Author's version — provided for personal and academic use. Do not redistribute.

Self-Supervised Cross-Modal Online Learning of Basic Object Affordances for Developmental Robotic Systems

Barry Ridge, Danijel Skočaj, and Aleš Leonardis
Faculty of Computer and Information Science
University of Ljubljana, Slovenia
{barry.ridge, danijel.skocaj, ales.leonardis}@fri.uni-lj.si

***Abstract*— For a developmental robotic system to function successfully in the real world, it is important that it be able to form its own internal representations of affordance classes based on observable regularities in sensory data. Usually successful classifiers are built using labeled training data, but it is not always realistic to assume that labels are available in a developmental robotics setting. There does, however, exist an advantage in this setting that can help circumvent the absence of labels: co-occurrence of correlated data across separate sensory modalities over time. The main contribution of this paper is an online classifier training algorithm based on Kohonen's learning vector quantization (LVQ) that, by taking advantage of this co-occurrence information, does not require labels during training, either dynamically generated or otherwise. We evaluate the algorithm in experiments involving a robotic arm that interacts with various household objects on a table surface where camera systems extract features for two separate visual modalities. It is shown to improve its ability to classify the affordances of novel objects over time, coming close to the performance of equivalent fully-supervised algorithms.**

I. Introduction

The term *affordance*, introduced by Gibson [1], is used to characterise the action possibilities that an environment offers an agent acting within that environment. In this paper we address the issue of object affordance learning in a developmental robotic system by developing a self-supervised classifier that operates across two different sensory modalities mediated by object interactions. The main idea behind this is illustrated in Fig. 1 and Fig. 2(a). In our scenario, a robotic arm is mounted on a table surface while camera systems observe the scene. Objects are placed in the workspace where the arm is allowed to interact with them using pushing actions. Object features (e.g. shape features) derived from image data taken prior to arm-object interaction provide data for the first sensory modality, hereby referred to as the *input modality*. After an action has been initiated on an object, video footage is recorded of the object in motion and effect features are extracted from the video footage, forming the basis of the *output modality*. Often when different sensory modalities (or stimulus modalities) are discussed in the literature, they tend to be modalities from different sensory systems, e.g. auditory and visual. Here, instead, we consider two different sensory modalities

This research has been supported by: EU FP6 project VISIONTRAIN (MRTN-CT-2004-005439), EU FP7 project CogX (ICT-215181), and Research program P2-0214 Computer Vision (Republic of Slovenia).

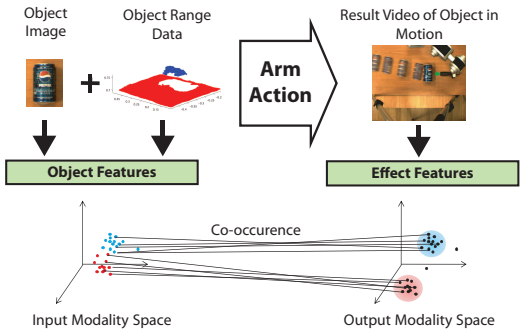
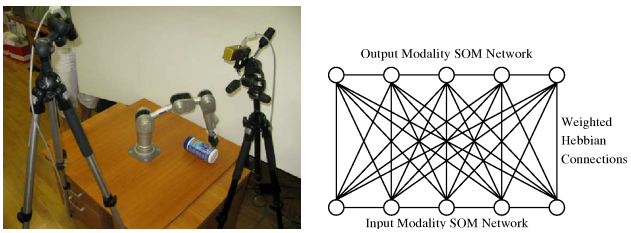


Fig. 1. The main idea of our affordance learning framework.

from the same sensory system (visual) dealing with shape and motion respectively. Though there is a temporal delay when gathering data from each modality during an interactive episode, for the purposes of our discussion here we consider such data in each modality to be *co-occurrences*. Given a series of interactive episodes, the learning task is to find clusters in the output modality feature space that may be identified as affordance classes, and use them to train a classifier in the input modality space. Thus, when the system encounters novel objects, it can predict their affordance classes by observing their respective object features.

The main contribution of our algorithm is that it removes the need for class labels of any kind during the training stage by introducing a probabilistic heuristic based on the co-occurrence information. When designing the algorithm, we were subscribing to an online learning paradigm suitable for developmental robotic systems. Requirements for such an algorithm include: 1) No, or limited, access to previously viewed training samples. 2) An incremental training mechanism. 3) Fixed, or limited, memory requirements. To meet these criteria, we use a cross-modal neural network, as in Fig. 2(b) consisting of two layers of codebook vectors fully connected via a Hebbian weight mapping. The codebook vector layers are of a fixed size, thus meeting the third online learning criterion. The learning algorithm that we present, through the use of Kohonen's *self-organizing map (SOM)* [2] method, as well as a variation of Kohonen's *learning vector quantization (LVQ)* [2], does not require access to previously viewed training samples and can be trained incrementally, thus satisfying the first and second criteria.

Author's version — provided for personal and academic use. Do not redistribute.



(a) (b)

Fig. 2. (a) Experimental setup. (b) Our cross-modal neural network. Two SOM networks operating in separate modalities, fully connected via a weighted Hebbian mapping.

