pooling

In this section we address the problem of defining a pooling operator that aims at capturing multi-scale spatial relations of the scene without relying exclusively on a Spatial Pyramid Representation. In particular we propose a supervised pooling approach to perform a data-dependent combination of both the spatial and semantic contents of an image in a single representation.

The idea of introducing a Spatial Pyramid Representation to capture the spatial relations within an image can greatly improve the descriptiveness of the final representation. Unfortunately, the way a specific spatial pyramid affects the classification performance is heavily dependent on the particular dataset at hand. For instance, it is well known among the computer vision community, that for the Caltech-101 dataset [34], a partition of $2^l \times 2^l$ regions (like the one in Fig. 4) is favourable, since the objects appear at different scales but in general tend to occupy the center of the image. On the other hand, for the PASCAL VOC dataset [25], the object of interest is usually located in the upper, center or lower regions of the image, and thus the typical approach is to define three horizontal regions each associated with one of these locations [54, 82, 84].

Fig. 6 shows different effects related to spatial bias for three image categorization datasets. The image on the left, from Caltech-101, is of particular interest: the object is not positioned in accordance with the majority of images of the same dataset, and it is wrongly classified by a standard spatial pyramid + SVM method. However, as soon as we crop out a part of

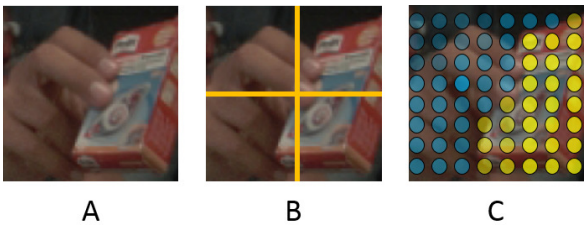


Fig. 7. Another image from the iCubWorld 1.0 (A) correctly classified by our pooling method (with a combination of spatial pooling (B) and supervised semantic pooling (C)) and erroneously classified by approaches that rely solely on spatial pyramids.

the image (the one outside the red square), the same classifier produces the correct categorization since the spatial features of the object "fall in" the appropriate regions of the pyramid.

We can infer that pooling on hand-crafted regions is mostly effective when we have a strong prior information regarding the objects position across the whole dataset. However, in natural images such as those acquired by an autonomous agent, this information is not available or is not reliable. A clear example of this inconvenience can be observed for instance in Fig. 7 (B), which depicts a typical image acquired from the iCub cameras during the Human-Robot Interaction session described in Sec. 2.2. As it can be noticed, since the object of interest is held in the hand of the demonstrator, its location within the image is clearly unrelated to the way the pyramid grid is positioned (indeed, while it is true that in Figure 7 the object lies mainly within the rightmost regions, on a subsequent image it could appear positioned only within the leftmost ones).

Pooling in the Feature Space We start our analysis by reviewing an extension of the SPR framework originally proposed in [11]. Such work was motivated by the observation that spatial pooling can often time combine together features that happen to be in the same image region but that are semantically very different. This approach can therefore lose relevant information by pooling together local descriptors that lie close on the image plane but very far apart in feature space. To overcome this problem, it was proposed to perform a data-dependent partition of the feature space for each individual image region $Y_1, \ldots, Y_S$ and then apply the pooling operator on those local descriptors that would belong simultaneously to the same image and feature space partition.

More formally, features associated to an image region Y_i were clustered in P separate feature space regions $X_1^{(i)}, \ldots, X_P^{(i)}$ using K-means. Then, feature codes $\mathbf{u}_i$ were pooled jointly considering each spatial region s and each feature cluster p separately, so that

$$\omega_{s,p} = g\left(\{\mathbf{u}_{i,j}\}_{(i \in Y_s, j \in X_p^{(i)})}\right). \tag{8}$$

The final image descriptor $\mathbf{z}$ was obtained by concatenating all the $\omega_{s,p}$ descriptors, namely $\mathbf{z} \in \mathbb{R}^{KSP}$. Notice that these descriptors, while being extremely effective are typically remarkably large. For instance, when the dictionary size is $K = 1024$, the spatial regions are $S = 21$ (namely the spatial pyramid is composed of $2^l \times 2^l$ segments with scales $l = 0, 1, 2$) and we choose $P = 64$ as suggested in [11], the final representation $\mathbf{z}$ has dimension $KSP = 1024 \times 21 \times 64 = 1,376,256$. Therefore, this method is inapplicable to settings where both space and computational times are of critical importance.

Separating the Image and Feature Domains The image descriptor proposed in [11] is advantageous in that it accounts for relevant configurations in the feature space. However, it forces all the S spatial regions to be partitioned in an equal number P of states within the feature space, leading to representations of equal length but very different amount of information. In addition, semantically proximal features (e.g. features lying on a given object) might be separated due to the regular partitioning of the spatial pyramid. This is undesirable since such an adjacency in feature space would not be reflected by the final description as we have already observed in Fig. 7.

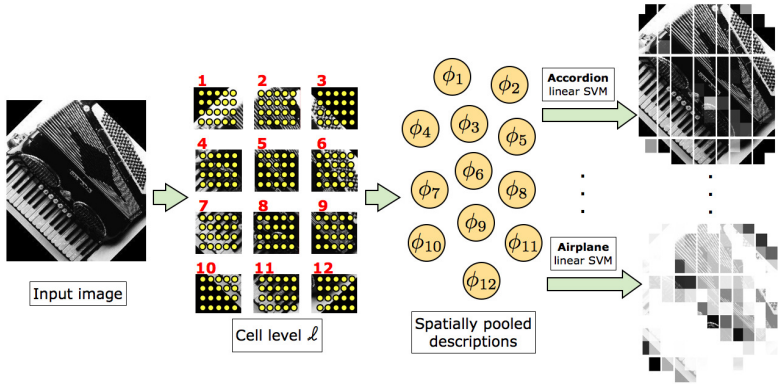


Fig. 8. Mid-level classification stage to learn the weights adopted for pooling the codes. Each image is decomposed in cells, and codes within each cell are pooled together. Such descriptions are fed to a linear One-vs-all SVM for each objects class. The scores are the weights, that give an idea of what a certain classifier is able to see in the image (the class *Accordion* "can see" better that the class *Airplane*).

