Unsupervised Learning of Basic Object Affordances from Object Properties

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

Affordance learning has, in recent years, been generating heightened interest in both the cognitive vision and developmental robotics communities. In this paper we describe the development of a system that uses a robotic arm to interact with household objects on a table surface while observing the interactions using camera systems. Various computer vision methods are used to derive, firstly, object property features from intensity images and range data gathered before interaction and, subsequently, result features derived from video sequences gathered during and after interaction. We propose a novel affordance learning algorithm that automatically discretizes the result feature space in an unsupervised manner to form affordance classes that are then used as labels to train a supervised classifier in the object property feature space. This classifier may then be used to predict affordance classes, grounded in the result space, of novel objects based on object property observations.

1 Introduction

J.J. Gibson's [10] famous quote on affordances,

> *"The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill."*

deftly encapsulates the broad meaning of the term, as well as hinting at the more subtle aspects of such an idea. There is a unique relationship between an individual cognitive agent and its environment such that affordances will mean different things to different cognitive agents depending on their individual characteristics and their current situation. Thus, when discussing how an agent learns affordances, a number of important considerations present themselves, namely, the morphology, or shape, of the cognitive agent, the morphology of the environment and the objects that inhabit it, the environmental context and object contexts, e.g., how an object is positioned in the environment, the agent context, and the possible actions that are available to the agent in the present context. As indicated in the title, we are chiefly concerned with how a cognitive agent might discover and learn to exploit basic affordances of objects in its environment.

By using its sensors to observe objects and by using its actuators to interact with them, an agent may infer relationships between the morphologies of objects, the different actions used to interact with them and the way they behave during interaction. If the information that is used to infer such relationships is rich enough and these relationships are learned effectively, then it should be possible to generalize over the relationships and aid the agent in making predictions in novel situations by applying such generalized knowledge. In other words, the cognitive system would perform an action, e.g. "push the centre of the object", on similar objects, e.g. a football and a tennis ball, and observe what happens afterwards- in the case of our example, the objects would roll away from the effector because they are round. When the system subsequently encounters an orange, it should be able to infer that it will also roll away when pushed in its centre, based on what it observed in the previous situations pushing similar round objects.

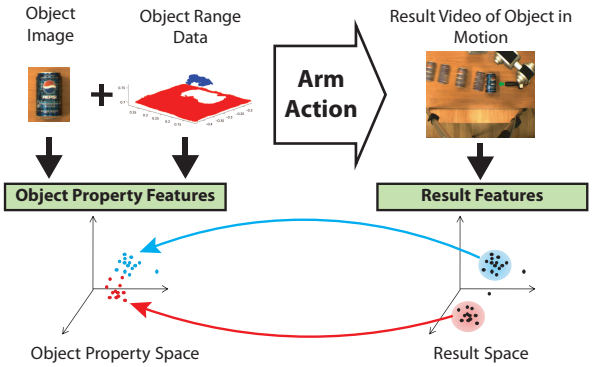

Figure 1: The main idea of our affordance learning framework.

Our working experimental system and environment, shown in Fig. 2, involves a robotic arm attached to a table surface, coupled with both monocular and stereo camera systems that allow for visual observation of objects and interactions in the learning environment. The learning environment itself is the table surface to which the arm is attached and the arm interacts with household objects that are placed on the table surface by pushing them with its two-fingered gripper. The cameras allow for the recording of both still images and video of the objects and the stereo camera in particular, records range data of the objects which may be used to generate 3-dimensional shape features.

The main idea behind our affordance learning framework is illustrated in Fig. 1. During experimental trials, when an object is placed in the working environment, both in-

camera systems. An arm action is then performed on the object and its resulting behaviour is recorded to video. Various object property features are derived from the intensity and range images of the objects, forming an object property feature space, while various result features are derived from the videos of the objects, generating a result feature space. The main task of the affordance learning algorithm as defined in our framework, is to identify significant clusters in the result space and associate these clusters with data in the object property space. This allows for the affordances of novel objects to be broadly classified in terms of result space clusters, by observing their respective object property features and using them as input to a classifier trained by the affordance learning algorithm.

This paper is organised as follows. In the next section we discuss related work, highlighting where appropriate how our work differs. In Section 3 we outline the system architecture and setup. Following that, in Section 4, we discuss the computer vision methods used to extract both the object property features and result features we used in experiments. Section 5 describes our novel affordance learning algorithm. In Section 6, we describe an experiment, as well as various techniques that were used to evaluate the method with respective results. Finally, in Section 7 we conclude and discuss potential future work.