Perhaps the most closely related work in the literature with respect to affordance learning to our work is by Fitzpatrick *et al.* [3], [4]. The authors trained a humanoid robot to recognize "rolling" affordances of 4 household objects using a fixed set of actions to poke the objects in different directions as well as simple visual descriptors for object recognition. There are two main differences between their method and ours. Firstly, in [3], [4], the feature associated with the rolling direction affordance was pre-determined, whereas in our system, the learning algorithm is provided with a number of different output features and it must determine for itself the affordance classes within that feature space. Secondly, their system used object recognition to identify the affordances of individual objects, whereas our system determines the affordance class of objects (grounded in output modality features) based, not on their individual identity, but on a broad set of input features (e.g. shape). In [5], the authors used a humanoid robot to push objects on a table and used a Bayesian network to form associations between actions, objects and effects. Though quite similar to our approach, their learning method may not be as amenable to full online learning, as they have to gather a certain amount of data initially to form categories within the various modalities before the network can be trained.

Saxena *et al.* [6], [7] used the same robotic arm used in our work (see Sec. III-A) to attempt to grasp novel objects based on a probabilistic model trained on synthetic images of various other objects labeled with grasp points. What differs in our work is that rather than training our learning algorithm on synthetically generated object examples, we train on interactions with real objects. Moreover, their two possible affordances were specified in advance: graspable or non-graspable, whereas our system generates its own affordance classes through interaction with objects. In [8], the authors describe a mobile robotic system equipped with a 3D laser scanner that learns to perceive traversability affordances of various objects, such as spheres, cylinders and boxes in a room. The robot was provided with a set of seven possible actions and used its range scanner to gather angle and distance features aggregated over a grid-division of the range image. It then learned mappings between environmental situations and the results of its actions by first selecting relevant features from the full set, then using support vector machines to classify the relevant features into affordance categories. Though good results were acheived, the affordance categories were, again, pre-defined: traversable or non-traversable.

With regard to our learning algorithm, one of the first examples of Hebbian-linked SOMs was provided in [9], where they were used for developing an artificial neural network model of the mental lexicon. The structure of the network in [9] is identical to the one presented here: two SOMs fully connected via weighted Hebbian mappings. Moreover, the training scheme presented in [9] is the same as our phase 1 training (see Sec. II-B). However, this training scheme by itself, is not optimized for classification purposes as we shall see later in Sec. V, and the SOMs do not influence each other during training. de Sa *et* al. [10], [11] greatly improved upon this by creating a cross-modal neural network where two competitive learning maps in each modality influenced each others' training by learning to agree upon common class labels for co-occurring data samples. Similarly to us, they employed LVQ to train the maps in each modality based on the class information. One drawback, however, is that the class labels have to be determined a priori and maintained throughout the training process. A SOM with a Hebbian learning mechanism called a *Growing When Required (GWR)* network was used in [12] to aid a simulated mobile robot in learning affordances of objects with survival values such as nutrition and stamina so that it could prosper over time in its environment. The SOM was used to cluster visual sensor data in the input space where nodes were assigned weights based on the success or failure of actions. While our method also uses SOM training, in our case it is used on both the output data, where the nodes are meta-clustered to form affordance classes, and the input data, where at a certain point it is swapped for a variation of LVQ which is better suited for classification optimization.

II. The Learning Algorithm

A classifier could be constructed in either of these modalities by attaching class labels to the training data and employing a supervised learning algorithm, but this is hardly ideal for an autonomous cognitive system like a robot since it assumes the existence of an external tutor who is willing to label the data. However, once we have noted that in this type of learning scenario correlated training data *co-occur* in each modality, this opens up some alternative possibilities. For example, k-means clustering or density estimation might be possible in the joint feature space, however, as was discussed in [11], these are not ideal solutions. The problem with simple k-means clustering or competitive learning in the joint space is that all feature dimensions would be required for the classification of test samples; these methods would not be able to marginalize over the missing dimensions when trying to predict the outcome of one modality from another. Density modeling would account for this problem, but requires fitting many parameters which would become infeasible in high dimensions. Moreover, neither approach complies naturally to our online learning criteria. Thus, perhaps a better approach would be to use the natural structure of the data in one modality, as well as the co-occurrence

Author's version — provided for personal and academic use. Do not redistribute.

information, to train a supervised classifier in the other modality. This could be accomplished, for example, by using an unsupervised clustering algorithm like k-means to derive clusters in one modality which could be used as class labels to train a supervised classifier in the other ([11] provides a similar approach). However, it is computationally expensive to cluster at every training step in an online algorithm. One alternative, that of clustering early and maintaining the clusters over time, could potentially introduce inaccuracies as training progresses if the sample distribution changes significantly. In the following we describe a cross-modal neural network and a two-phased training scheme that aims to address these issues.

A. Cross-modal Neural Network

The structure of our cross-modal neural network is illustrated in Fig. 3 (a). Two codebook vector layers, one in the input modality and one in the output modality, are fully connected to each other via a weighted Hebbian mapping. The idea is that the codebook vector layers are trained to form a representation of the information contained in their respective modalities, while the Hebbian mapping is trained on the basis of the co-occurrence of data across these modalities.