Here we slightly change perspective and propose a different combination of pooling in the image and feature space, that leads also to a remarkable reduction in the image descriptor dimension. Indeed notice that, while being true that spatial cells and feature space bins are both designed to capture the geometric properties of the objects, they operate on two very different domains, image and feature space respectively. Therefore it seems more natural to perform pooling separately. We propose to extract two distinct descriptors, the first one $\mathbf{\Phi} \in \mathbb{R}^{KS}$ derived by a standard SPR on the image domain, the second one $\mathbf{\Psi} \in \mathbb{R}^{KP}$ encoding the *Feature Space Representation* (FSR). These two descriptors are obtained by concatenating vectors $\phi_s \in \mathbb{R}^K$ and $\psi_p \in \mathbb{R}^K$ separately:

$$\mathbf{\Phi} = [\phi_1, ..., \phi_S] \tag{9}$$

$$\mathbf{\Psi} = [\psi_1, ..., \psi_P] \tag{10}$$

where

$$\phi_s = g^1\left(\{\mathbf{u}_i\}_{i \in Y_s}\right) \quad \forall s = 1, \ldots, S \tag{11}$$

$$\psi_p = g^2\left(\{\mathbf{u}_i\}_{i \in X_p}\right) \quad \forall p = 1, \ldots, P. \tag{12}$$

In our method g^1 is the usual max pooling operator, while g^2 will be described in details later in the section. The final image representation is then obtained by concatenating the two descriptors $\mathbf{z} = [\mathbf{\Phi}, \mathbf{\Psi}] \in \mathbb{R}^{K(S+P)}$.

Notice that if we consider a standard dictionary size $K = 1024$, a spatial pyramid composed of $2^l \times 2^l$ segments with 3 layers ($S = 21$) and $P = 64$ the image representation proposed in [11] has a size $1.3E06$, while our size would be 87040 (corresponding to a 6% of the descriptor proposed by [11]).

3.4. Weighted Supervised Pooling

In our approach the Feature Space Representation (FSR) is built in a supervised way, taking into account the likelihood of a given feature to be observed in an image belonging to a given class. In other words, the representation is aware of the statistically relevant properties of each class individually.

Formally, we consider $P = N$ bins, where N is the number of classes of the problem under exam, and define a weighted version of the max-pooling operator as follows

$$g^2\left(\{\mathbf{u}_i\}_{i\in X_p}\right) = \max_{i\in X_p}(w_i^p \mathbf{u}_i) \quad \forall p = 1,\ldots,N \tag{13}$$

and thus, according to Eq. 12,

$$\psi_p(j) = \max_i w_i^p u_i(j) \quad \forall j = 1,\ldots,K. \tag{14}$$

The weights w_i^p have a natural interpretation as confidence values reflecting how likely it is to observe the code $\mathbf{u}_i$ in an image depicting class p. Therefore, in principle, it would be ideal to set for each $\mathbf{u}_i$ the weight $w_i^p = \mathbb{P}(Class = p|\mathbf{u}_i)$. However, since we do not have access to such latent distribution, we try to estimate it.

The underlying idea is to train N classifiers able to recognize subregions of the image and then use their scores as weights w_i^p. Indeed, most classification algorithms are somewhat related to the Bayes rule. For instance in binary settings, the predictor provided by Regularized Least Squares (RLS) converges asymptotically to the target function $\mathbb{E}(y|x)$, that is the expectation of the class label, given the input x [27]. In the multi-class case, by adopting a one-vs-all approach to learn the label associated with each $\mathbf{u}_i$, RLS would (asymptotically) provide the N score functions $f_p(\mathbf{u}_i) = \mathbb{E}(y_p|\mathbf{u}_i) = \mathbb{P}(y_p = 1|\mathbf{u}_i)$, where we have associated label $y_p = 1$ or 0 according to the presence or absence of class p in the image. Thus RLS could recover exactly the desired value for the w_i^ps.

In practice in this work we adopt a Support Vector Machine (SVM) [72], after we empirically observed that the two algorithms lead to comparable performances. Indeed, such evaluation needs to be performed several times for each image, causing a demanding computation. One advantage of SVM is that it reduces the computational effort in classification due to the sparse set of support vectors identified during training.

Mid-Level Classification Weights (MLCW). We now describe in details how we estimate the weights. First, we consider mid-level features, pooling together coded SIFTs belonging to a small spatial neighborhood into a single descriptor $\phi_s \in \mathbb{R}^K$. This descriptor is more robust to noise than considering single codes independently, and it gives invariance for small changes in the images. More in details we consider a single level l from the SPR – with small cell size – and decompose the images in the corresponding $2^l \times 2^l$ cells. Then, the codes in each cell s are pooled together with max pooling obtaining a mid-level (or object part) descriptor $\phi_s \in \mathbb{R}^K$ which is then fed to N SVM classifiers. They produce N scores for each descriptor: $f_p(\phi_\mathbf{s})$ $p = 1,\ldots,N$ which we use as weights $w_s^1,\ldots,w_s^N$ for all the codes $\mathbf{u}_i$ belonging to the cell s. Fig. 8 provides a visual intuition of the method.