2 Related Work

Perhaps the most closely related work in the literature on affordance learning to our proposed work is by Fitzpatrick *et al.* in [9, 8, 16]. The authors trained the Cog robot to recognize "rolling" affordances of four household objects (a plastic bottle, a toy car, a toy cube and a toy ball) using a fixed set of actions to poke the objects in different directions as well as simple visual descriptors for object recognition. During training, the robot would localise objects viewed by its camera system, gather shape features, poke the object from a certain direction using one of four pre-defined actions, and track the object for a dozen frames afterwards. Over many trials, histograms were created for each of the actions and each of the objects to determine both the likely direction of movement when performing a given action on an arbitary object, and the likely direction of movement when performing an arbitrary action on a given object. There are two main differences between our method and that of Fitzpatrick *et al.* Firstly, in their system, 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 result features and it must determine for itself which ones are important for identifying the different affordances that it observes. Secondly, their system used object recognition to identify the affordances of individual objects, whereas our system determines the affordance class of objects (grounded in result features) based, not on their individual identity, but on their *class* with respect to a broad set of object property (e.g. shape) features.

Stoytchev examined an interesting extension of the general affordances idea in [20, 21] by enabling a robotic arm ferent tools afford the system when used to interact with an object. He distinguished between two sub-classes of tool affordances: *binding affordances*, that describe how tools afford being grasped by the arm, and *output affordances* that describe the effect that the various tools afford during interaction with the environment. Like Fitzpatrick *et al.* in [20], he used a small set of pre-defined exploratory actions and, by allowing the arm to explore these actions with different T-hook tools acting on a hockey puck, enabled the robot to build a tool output affordance table. The T-hook tools were colour-coded to facilitate recognition and affordances were learned with respect to individual tools rather than classes of tools. It is also worth noting that the features used to learn these tool output affordances were pre-selected and explicitly associated with these particular affordances based on prior knowledge of the experimenters.

Cos-Aguilera *et al.* [6] used a *self-organizing map (SOM)* [11] with a Hebbian learning mechanism called a *Growing When Required (GWR)* network to aid a simulated Khepera 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 GWR network was used to cluster sensor data in the input space. When the robot encountered a particular situation or object, the closest matching node in the network was found and weights were assigned to the node for each action the robot attempted to use in that situation based on the success of the action. Their prior work in [5] used a more simplified model to estimate the likelihood that a particular object, perceived as a set of features, affords the robot a particular action. While we also make use of SOMs in our learning algorithm, we use them in a different way to Cos-Aguilera *et al.*. Rather than using individual SOM nodes as exemplars, a technique that was employed in one of our previous works [17], we use SOMs to vector quantize feature space and cluster over the SOM nodes to form affordance classes. For more information on this methodology, refer to Section 5.

Saxena *et al.* [18, 19] provided an interesting blend of ideas from function-based object recognition and affordance learning when constructing a robotic system for grasping novel objects. Their robot used the same robotic arm used in our work (see Section 3.1) that had a webcam mounted on the end-effector. In [18] they trained a probabalistic model on features extracted from ray-traced images of synthetically generated household objects with labeled grasp points. After training, the system was tested on attempts to grasp real objects that it had not encountered in the training set by finding a potential grasping point and choosing between one of two possible types of grasping actions. What differs in our work is that rather than training our learning algorithm on synthetically generated object examples, we train our system through interactions with real objects. Moreover, in the work described by Saxena *et al.*, though the term "affordance" is not explicitly mentioned, the two possible grasping outcomes are specified in advance: graspable or non-graspable. Our system generates its own affordance classes by observing signifcant differences in behaviour through interactions with different objects.

3.1 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 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 be used for this work is a modified version of Golem [1], control software for the Katana arm developed by researchers at the University of Birmingham. 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, the software was modified to optimize for a linear end-effector motion trajectory when moving between workspace positions. We have developed the system to be controlled from the Matlab software environment using a CORBA interface and a Java control client which can easily be called from Matlab. This allows for swift arm/workspace calibration from within Matlab and provides simple *moveTo*(x, y, z) functionality for moving the end-effector to a localized position (x, y, z) in the workspace.

3.2 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 Section 6. The camera system is operated using a similar interface to that of the robotic arm. A Java client is called from Matlab to interface with a CORBA server that implements the low-level camera functionality.