Learning proceeds in two phases. In the first phase of training, as training samples for each modality are concurrently presented to the network, the codebook vector layers are trained separately using the usual SOM algorithm (described below). While training is ongoing, the Hebbian links that connect the best-matching unit nodes in each of the codebook vector layers are then updated appropriately based on co-occurence. Though SOM training is good for producing low-dimensional represenations of data distributions, it is not the best solution for optimizing decision borders, thus we employ a second phase of training that exploits the co-occurence information captured by the Hebbian mapping between the codebook vector layers. In the second phase, the codebook vector layer in the output modality continues to be trained in the usual way, as does the Hebbian mapping, while the codebook vector layer in the input modality is trained using our variation of LVQ. Rather than using class labels, the LVQ training rules are selected using a Hellinger distance-based heuristic that exploits the cross-modal Hebbian mapping to indicate whether a given codebook vector is of the "correct" or "incorrect" class for a given training sample.

A classifier can then be formed after training by performing unsupervised meta-clustering over the output modality nodes in order to form class labels, although it should be emphasised that these class labels are not required during training. It should also be noted that the algorithm, in largely unmodified form, could also perform regression, though results for this are not presented in this paper. The training and classification processes are described in more detail in the following sections.

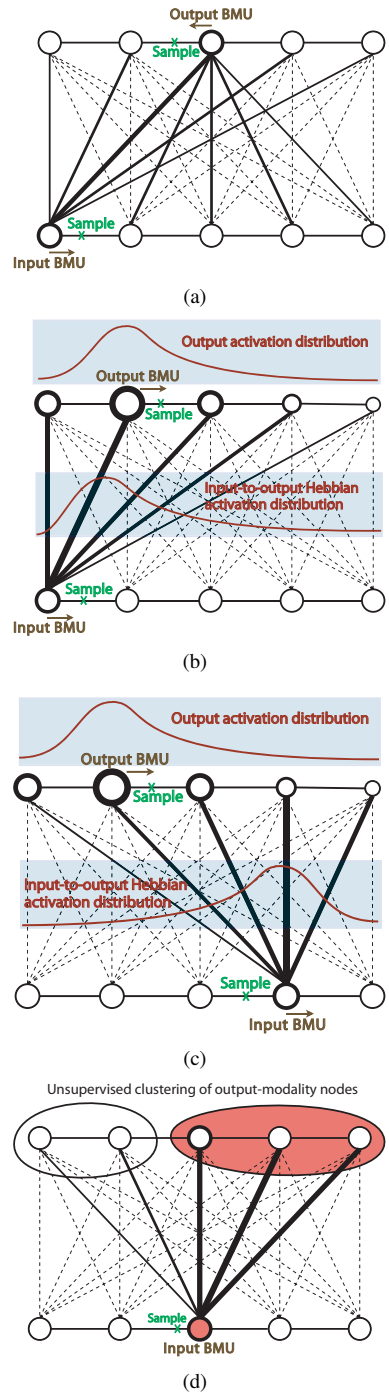


Fig. 3. Training and classification. In each figure, the bottom layer is the input modality codebook & the top layer is the output modality codebook. Arrows indicate node movement during training. (a) Phase 1 training: regular SOM training performed in each modality while Hebbian links are also updated. (b) Phase 2 training: Hellinger distance heuristic indicates correct training sample classification based on activation distributions. Input best-matching unit node (BMU) is moved towards the training sample. (c) Phase 2 training: heuristic indicates incorrect sample classification because the activation distributions differ significantly. Input BMU is moved away from the training sample. (d) Classfication: output nodes are meta-clustered and the cluster with the strongest weighted connections to the input BMU wins.

B. Training: Phase 1

The following two sub-sections describe how both the individual modality networks and the Hebbian mapping that

Author's version — provided for personal and academic use. Do not redistribute.

connects them are trained in the first phase of training. Phase 1 training is illustrated in Fig. 3 (a).

1) Modality Codebook Vector Layers: In the first phase, network training in each modality proceeds in accordance with Kohonen's original SOM formulation [2] which we summarise here. The nodes of the network layers of each modality contain codebook vectors $\mathbf{m}_i = [m_{i1}, \ldots, m_{id}]$, where d is the dimensionality of the modality feature vectors, that are randomly initialized before training begins. At each training step, a data vector $\mathbf{x} = [x_1, \ldots, x_d]$ is is measured against each codebook vector using the Euclidean distance metric, as follows:

$$||\mathbf{x} - \mathbf{m}_i||^2 = \sum_{j=1}^{d} w_j (x_j - m_{ij})^2, \tag{1}$$

where w_j is an element of weight vector $\mathbf{w} = [w_1, \ldots, w_d]$ which is used for a feature selection algorithm in the second phase of training, described in Sec. II-C. The node that is closest to the input data vector based on this metric is called the *best matching unit (BMU)* and both it and its neighbouring nodes are updated using the following update rule

$$\mathbf{m}_i(t{+}1) = \begin{cases} \mathbf{m}_i(t) + \alpha_{\text{SOM}}(t)\left[\mathbf{x}(t) - \mathbf{m}_i(t)\right] & \text{if } i \in N_c, \\ \mathbf{m}_i(t) & \textit{otherwise}, \end{cases} \tag{2}$$

operating over all $i \in [1, n]$, where $\alpha_{\text{SOM}}(t)$ is the learning rate at time t and N_c is the neighbourhood around the BMU c.