Combining SPR & FSR. As already mentioned, the simplest way to combine the descriptors $\mathbf{\Phi} \in \mathbb{R}^{KS}$ for the SPR and $\mathbf{\Psi} \in \mathbb{R}^{KP}$ for the FSR is to concatenate them in a single vector. However, in general it could happen that one of the two representations results more useful and relevant than the other. A principled approach often used to combine heterogeneous features is to resort to Multiple Kernel Learning (MKL), based on the idea of associating each feature with an appropriate kernel independently. Then a global kernel $\mathbf{K}_{opt}$ is obtained as a weighted sum of the multiple contributions (in our case just two) where the actual relevance of each component is learnt directly from the data. More formally, $\mathbf{K}_{opt}$ is a linear combination of kernels:

$$\mathbf{K}_{opt}(\mathbf{z}_i,\mathbf{z}_j) = d_S\mathbf{K}_S(\mathbf{\Phi}_i,\mathbf{\Phi}_j) + d_P\mathbf{K}_P(\mathbf{\Psi}_i,\mathbf{\Psi}_\mathbf{j}) \tag{15}$$

with $\mathbf{K}_S$ and $\mathbf{K}_P$ denoting just the linear kernels on SPR and FSR respectively. The weights d_S and d_P are learnt using the algorithm originally proposed in [73]. We refer the reader to [73, 77] for all the details on the method adopted. Notice that when $d_S + d_P = 1$ the kernel $\mathbf{K}_{opt}$ reduces to the standard linear kernel applied to the concatenation vector $\mathbf{z} = [\mathbf{\Phi},\mathbf{\Psi}] \in \mathbb{R}^{K(S+P)}$.

		plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	avg
CODING	BOW	69.2%	55.8%	40.2%	62.4%	22.9%	52.0%	71.8%	52.5%	48.9%	39.0%	39.9%	40.2%	71.8%	58.7%	78.5%	25.0%	40.3%	46.2%	68.6%	45.9%	51.5%
	SC	70.5%	59.2%	41.8%	66.5%	25.7%	57.8%	74.1%	54.6%	51.4%	39.3%	44.1%	41.0%	74.9%	61.4%	80.4%	26.7%	39.2%	50.7%	70.7%	48.9%	**54.0%**
	LLC	68.2%	57.7%	39.9%	61.6%	24.0%	57.4%	73.4%	53.5%	49.7%	36.9%	42.3%	39.6%	73.4%	62.2%	79.4%	23.8%	42.7%	48.4%	68.0%	47.7%	52.5%
	BCE	68.5%	56.6%	42.3%	62.4%	24.4%	56.5%	72.3%	54.9%	52.3%	41.3%	43.7%	39.6%	71.7%	60.2%	80.4%	26.1%	41.2%	50.28%	69.4%	47.8%	**53.1%**
POOLING	max	68.2%	57.7%	39.9%	61.6%	24.0%	57.4%	73.4%	53.5%	49.7%	36.9%	42.3%	39.6%	73.4%	62.2%	79.4%	23.8%	42.7%	48.4%	68.0%	47.7%	52.5%
	MWL	68.4%	56.9%	41.1%	62.9%	23.8%	58.8%	73.9%	53.4%	50.1%	37.2%	41.7%	40.4%	74.3%	62.1%	79.5%	24.1%	42.4%	49.3%	68.8%	48.8%	52.8%
	MLCW	74.2%	62.5%	49.8%	59.3%	32.6%	62.7%	78.8%	56.1%	52.1%	43.2%	48.1%	45.6%	78.7%	63.3%	88.7%	31.5%	47.3%	52.3%	73.2%	51.1%	**57.5%**

Table 1. Average Precision (%) on PASCAL VOC 2007 for different coding methods (using max pooling) and three pooling strategies (using LLC as coding baseline).

4. Experiments

In this section we report the experimental analysis we performed. To provide a quantitative assessment of the proposed coding and pooling techniques with respect to the state of the art, we need to adopt computer vision datasets (Caltech 256 and PASCAL VOC 2007), lacking benchmark data explicitly referring to a robotics scenario. Then, to quantitatively evaluate the appropriateness of our full pipeline for robotics, we rely on benchmarks we recent proposed (the iCubWorld datasets), which will hopefully be considered as a reference by the research community in the next future.

Implementation Details. In our experiments, when comparing our methods with other algorithms, we use the code provided by the authors whenever available, and set the same parameters for all competing methods. As only exceptions, we use in-house implementations of BOW and of the pooling method suggested by [11], since there is no publicly available implementations.

To facilitate the reproducibility of the results, we report here all parameters needed to run the various methods. For the local feature extraction, we use a dense grid of points located every 4 pixels on the image, and we extract SIFT descriptors from 16×16 scale using SIFT GPU [79].

The dictionary is learned with K-Means using $K = 1024$ for the iCubWorld Data-sets, whereas for Pascal VOC 2007 and Caltech-256 we used $K = 4096$. The number of nearest neighbors for LLC and BCE has been set to $k = 5$, the λ parameter of SC is set to 0.1 following [81]. For the experiments employing BCE, we used an approximate version of K-NN (ANN) with a fixed number of comparisons $c = 500$, making use of efficient tree structures such as Kd-tree.
In the pooling stage, for Boureau's method [11] the number of centroids is set to $P = 16$ that seems a good tradeoff between accuracy and descriptor size according to [11].
For the spatial layout we use standard $2^l \times 2^l$ segments with scales $l = 0, 1, 2$ for Caltech-256 and iCubWorld, whereas in the PASCAL Benchmarks we use the layout suggested by the winner of the VOC 2007 [54], see Fig. 4. Mid-level object classifiers of Sec. 3.4 have been trained on the scale pyramid level $l = 4$.
Concerning the classification stage, we use LibLinear [28] as a batch solver. The SVM cost parameter C has been estimated on a subset of the training data through a standard cross-validation procedure. In general this value is not crucial for the final system accuracy.

4.1. A first assessment on Computer Vision Benchmarks

Although the motivating application for our contribution is the realistic HRI scenario, we first perform a thorough comparison of our methods with state of the art approaches on standard computer vision settings. To this end, we chose two of the most challenging datasets for recognition, namely the Pascal VOC 2007 and Caltech-256. In the following we briefly describe the main aspects of these datasets and comment on the performance of our contributions with respect to the current

	Method	Training Class Size			
		15	30	45	60
CODING	BOW	30.7	36.4	39.6	41.2
	SC	27.7	34.0	37.4	40.1
	LLC	32.0	38.4	42.2	44.3
	BCE	**32.4**	**39.0**	**43.1**	**44.9**
POOL.	max	32.0	38.4	42.2	44.3
	MWL	32.8	39.0	42.8	45.4
	MLCW	**35.2**	**40.1**	**44.9**	**47.9**

Table 2. Average accuracy (%) on Caltech-256 for different sizes of training samples per class. Coding methods are compared using standard max pooling, while pooling strategies are tested using LLC as coding baseline.

state of the art.

PASCAL VOC 2007. It is a challenging dataset depicting images from 20 object categories gathered from Flickr and characterized by a high variability of viewing angle, illumination, objects size, pose and appearance. Also, occlusions are frequent. The classification performance is evaluated using the Average Precision (AP) measure, the standard metric used by PASCAL challenge [25].

The first 4 rows of Table 1 report the classification performance of different coding methods. In order to isolate the effect of coding, the standard max operator was employed during the pooling stage. As it can be observed, in this set of experiments Sparse Coding (SC) resulted the best methods in terms of pure classification. However, since our final goal is the robotic application in real-time settings, we are more interested in the method achieving the best trade-off between accuracy and computational efficiency. Among the 4 methods considered, BCE is clearly the one fulfilling these requirements, having very similar performance to the SC but at the same time being several orders of magnitude faster (see Sec. 4.3).

The last 3 rows of Table 1 compare different pooling methods using LLC for coding. Clearly our proposed methods, MLCW systematically outperforms the current state of the art, leading to remarkable improvements in AP of about 5%.

Caltech-256. It consists of 256 object categories. Classes distinction is performed on a semantical principle and therefore can have large variability between instances (e.g. chairs have the same functional goal but can be visually very different one from the other). Similarly to the PASCAL dataset, we evaluated the individual impact of the coding and pooling strategies proposed in this paper (Table 2). In accordance with previous works [11, 36] we report the average accuracy (ratio of correct guesses on total entries) of all methods with respect to increasing sizes of training sets.

We observed a similar behavior to the one reported for the PASCAL dataset: BCE exhibits comparable performances to the best state-of-the-art coding approach (in this setting even with better results) while being remarkably faster than any of the other methods tested (see Sec. 4.3). Regarding the pooling stage, the results reported in Table 2 show the superiority of our MLCW strategy with respect to both the standard max and the more sophisticated MWL operator.

4.2. Experiments with the iCub humanoid robot

In this section we focus specifically on the robotic domain.

In the following, we report classification results of different methods over two datasets that were acquired during several runs of the HRI *On The Fly Learning and Recognition* application described in Sec. 2.2 in both Human and Robot mode. The goal of these datasets is to build a reproducible setting that could serve as a benchmark for the application of visual

Fig. 9. The iCubWorld 1.0 Dataset. Samples of the 7 classes collected for the *robot* (top strip) and *human* (bottom strip) datasets.

	Method	Acc. RM (%)	Acc. HM (%)
CODING	BOW	82.8	69.4
	SC	84.1	**75.4**
	LLC	**84.7**	73.2
	BCE	**84.3**	**74.9**
POOL.	max	83.3	75.37
	MWL	84.2	76.7
	MLCW	**87.1**	**78.3**
	BCE + MLCW	**88.5**	**79.6**

Table 3. Accuracy (%) on the iCubWorld 1.0, for both Robot Mode (RM) and Human Mode (HM).

Fig. 10. The iCubWorld Categorization data-set. Examples from the 10 object categories.

	Method	Accuracy (%)
CODING	BOW	43.6
	SC	**47.7**
	LLC	43.8
	BCE	**46.0**
POOL.	max	43.8
	MWL	45.7
	MLCW	**49.4**
	BCE + MLCW	**50.1**

Table 4. Accuracy (%) on the iCubWorld Categorization Dataset.

recognition methods on robotics domains. Details on these datasets can be found in [14, 33].

iCubWorld 1.0[3]. It has been designed for single-instance recognition problems (as opposed to categorization scenarios where the system is required to determine the category to which the query instance belongs). It contains images depicting 7 different objects (Fig. 9) acquired both in Human and Robot mode. For both training and testing and for each exploratory modality we employed 500 images per class for training and 500 for testing.

In Table 3 we report the classification results of the methods we tested on this dataset. Similarly to the experiments in Sec. 4.1, we isolated the contribution of different coding and pooling methods by adopting the standard max operator for all coding methods (SC, LLC, BOW and BCE), while using LLC as coding baseline to compare different pooling strategies.

As already observed on the standard computer vision benchmarks, our methods consistently outperform the competitors. However, the main characteristic of the iCub World datasets is that the object location within images can vary dramatically from sample to sample. Therefore it is unlikely that spatial pyramid methods could exploit position bias to improve results. To verify this aspect we performed a further experiment where we discarded the spatial pyramid representation while comparing different pooling methods: max pooling and MWL pooling are applied to a single scale pyramid ($l = 0$), while our MLCW is applied considering the feature space representation only, with no contribution from the spatial pyramid side. Notably, even when no spatial information is employed, our method exhibits high accuracy, with 81.9% for the robot mode and 73.3% for the human mode, while MWL achieves 78.0% and 68.3% respectively, and max pooling 70.1% and 65.9%. This suggests that, if we can not rely on a prior on the object position in the image, hand-crafted image regions may not be a suitable choice for pooling, while some supervision in the pooling partitioning is beneficial.

[3] The iCubWorld 1.0 Dataset can be downloaded at `http://www.iit.it/it/projects/data-sets.html`.

iCubWorld Categorization[4] . It was recently presented in [33] with the goal to serve as a benchmark for visual categorization methods in robotics. At the time being, the dataset features 10 object categories of different complexity with respect to shape and textures (see Figure 10). For each category, 3 objects instances with 200 training and 200 test frames each are collected.

The most challenging aspect of this dataset is the presence of structured clutter, which means that the image background does not provide useful contextual information that could be exploited by the learning system as opposed to what usually happens in typical retrieval data-sets such as Caltech or PASCAL. For a more detailed discussion on the impact of structured clutter we refer the reader to [33].