2) Cross-modal Hebbian Mapping: The following Hebbian weight training procedure is taken from Miikulainen [9]. In order to train the Hebbian mapping, we require a measurement of the activation of a given modality layer node a_i, formulated as follows:

$$a_i(t) = \begin{cases} 1 - \frac{||\mathbf{x}(t) - \mathbf{m_i}(t)|| - d_{\text{min}}}{d_{\text{max}} - d_{\text{min}}} & \text{if } i \in N_c, \\ 0 & \textit{otherwise}, \end{cases} \tag{3}$$

where d_{min} is the smallest and d_{max} the largest distance of $\mathbf{x}(t)$ to a unit in the neighbourhood.

During training, the link weight changes are made proportional to the product of the activation of the two nodes in each modality that are being associated, as follows:

$$\Delta h_{kl}(t) = \alpha_{\text{HEB}}(t) a_k(t) a_l(t), \tag{4}$$

where Δh_{kl} is the unidirectional associative weight leading from node k in the input modality layer to node l in the output modality layer, $\alpha_{\text{HEB}}(t)$ is the Hebbian learning rate, and $a_k(t)$ and $a_l(t)$ are the activations on the two nodes at time t. Each link weight h_{kl} is then updated using the following normalization equation:

$$h_{kl}(t+1) = \frac{h_{kl}(t) + \Delta h_{kl}(t)}{\sqrt{\sum_l \left[h_{kl}(t) + \Delta h_{kl}(t)\right]^2}}. \tag{5}$$

The Hebbian mapping, when trained as above, provides a type of memory of previous training experience in terms of modality co-occurrences, or of what one modality "looks like" from the perspective of the other. As we shall see, this information can be effectively employed to augment classifier training.

C. Training: Phase 2

After Phase 1 training has proceeded for a reasonable amount of time, i.e., long enough to provide a robust Hebbian mapping, Phase 2 training may be initiated. In the algorithm presented in this paper, the network layer in the output modality continues to be trained with the usual SOM algorithm in Phase 2. The network layer in the input modality however, switches to a modified version of learning vector quantization (LVQ) [2] training that employs a probabilistic heuristic. In order to develop this heuristic, we require the Hellinger distance metric which we discuss next. Phase 2 training is illustrated in Figs. 3 (b) and 3 (c).

1) Hellinger Distance Heuristic: Using the definition from [13], for a countable state space Ω, given probability measures μ and ν,

$$d_H(\mu, \nu) := \left[\sum_{\omega \in \Omega} \left(\sqrt{\mu(\omega)} - \sqrt{\nu(\omega)}\right)^2\right]^{\frac{1}{2}}. \tag{6}$$

We use the Hellinger distance metric as defined above to create a heuristic that allows us to measure the similarity between nodes in the input modality layer and the output modality layer with respect to the Hebbian mapping. The Hellinger distance takes values in the bounded interval $[0, \sqrt{2}]$, making it amenable to statistical analysis, e.g. calculating mean distance. Given input modality node k, we define

$$f_k(t) = \{h_{kl}(t) : \forall l \text{ in the output modality layer}\}, \tag{7}$$

or all the Hebbian link weights that connect node k in the input modality layer to the nodes in the output modality layer at time t. We define

$$g(t) = \{a_l(t) : \forall l \text{ in the output modality layer}\}, \tag{8}$$

or all the node activations in the output modality layer at time t.

f_k can be thought of as a distribution of the Hebbian map activity from node k in the input layer projected onto the output layer. Loosely put, this gives us a picture of what the output map looks like from the perspective of node k in the input map based on previous training experience. g, on the other hand, gives us a distribution of the output map activity with respect to the current training sample. Thus, when given a training sample for the input modality, if we employ the metric $d_H(f_k(t), g(t))$, we can get an impression of how well its best matching unit node in the input modality layer predicts the activity of the output modality layer given its co-occurring training sample. This heuristic can of course be used in the opposite direction, from the output modality layer to the input modality layer, but for the algorithm we present in this paper it is employed strictly in the above way to augment the training of the input modality layer. Now that we have the necessary tools in place, we may proceed to present our modified LVQ algorithm.

Author's version — provided for personal and academic use. Do not redistribute.

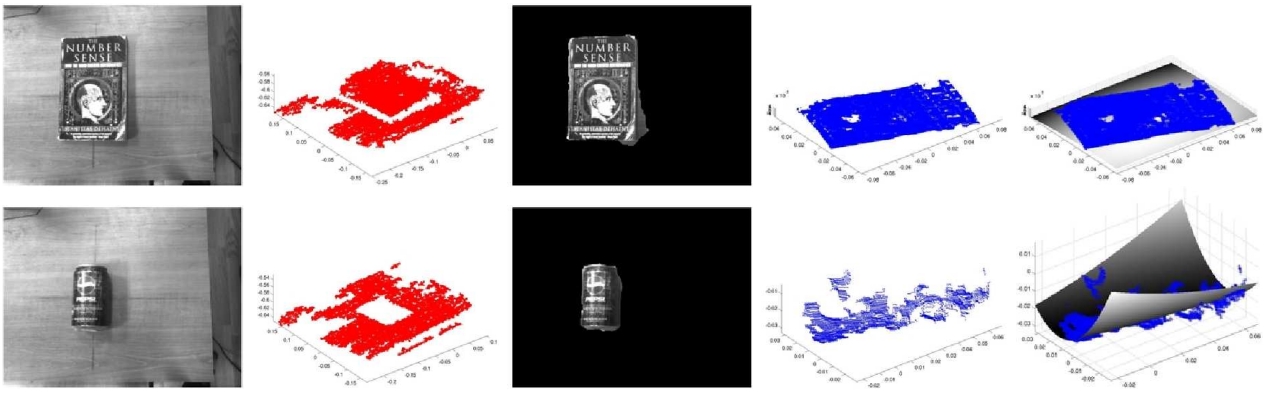


Fig. 4. Examples of image and range data taken with the stereo camera for two different types of objects: a book which slides when pushed by the robotic arm, and a Pepsi can which rolls when pushed by the arm. From left to right: intensity image, range data of the scene, segmented object, segmented object range data, object range data with a fitted quadric surface.

2) Learning Vector Quantization Without Class Labels: In traditional LVQ training [2], codebook vectors are given fixed class labels a priori. Subsequently, as training samples are presented with accompanying class labels, the best matching codebook vectors are updated according to a set of rules. If the best matching codebook vector class label matches that of the training sample, the codebook vector is moved towards the sample. If the labels do not match, the codebook vector is moved away from the sample.

In the modified form of LVQ we present here, which is the main contribution of this paper, we refrain from labeling the codebook vectors altogether. Codebook vectors are updated based on the heuristic presented in the previous section. Given the best matching node c in the input modality layer for a given input modality training sample x, we apply the following update rule:

$$\mathbf{m}_c(t+1) = \begin{cases} \mathbf{m}_c(t) + (1-\gamma)\alpha_{\mathrm{LVQ}\,c}\,[\mathbf{x}(t) - \mathbf{m}_c(t)] \\ \textit{if } d_H(f_c(t), g(t)) < \epsilon \\ \\ \mathbf{m}_c(t) + (\gamma-1)\alpha_{\mathrm{LVQ}\,c}\,[\mathbf{x}(t) - \mathbf{m}_c(t)] \\ \textit{otherwise}, \end{cases} \tag{9}$$

where, assuming $d_H(f_c(t), g(t))$ is normalised, ϵ is usually set to its mean value over all t, and γ is set to either 0 or $2d_H(f_c(t), g(t))$ if the rule is to be applied in either a binary or fuzzy fashion respectively.

In more simple terms, the effect of applying the above rule is that, when the output modality appears to have the same activity distribution as predicted by the best matching node in the input modality based on past experience, the best matching node in the input modality is moved closer to the training sample. Conversely, if the output modality appears to have a significantly different activity distribution, it is moved away from the training sample. These two alternative cases are visualised in Figs. 3 (b) and 3 (c).

Note that the above also incorporates optimized-learning-rate learning vector quantization (OLVQ), where separate $\alpha_{\mathrm{LVQ}\,c}$ are stored and updated for each node. See [2] for more details.

3) Feature Selection: As alluded to in Sec. II-B.1, we also employ a feature selection method to boost classifier training. We use the relevance determination learning vector quantization (RLVQ) algorithm from [14] to do this. Given the best matching node c in the input modality layer for a given input modality training sample x at time-step t, we perform the following operation on the w_j from (1). For each feature dimension j:

$$w_j(t+1) = \begin{cases} \max\{w_j(t) - \alpha_{\mathrm{F}}(t)|\mathbf{x}(t) - \mathbf{m}_c(t)|,\, 0\} \\ \textit{if } d_H(f_c(t), g(t)) < \epsilon \\ \\ w_j(t) + \alpha_{\mathrm{F}}(t)|\mathbf{x}(t) - \mathbf{m}_c(t)| \\ \textit{otherwise}. \end{cases} \tag{10}$$

We then normalise, as follows: for all j, $w_j := w_j/|\mathbf{w}|$.

D. Classification

After training, a classifier may be formed from the network by performing unsupervised meta-clustering over the nodes of the output modality codebook layer. To this end, we used k-means with automatic selection of k based on votes from the following validity indices: Davies-Bouldin [15], Calinski-Harabasz [16], Dunn [15], Krzanowski-Lai [16] and the silhouette index [16], [15]. These clusters define the output modality categories to be used for classification purposes. Given an input modality test sample to be classified, the best matching node in the input modality layer is found and its Hebbian weight links are mapped to the output modality layer. The weights for the links connecting to each cluster are summed, and the cluster with the highest score is deemed to be the winning class for that input test sample.

III. System Architecture & Setup

A. Robotic Arm

In our system, we use a Neuronics Katana 6M robotic arm which features 5 DC motors for main arm movement, as well as a 6th motor to power a 2 fingered gripper that houses both infrared and haptic sensors (note: these sensors are not used in the experiment presented here). The base of

Author's version — provided for personal and academic use. Do not redistribute.