In Table 4 we report the results for our experiments in Human exploratory modality ($T3$ test in [33]). In this very challenging scenario the best performing coding method is SC, even though BCE is again very close (just $> 1.6\%$ in accuracy) while being 3 time faster. Furthermore, the best combination between coding and pooling results to be our contribution, namely BCE + MLCW.

4.3. Computational Complexity

One of the most critical aspects of realistic applications in robotics is computational efficiency. Indeed in these settings (see the applications described in Sec. 1), the system is required to interact with a highly dynamical environment, and therefore its perceptual capabilities need to be as reactive as possible. In this section we thus compare the different coding and pooling schemes we consider with both a theoretical analysis of the costs and an empirical evaluation of the computational times.

Theoretical analysis We start by analyzing the computational requirements from a theoretical standpoint. A lot of efficient implementations for SIFT detection and extraction are publicly available, leading to real-time performances of the very first low-level analysis. Similarly, the learning stage can be done efficiently: when linear classifiers are employed it requires $\mathrm{O}(Nd)$ (N total number of examples) for batch solvers. The testing phase has constant d complexity time, since in the linear case it consists of a simple dot product. It goes that the computational effort required by the visual recognition methods are mainly due to coding and pooling steps.

In particular, the bottlenecks of the typical pipelines are the coding methods. VQ is very fast and it has a computational cost $\mathrm{O}(MKd)$ where M is the number of local descriptors in the image, K is the dictionary size and d the local descriptor size. Sparse Coding has a computational time in the optimal case of $\mathrm{O}(MKTd)$, with T the number of non-zero elements. In the worse case, i.e. no sparsity, it costs $\mathrm{O}(MK^2d)$. LLC is much more efficient, indeed the complexity is $\mathrm{O}(MKd + Mk^2d) \sim \mathrm{O}(MKd)$ where k is the number of nearest neighbors employed, and typically $k \ll K$.

The final complexity of our coding algorithm is $\mathrm{O}(\log(c)Md) \sim \mathrm{O}(Md)$, where c is the number of comparisons for K-NN. Remarkably it does not depend on the dictionary size, and therefore satisfies real-time performances even when huge dictionaries are employed.

The pooling stage is very fast when max or average pooling are employed. It goes by all the M descriptors in the image, therefore is not a bottleneck since the computational cost is $\mathrm{O}(MSd)$, where S is the number of image cells. Pooling in feature space (sec. 3.3) instead, has computational cost equal to $\mathrm{O}(MSd + P(MKd))$, where P is the number of feature space locations.

The computational cost of our pooling method is $\mathrm{O}(MSd + N(pd))$, where M is the number of local descriptors in the image, K is the dictionary size and d the local descriptor size, N the number of classes, p the number of cells at the considered pyramid level. Notably the cost is lower than [11].

Tab. 5 reports a summary of the theoretical complexities of coding and pooling methods.

[4] The iCubWorld Categorization can be downloaded from: `http://www.iit.it/it/projects/data-sets.html`.

	Method	Complexity (%)
CODING	VQ	$\mathrm{O}(MKd)$
	SC	$\mathrm{O}(MKTd)$
	LLC	$\mathrm{O}(MKd + Mk^2d) \sim \mathrm{O}(MKd)$
	BCE	$\mathrm{O}(\log(c)Md) \sim \mathrm{O}(Md)$
POOLING	max/avg	$\mathrm{O}(MSd)$
	Feature space	$\mathrm{O}(MSd + P(MKd))$
	MLCW	$\mathrm{O}(MSd + N(pd))$

Table 5. A summary of the theoretical computational complexities of coding and pooling strategies considered in our comparison (see text for details).

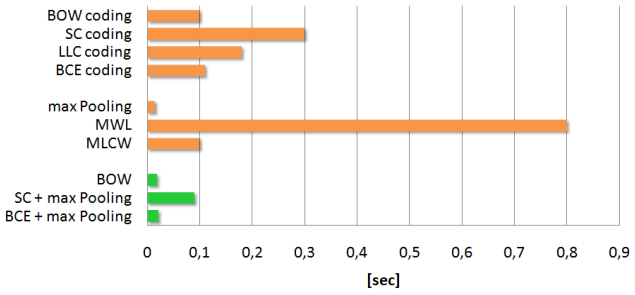


Fig. 11. Computational times (in sec.) of the visual recognition pipeline components on the 160×160 px images in iCubWorld Categorization dataset. MATLAB (orange) and C++ (green).

Computational efficiency Figure 11 reports the running times required by different methods using a MATLAB implementation (orange) on 160×160 images. For those methods for which a C++ (single-threaded) implementation was available running times are also reported (green). All experiments were performed on a 2.4Ghz Core 2 Duo machine.
A first observation is that the SIFT features extraction is based on GPU, thus the computational cost is negligible with respect to the total effort. The experiments also confirm the observation arisen from the theoretical analysis, stating that our coding method, BCE, is the best trade-off between accuracy and efficiency. Indeed, in terms of accuracy (or AP) it exhibits similar or even better performance to that of the competitors, while, in terms of efficiency it is definitely much faster than the others. Nevertheless, our MLCW method is almost 10 faster than the MWL proposed in [11] while being at the same time more accurate in classification. Clearly, its computational requirements are still far from the max pooling operator which is unsurprisingly extremely fast. However, due to the boost in accuracy provided by our method, we believe that it is in general worth to employ MLCW.

5. Tips & Tricks

In this section we summarize common practices to the benefit of robotics researchers interested in the visual perception field. We cite the appropriate source whenever it is available, otherwise we report experimental evidence supporting our considerations.