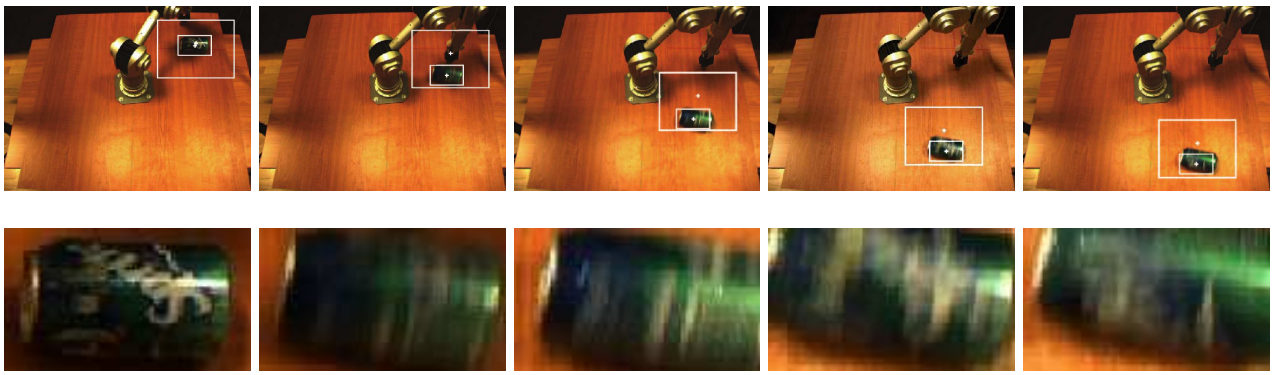


Fig. 5. An example of the object tracking mechanism described in Sec. IV-B using the images in the first row show a progression of frames tracking a Sprite can being pushed by the arm. The outer rectangle is a likelihood window around the object obtained using the particle filter tracker. The inner rectangle is the result of using histogram back-projection within that window to localise the object. The second row of close-up images shows how the appearance of the object within the inner rectangle changes during the course of object motion.

the arm is mounted on a flat table with a wooden laminate surface, and the arm is allowed to move freely in the area above the table surface, avoiding collisions with the table through the use of specialized control software. The arm control software that was used for this work is a modified version of Golem [1], control software for the Katana arm. Given desirable parameters, Golem uses forward kinematics to generate arm joint orientations and motion paths, then uses cost functions and searches to select the ones that most closely fit the parameters. In order to ensure that the actions, and by extension the object affordances, that are available to the system are as consistent and learnable as possible, we optimized for a linear end-effector motion trajectory when moving between workspace positions.

B. Camera Systems

2 Point Gray Research cameras- the Flea monocular camera (640x480 @ 60FPS or 1024x768 @ 30FPS) and the Bumblebee 2 grayscale stereo camera (640x480 at 48FPS or 1024x768 at 20FPS) were used to gather intensity images, range data and video for the experiment listed in Sec. V.

IV. Visual Feature Extraction

A. Input Modality Feature Extraction

With regard to the input modality features, for the purposes of this particular affordance learning scenario, we are mostly interested in extracting features that describe the global shape of an object as they are likely to be most relevant for determining how the object will behave. However, in theory, any types of features that describe properties of the objects under consideration could be used here.

1) Range Data: We have developed a method for segmenting the object from range images that uses RANSAC (RANdom SAmple Consensus) [17] to fit a plane to the table surface for removal, then mean-shift clustering [18] as well as a graph-cut segmentation in the corresponding intensity image to isolate the object range data with minimal noise. The graph-cut segmentation method we used was from [19], which uses the min-cut/max-flow algorithms oulined in [20], [21], [22] to apply the standard graph cut technique to segmenting multimodal tensor valued images. A quadratic surface may then be fitted to the object range data to derive curvature features from the object surface. We derive 2 curvature features in this way from the coefficients of the polynomial of the fitted quadratic surface that provide a good description of the global curvature of the object. This surface fitting technique is illustrated on the two objects shown in Fig. 4.

2) Image Data: The segmentation technique produces reasonably good intensity image segmentations of objects. These are then used to calculate the following 10 shape features: area, convex area, eccentricity, equivalent cicular diameter, Euler number, extent, filled area, and the major axis length.

B. Output Modality Feature Extraction

After an arm action has been performed on an object, the resulting videos of the interaction are processed for output modality features. This is primarily acheived by tracking the object in motion using a probabilistic tracker from [23]. This tracker is in essence a colour-based particle filter, which also makes use of background subtraction using a pre-learned background image. Background subtraction by itself is insufficient to localise the object in our experimental setup due to changes in lighting and the motion of the arm, but it is helpful in reducing ambiguities for the tracker. Object shapes are approximated by elliptical regions, while their colour is encoded using colour histograms. The dynamics of objects are modeled using a dynamic model from [24], which allows for tracking with a smaller number of particles, and consequently, near real-time tracking performance.

1) Global Object Motion Features: The following 9 features are calculated from the particle filter tracker output data: total distance traveled in x-axis, total distance traveled in y-axis, total Euclidean distance traveled, mean velocity in

[1]Golem was developed by researchers at the University of Birmingham who kindly provided us with a copy for our research. More information can be found at: http://www.cs.bham.ac.uk/~msk/ .

Author's version — provided for personal and academic use. Do not redistribute.

x-axis, mean velocity in y-axis, velocity variance in x-axis, velocity variance in y-axis, final x position, final y position.

2) Object Appearance Changes: To estimate how the appearance of the objects change during motion, we chose to calculate the average difference of both colour and edge histograms between video frames of the objects, the aim being to detect both motion blur and the texture changes characteristic of many rotating objects. This required an extension to the particle filter tracker previously described. The tracker by itself is sufficient for tracking the motion of objects, but it is slightly inaccurate at times. For example, if an object is rolling and stops suddenly, the tracker sometimes briefly overshoots the object before returning to it a few frames later. To avoid this, we use the output of the tracker to define a broad window around the object in the video frames, before using colour histogram back-projection [25] to localise the object within the window. Histogram difference averages are then calculated from the start of object motion until the end. See Fig. 5 for sample frames from an interaction with an object that illustrates this technique at work. We derive 3 output modality features from this procedure: average colour histogram difference, average edge histogram difference, and the product of these two values.

V. Experiments

To test our affordance learning system, the experimental environment was set up as previously described and as shown in Fig. 2(a). During experiments, objects were placed at a fixed starting position prior to interaction. Two cameras were used to provide both sufficiently detailed close-up range data of the object surfaces and a sufficiently wide field of view to capture object motion over the entire work area. To achieve this, the stereo camera was positioned above the object start position, while the monocular camera was positioned at a higher position in front of the workspace.

We selected 8 household objects to be used in the experiments: 4 flat-surfaced objects; a book, a CD box, a box of tea and a drink carton, and 4 curved-surfaced objects; a box of cleaning wipes, a Pepsi can, a Sprite can and a tennis ball box. Each of these objects was placed centred at the start position with a consistent orientation, and the robotic arm pushed the object at a fixed speed using a fixed pushing action. During trials, the curved objects would tend to roll after being pushed, whereas the flat objects would stop suddenly after the push. Before an action was performed on an object, both intensity and range images were gathered from the stereo camera. This data was then processed to produce the 12 input modality features discussed in Sec. IV-A. After an action was performed on an object, images were gathered and passed to the tracking system described in Sec. IV-B to produce 12 output modality features.

To evaluate the algorithm, we first collected a dataset as follows. 20 object push tests were carried out for each of the 8 objects listed previously and the resulting data was processed, leaving 160 data samples. The samples were then hand-labeled with two ground truth labels: rolling and non-rolling. In the following evaluations, leave-one-out cross validation was performed by splitting the dataset into a training set of 140 samples consisting of all data for 7 of the objects and a test set of 20 samples consisting of all data for the remaining object. The classification task was then to train on 7 objects, find the affordance classes in the output modality and try to classify the remaining object on that basis. In the experiments for this paper, the training set was doubled and randomized, effectively allowing for 2 epochs of training over the training set, i.e. training over 280 samples. Cross validation was performed by using each of the 8 objects in turn as the test object and averaging classification scores across all 8 subsequent training and test sets and the 20 test samples contained therein. Fig. 6 shows the results of incrementally cross-validating 6 algorithms every 20 training steps and averaging over 40 trials. In 4 of the algorithms, each of the two codebook vector layers in the input and output modalities contained 100 nodes arranged in a 10×10 hexagonal lattice with a sheet-shaped topology. In the other 2 algorithms, the codebook vector layers contained 5 nodes arranged in a 1×5 linear topology.

The goal of the evaluation was to compare the performance of our self-supervised algorithm to fully-supervised learning using ground truth labels. In the case of the self-supervised algorithms, classification of a test sample was deemed to be correct if the output modality meta-cluster (c.f. Sec. II-D) matched the ground truth (c.f. [26] for more details on matching the meta-clusters to ground truth labels). Of the 6 algorithms evaluated, 2 were variations on fully-supervised LVQ1, OLVQ1 with 100 nodes and OLVQ1 with 5 nodes, while the remaining 4 were variations of self-supervised cross-modal learning. Of the 4 variations of cross-modal learning, one was cross-modal SOM training as in [9] with 100 nodes and the other 3 were modifications of our proposed heuristic-based LVQ algorithm: fuzzy heuristic OLVQ1 with RLVQ feature selection (HeurORLVQ), binary HeurORLVQ with 100 nodes, and binary HeurORLVQ with 5 nodes. For each of these, the initial α_{SOM} learning rate in each modality was set to 1 with a linear profile descending to 0 over the 280 timesteps in the output modality and 140 timesteps in the input modality. The RLVQ α_{F} learning rate was set to a constant 0.1. Training shifted from Phase 1 to Phase 2 halfway through the training set (140 timesteps), In Phase 2, α_{LVQ} was set to a constant 0.3. These learning rates were selected both through trial and error, and as advised by [2].

As can be seen in Fig. 6, the fully supervised algorithms performed the best, as expected, with the 5-node OLVQ reaching a correct classification rate of 97.11% by the end of training, and the 100-node OLVQ reaching a score of 93.53%. Of the self-supervised cross-modal classifiers, 100-node Fuzzy HeurORLVQ performed the best, reaching a correct classification rate of 91.64%, while 100-node HeurORLVQ and 5-node HeurORLVQ reached rates of 90.16% and 86.67% respectively. The cross-modal SOM finished with a score of 81.91%, thus justifying our two-phased learning approach. Our algorithm works best when there are enough nodes in the network to give a decent approximation of the

Author's version — provided for personal and academic use. Do not redistribute.