5.1. Multi-Scale Local Descriptors

When computing SIFTs on a dense grid, two important parameters must be set: the grid spacing and the scale. The first one represents the sampling distance of the local descriptors within the grid and it has been shown that denser grids lead to better performances [12] to the price of a high computational demand. A good trade-off between accuracy and efficiency is achieved with grid spacing equal to 4, as reported in [12].
The scale parameter, instead, represents the patch size of the local descriptor (alternatively, it may represent the bin size [74]). Since the choice of an appropriate scale is tricky, it is a common practice to extract local descriptors at multiple scales, typically with patch size $8, 16, 24, 32$. However in our experience this choice strongly depends on the considered data-set, e.g. in Caltech-101 there are no benefits in using multi-scale descriptors, see [81]. On other data-set, such as Caltech-256, Pascal VOC 2007 and iCubWorld Categorization, we noticed an increment of the final accuracy up to 4%.

5.2. Pre & Post Processing of the Data

Pre and post-processing of the data have been shown to improve the recognition accuracy. The work in [44] addresses image retrieval adopting Fisher vectors and shows how data whitening – i.e., data centering and rotating – improves the

performances by enforcing the Fisher vector assumption that SIFTs descriptors have diagonal covariance matrices.
It has also been experimented that there is a benefit when reducing the dimensionality of the problem, e.g. by selecting a subset of relevant data directions as in [65] where the best accuracy is achieved with 64 to 128 directions.
Dimensionality reduction techniques (PCA) on SIFT descriptors before dictionary learning can also improve performances [44]. An alternative strategy, instead, consists of performing a PCA analysis after the computation of the whole image representation. This approach is analyzed in [44], where it is observed that performances are quite stable even with a very low-dimensional descriptor (128 instead of $65k$). However this second strategy is data-set dependent, and it performs well in content-based image retrieval tasks, but we are not aware of any method employing it in categorization scenarios.

Method	Dictionary Size			
	512	1024	2048	4096
K-Means	83.1	83.5	83.0	83.1
Random	75.6	79.7	80.5	82.1

Table 6. Classification results (Average accuracy (%)) on 20 classes of Caltech-101 showing the difference between a random dictionary and a learned dictionary. The image representation is obtained with LLC and max pooling.

5.3. Dictionary Learning

In this section we reason on how critical dictionary learning is for the recognition performances, considering that this phase is the slowest of the whole pipeline. We experimentally observed how, even if a dictionary learnt on an appropriate selection of data can help boosting recognition, good performances may also be obtained by a random dictionary. It is only important to sample the dictionary from the same distribution of the data.

To show this, we consider a subset of 20 classes of Caltech-101, the same used in [30, 31]. We consider a well assessed coding and pooling procedure (LLC and max pooling) and image representations with different dictionary sizes $(512, 1024, 2048, 4096)$. Table 6 reports a comparison between random dictionaries and dictionaries learned with K-means. The obtained results confirm that as the dictionary size grows the performance difference between a random and a learned dictionary decreases.

Higher order coding methods often assume the dictionary to be generated according to a mixture model: for those techniques useful tips can be found in [65].

5.4. The Power (of) Normalization

A proper data normalization must be employed before the classification stage. It is a common choice to $l2$ normalize the image representations when coupled with linear classifiers. Here we discuss the so-called power normalization [19, 65] as alternative minimization strategy to $l2$. We consider a vector $\mathbf{v} \in \mathbb{R}^n$, where each component of the vector is normalized with the following power normalization:

$$v_i = sign(v_i)|v_i|^\alpha \quad \forall i = 1, \ldots, n \tag{16}$$

with $0 \leq \alpha \leq 1$. Intuitions on the role of normalization are given in [4, 19]: in particular it is interesting to observe how, with $\alpha = 0.5$, power normalization treats descriptor vectors as probability distributions and dissimilarity functions use the Bhattacharya distance instead of the Euclidean one. This is basically an explicit mapping to another feature space, where the highest code responses have less impact in the descriptor. We now give a visual intuition of this normalization: in Fig. 12, we consider a 3D data uniformly distributed on a unit sphere, indeed image representations typically belong to a n-dimensional sphere after the $l2$ normalization. Applying a square-root normalization $\alpha = \frac{1}{n}$, $n = 2$ we see that

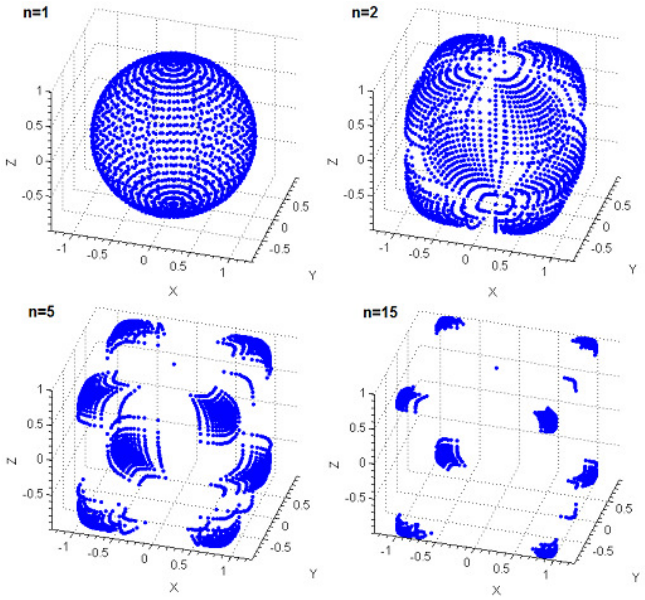


Fig. 12. Example showing the effects of the power normalization. 3D Data $l2$ normalized, are distributed on a uniform unit sphere in the 3D space. When $n = 1$ no normalization is performed. When $n = 2$ the square-root is performed, in general n is the n-root of the data components.

components are pushed to the boundaries of the sphere. If classes are well represented with the descriptors it is likely that when n increases the separation among two different object will become more clear, therefore linear classifiers are facilitated.

To prove this we used again 20 classes of Caltech-101 with BCE coding and max pooling. BCE starts from an accuracy of 85.3% which becomes 86.6% when the power normalization with $n = 2$ is applied, for $n = 3$ we obtain 87.0% and $n = 4$ achieves 87.06%, then the results stabilize.