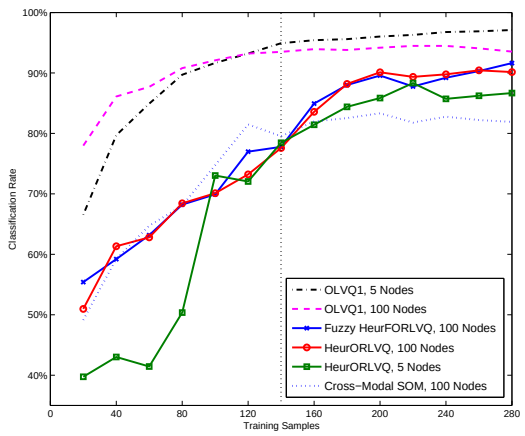


Fig. 6. Incremental leave-one-out cross-validation evaluation averaged over 40 trials (c.f. Sec. V). The switch from Phase 1 training to Phase 2 training (c.f. Sec. II) is indicated by the vertical dotted line.

sample density in the output modality, thus providing a more accurate Hellinger distance heuristic. The results show that, when evaluated with such a number of nodes, our algorithm performs almost as well as fully-supervised OLVQ1 using the same number of nodes. On the other hand, OLVQ1 works best when the number of training steps is 30 to 50 times the number of codebook vectors [2], i.e. for our small training set size of 280 samples, the number of codebook vectors should be low, e.g., 5. However, even when using such a small number of nodes, in this 2-class learning scenario our algorithm performed robustly, and still produced better results than the 100-node cross-modal SOM.

VI. Conclusion

In conclusion, we have presented a robotic system that uses a novel self-supervised cross-modal online classifier training algorithm to learn basic object affordances. We have shown that it can be successfully trained to learn affordances of household objects by interacting with them, and subsequently predict the affordance classes of novel objects by observing their object features, e.g. shape. The experimental results also demonstrated how the system, through the use of the proposed novel algorithm, can start learning with little or no experience, and improve results over time to the point where the classification rate is close to that of a fully-supervised system. Although the results presented here only account for one type of action, multiple classifiers may be trained to account for different types of actions. We aim to improve on this in future work by modifying the algorithm such that actions may be parameterized, perhaps in a separate modality. We would also like to test the algorithm on more challenging problems where there are more than two classes present in the training data.

References

[1] J. J. Gibson. *The ecological approach to visual perception*. Lawrence Erlbaum, 1986.

[2] T. Kohonen. *Self-organizing maps*. Springer, 1997.

[3] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action-initial steps towards artificial cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, volume 3, 2003.

[4] G. Metta and P. Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, 2003.

[5] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.

[6] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng. Robotic grasping of novel objects. In *In Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems Conference*, Vancouver, Canada, 2006.

[7] A. Saxena, J. Driemeyer, and A. Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157, 2008.

[8] E. Ugur, M. R. Dogar, M. Cakmak, and E. Sahin. The learning and use of traversability affordance using range images on a mobile robot. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, page 1721–1726, 2007.

[9] R. Miikkulainen. Dyslexic and Category-Specific aphasic impairments in a Self-Organizing feature map model of the lexicon. *Brain and Language*, 59(2):334–366, 1997.

[10] V. R. de Sa. Learning classification with unlabeled data. In *In Proceedings of the Eight Annual Conference on Neural Information Processing Systems*, volume 93, pages 112–119, 1994.

[11] V. R. de Sa and D. H. Ballard. Category learning through multimodality sensing. *Neural Computation*, 10:1097–1117, 1998.

[12] I. Cos-Aguilera, L. Canamero, and G. Hayes. Using a SOFM to learn object affordances. In *Proceedings of the 5th Workshop of Physical Agents (WAF'04), Girona, Spain*, 2004.

[13] A. L. Gibbs and F. E. Su. On choosing and bounding probability metrics. *International Statistical Review/Revue Internationale de Statistique*, pages 419–435, 2002.

[14] T. Bojer, B. Hammer, D. Schunk, and K. T. von Toschanowitz. Relevance determination in learning vector quantization. In *European Symposium on Artificial Neural Networks*, pages 271–276, 2001.

[15] N. Bolshakova and F. Azuaje. Cluster validation techniques for genome expression data. *Signal Processing*, 83(4):825–833, 2003.

[16] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7), 2002.

[17] M. A Fischler and R. C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[18] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[19] J. Malcolm, Y. Rathi, and A. Tannenbaum. A graph cut approach to image segmentation in tensor space. In *Workshop on Component Analysis Methods (CVPR)*, pages 18–25, 2007.

[20] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[21] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[22] Y. Boykov and V. Kolmogorov. An experimental comparison of Min-Cut/Max-Flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[23] M. Kristan, J. Perš, M. Perše, and S. Kovačič. Closed-world tracking of multiple interacting targets for indoor-sports applications. *Computer Vision and Image Understanding*, 113(5):598–611, 2009.

[24] M. Kristan, J. Perš, A. Leonardis, and S. Kovačič. A hierarchical dynamic model for tracking in sports. In *Proceedings of the Sixteenth Electrotechnical and Computer Science Conference*, September 2007.

[25] M. J Swain and D. H Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[26] B. Ridge, D. Skočaj, and A. Leonardis. Unsupervised learning of basic object affordances from object properties. In *Proceedings of the Fourteenth Computer Vision Winter Workshop*, pages 21–27, Eibiswald, Austria, 2009.