5.5. Data Augmentation

Typical image representations are rather poorly invariant to viewpoint, scale, rotation and translation changes. A simple method to enforce invariance is to generate synthetic samples by transforming the original image I into a new one I', with an affinity matrix A. However a generic affine transformation could lead to an implausible natural image: for example, rotating both the foreground and the background could swap sky with the ground. Therefore care should be taken, depending on the type of information content. It is quite common and safe to use scaled version of the images, even in benchmark datasets: for instance, in Caltech-101 all images are re-scaled to a maximum of 300×300 pixels, preserving the original aspect ratio. Another technique adopted in [12] is to augment the data-set with a flipped version of the images: this improves the overall results in Pascal VOC 2007 $\sim 3\%$.

5.6. Unbalanced Data-sets

Unbalanced data-sets are very frequent in real-world applications. In binary classification problems, it often occurs that training data have too many negative examples compared to the positive ones [69]. For a multi-class problem the above definition can be extended: a data-set is unbalanced when classes are represented by a different number of positive examples. In this sense, Caltech-256 is balanced since in the experimental protocol the same number of examples per class is used [81], while it is not the case of Pascal VOC 2007. In the latter, indeed, it is common practice to consider binary classification

problems, i.e. each classifier is learned to separate a particular class from the others. Each binary problem is still unbalanced, however the evaluation protocol used overcomes this issue. Indeed, performances are evaluated using the average precision per class [25]: in practice, each classifier is considered separatly from the others, and it is possible to tune its own recognition threshold independently. This means that there is no need to compare scores among different models that could lead to wrong results for unbalanced problems.

The HRI scenarios (described in Sec. 1) is inherently characterized by unbalanced data-sets, since examples are shown incrementally over time and on demand depending on the object complexity. As a consequence, the amounts of positive examples may differ by orders of magnitude. In order to solve this problem one may appropriately weight the positive and negative instances, e.g. by adopting a weighted SVM [60]. Therefore, given a binary linear classifier of the form $f(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$, the vector $\mathbf{w}$ is learned by minimizing:

$$\begin{aligned} &\min_{\mathbf{w},\xi,b} \ \frac{1}{2}\mathbf{w}^t\mathbf{w} + CC_p \sum_{i \in Pos} \xi_i + C \sum_{j \in Neg} \xi_j \\ &y_i(\mathbf{w}\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1, \ldots, N \\ &\xi_i \geq 0 \quad \forall i = 1, \ldots, N \end{aligned} \tag{17}$$

where the weight is $C_p = \frac{N_n}{N_p}$, with N_n total number of negative examples, N_p total number of positive ones and C is the regularization parameter. An implementation is available on the LibLinear SVM library [28]. This simple scheme greatly improves the live demos accuracy with iCub.

To provide a quantitative evaluation of the effectiveness of this weighted learning method we selected 3 classes of the Caltech-101 with 30 positive examples per class (Accordion, Airplane and Anchor). When classifiers are trained with all the examples the accuracy is 100% for all the classes. We then used only 5 examples to train Accordion, in this case the accuracy of the Accordion drops to 28%, whereas with weighing properly the loss function it achieves 72%.

6. Discussion

In this paper we considered the problem of object recognition in robotics with particular focus on interaction where training is performed in real-time through the interaction with a human supervisor. To this end we adopted the perspective of state-of-the-art recognition pipelines that are based on coding-pooling architectures to extract discriminative yet robust representation for the visual input. Specifically, we proposed two novel contributions to the coding-pooling strategy which reflect some aspects of the robotics setting and allowed us to improve recognition results while maintaining computational efficiency: first, with in mind the real-time constraints of the robotics application, we proposed Best Code Entries (BCE), a coding method that efficiently controls the sparsity of the visual representation, improves its robustness to small variations in the image and at the same time is extremely fast; second, we proposed a novel pooling operator, Mid-Level Classification Weights MLCW), which exploits data supervision to account for local properties of the object of interest, achieving robustness to clutter and improving overall recognition performance.

We investigated the effects of our proposed methods both separately and when combined, and compared them with state-of-the-art competitors. Our approach is generally equivalent or better than other approaches, with the further advantage of begin remarkably fast to compute both during the training and testing phases.

We implemented the general visual recognition architectures and the proposed methods within the framework of the iCub development community, making our results available for the community to use and build upon.

Future work will focus the use of different low level descriptors, possibly learned from the data like in deep-learning architectures, the use of 3D vision for invariant representations and the exploitation of the temporal component to improve the recognition rates.

Acknowledgment

This work was supported by the European FP7 ICT project No. 288382 (POETICON++), project No. 270273 (Xperience) and project No. 611909 (KoroiBot).

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. In *TPAMI*, 2013.

[2] J. Aleotti, D. Lodi Rizzini, and S. Caselli. Object categorization and grasping by parts from range scan data. In *ICRA*, 2012.

[3] F. Anselmi, J.Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio. Magic materials: a theory of deep hierarchical architectures for learning sensory representations. *CBCL Technical Report, Massachusetts Institute of Technology*, 2013.

[4] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Vangool. Speeded-up robust features. *CVIU*, 110:346–359, 2008.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359, 2008.

[7] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Depth kernel descriptors for object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[8] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013.

[9] T. Botterill, S. Mills, and R. Green. Speeded-up bag-of-words algorithm for robot localisation through scene recognition. In *IVCNZ*, 2008.

[10] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[11] Y.-L. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun. Ask the locals: multi-way local pooling for image recognition. In *ICCV*, 2011.

[12] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVA*, 2011.

[13] Q. Chen, Z. Song, Z. Hua, Huang Y., and S. Yan. Hierarchical matching with side information for image classification. In *CVPR*, 2012.

[14] C. Ciliberto, S.R. Fanello, M. Santoro, L. Natale, G. Metta, and L. Rosasco. On the impact of learning hierarchical representations for visual recognition in robotics. In *IROS*, 2013.

[15] C. Ciliberto, U. Pattacini, L. Natale, F. Nori, and G. Metta. Reexamining lucas-kanade method for real-time independent motion detection: Application to the icub humanoid robot. In *IROS*, 2011.

[16] Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. The MOPED framework: Object Recognition and Pose Estimation for Manipulation. *IJRR*, 2011.

[17] Alvaro Collet Romea, Dmitry Berenson, Siddhartha Srinivasa, and David Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *ICRA*, 2009.

[18] G. Csurka, C.R. Dance, L. Fan, J. Willamowski, and C. BrayLixin. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[21] A. Destrero, C. De Mol, F. Odone, and Verri A. A sparsity-enforcing method for learning face features. *IP*, 18:188–201, 2009.

[22] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[23] S. Ekvall, D. Kragic, and F. Hoffmann. Object recognition and pose estimation using color cooccurrence histograms and geometric modeling. In *Image Vision Computing*, 2003.

[24] K. Eunyoung and G. Medioni. 3d object recognition in range images using visibility context. In *IROS*, 2011.

[25] M. Everingham, L. Gool, C.K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[26] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012.

[27] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*,

2000.

[28] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9, 2008.

[29] S. R. Fanello, C. Ciliberto, L. Natale, and G. Metta. Weakly supervised strategies for natural object recognition in robotics. *ICRA*, 2013.

[30] S. R. Fanello, N. Noceti, G. Metta, and F. Odone. Multi-class image classification: Sparsity does it better. *VISAPP*, 2013.

[31] S. R. Fanello, N. Noceti, G. Metta, and F. Odone. Dictionary based pooling for object categorization. *VISAPP*, 2014.

[32] S.R. Fanello, C. Ciliberto, N. Noceti, G. Metta, and F. Odone. Ask the image: supervised pooling to preserve feature locality. In *CVPR*, 2014.

[33] S.R. Fanello, C. Ciliberto, M. Santoro, L. Natale, G. Metta, L. Rosasco, and F. Odone. icub world: Friendly robots help building good vision data-sets. In *CVPRW*, 2013.

[34] L Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVPRW*, 2004.

[35] L. Fei-fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531, 2005.

[36] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric lp-norm feature pooling for image classification. In *CVPR*, pages 2609–2704, 2011.

[37] D. Filliat. A visual bag of words method for interactive qualitative localization and mapping. In *ICRA*, 2007.

[38] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981.

[39] I. Gordon and D.G. Lowe. What and where: 3d object recognition with accurate pose. In *Lecture Notes in Computer Science*, 2006.

[40] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, 43(5):1318–1334, 2013.

[41] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. In *CVPR*, 2010.

[42] K. Huang and S. Aviyente. Wavelet feature selection for image classification. *IP*, 17:1709–1720, 2008.

[43] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[44] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurences in image retrieval: the benefit of pca and whitening. In *ECCV*, 2012.

[45] Y. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *CVPR*, pages 3370–3377, 2012.

[46] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[47] S. Kong and D. Wang. A dictionary learning approach for classification: separating the particularity and the commonality. In *ECCV*, 2012.

[48] P Koniusz, F Yan, and K Mikolajczyk. Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection. *CVIU*, 117, 2013.

[49] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[50] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[51] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006.

[52] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007.

[53] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[54] M. Marszalek, C. Schmid, H. Harzallah, and J. Van De Weijer. Learning Object Representations for Visual Object Class Recognition. Visual Recognition Challange Workshop, in Conjunction with ICCV, 2007.

[55] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The iCub Humanoid Robot: An Open Platform for Research in Embodied Cognition. In *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages

50–56. ACM, 2008.

[56] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *TPAMI*, 2005.

[57] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[58] L. Natale, F. Nori, G. Metta, M. Fumagalli, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz, and G. G. Sandini. *The iCub platform: a tool for studying intrinsically motivated learning.* springer-verlag, 2013.

[59] B A Olshauser and D J Field. Sparse coding with an over-complete basis set: a strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[60] Edgar E. Osuna, Robert Freund, and Federico Girosi. Support vector machines: Training and applications. *A.I. MEMO 1602, MIT A. I. LAB*, 1997.

[61] J Philbin, O Chum, M Isard, J Sivic, and A Zisserrman. Lost in quantization: improving particular object retrieval in large scale image databases. In *Proc of the IEEE CVPR*, 2008.

[62] T. Poggio. The computational magic of the ventral stream. *Nature*, 2012.

[63] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *JMLR*, 2014.

[64] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *ECCV*, 2012.

[65] J. Sanches, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. In *IJCV*, 2013.

[66] J Sanchez, F Perronnin, and T de Campos. modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33, 2012.

[67] M. Schwarz, H. Schulz, and S. Behnke. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *IEEE International Conference on Robotics and Automation (ICRA), 2015*, 2015.

[68] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *PAMI*, 29:411–426, 2007.

[69] Y. Tang, Y.Q. Zhang, N.V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. *Systems, Man, and Cybernetics*, 2009.

[70] G. Taylor and L. Kleeman. Fusion of multimodal visual cues for model-based object tracking. In *ACRA*, 2003.

[71] V. Tikhanoff, U. Pattacini, L. Natale, and G. Metta. Exploring affordances and tool use on the icub. *Humanoids*, 2013.

[72] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, Inc., 1998.

[73] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. *ICCV*, 2007.

[74] A. Vedaldi and B. Fulkerson. Vlfeat - an open and portable library of computer vision algorithms. In *ACM Int. Conf. on Multimedia*, 2010.

[75] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[76] P. Viola and M.J. Jones. Robust real-time face detection. *IJCV*, 57:137–154, 2004.

[77] S. V. N. Vishwanathan, Z. Sun, N. Theera-Ampornpunt, and M. Varma. Multiple kernel learning and the SMO algorithm. *Adv. in Neural Information Processing Systems*, 2010.

[78] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[79] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[80] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. *CVPR*, 2010.

[81] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[82] J. Yang, K. Yu, and T.S. Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010.

[83] Kai yuh Hsiao, S. Vosoughi, S. Tellex, R. Kubat, and D. Roy. Object schemas for responsive robotic language use. In *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pages 233–240, 2008.

[84] X. Zhou, K. Yu, T. Zhang, and T.S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.