4 Visual Feature Extraction

4.1 Object Property Features

With regard to the object property 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.

4.1.1 Range Data We have developed a method for segmenting the object from range images that uses RANSAC (RANdom SAmple Consensus) [7] to fit a plane to the table surface for removal, then mean-shift clustering [3] 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 [15], which uses the min-cut/max-flow algorithms oulined in [2,12,1] to apply the standard graph cut technique to segmenting multimodal tensor valued images. See Fig. 3 for sample range and intensity image segmentations for 2 different objects.

[1] http://www.cs.bham.ac.uk/~msk/

Figure 2: Experimental Setup.

A quadratic surface may then be fitted to the object range data to derive curvature features from the object surface. To achieve this, we fitted the following quadratic polynomial surface function to the segmented object range data:

$$Z = \frac{1}{2}aX^2 + bXY + \frac{1}{2}cY^2 + dX + eY + f$$

using least squares to solve for the coefficients a, b, c, d, e and f. Taking eigenvalues of the matrix $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$ then provided two features which form a good description of the global curvature of the object. This surface fitting technique is also illustrated on the two objects shown in Fig. 3.

4.1.2 Intensity Images The segmentation technique described in the previous sub-section 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.

4.2 Result Features

After an arm action has been performed on an object, the resulting videos of the interaction are processed for result features. This is primarily acheived by tracking the object in motion using a probabilistic tracker from [14]. 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 [13], which allows for tracking with a smaller number of particles, and

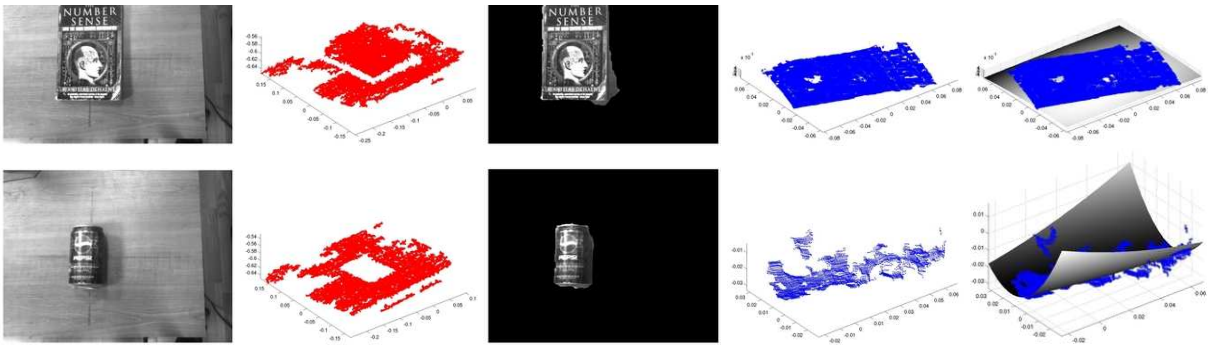

Figure 3: Examples of image and range data taken with the stereo camera for two different types of objects: a book which slides when pushed by the robotic arm, and a Pepsi can which rolls when pushed by the arm. From left to right: intensity image, range data of the scene, segmented object, segmented object range data, object range data with a fitted quadric surface.

consequently, near real-time tracking performance.

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

4.2.2 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. This required developing an extension to the particle filter tracker previously described. The tracker by itself is quite 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. While this does not affect global motion tracking too significantly, it is an unacceptable starting point if we wish to use the output of the tracker to segment an object from frame to frame before calculating histogram differences. If the tracker overshoots the object, this causes subsequent segmentation to be inaccurate, producing major inaccuracies in the histogram difference calculations. To compensate for 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 [22] to localise the object within the window. An initial segmentation of the object from the start frame of the video is used to get a colour histogram model which forms the basis for the histogram back-projection object localisation. See Fig. 4 for sample frames from an interaction with an object that illustrates this technique at work.

Once the object has been localised over the entire motion sequence, colour and edge histograms are calculated over its local area for every frame and histogram differences are calculated between frames using the Bhattacharyya coefficient. Histogram difference averages are then calculated from the start of object motion until the end. We determine the start frame of object motion in a similar way to Fitzpatrick *et al.* [8] by looking for a significant expansion of the image regions segmented by background subtraction, e.g. when the arm moves towards the object. We derive 3 result features from this procedure: average colour histogram difference, average edge histogram difference, and the multiplication of these two values. Although these features are sensitive to object rotation, this may not necessarily be a bad thing since objects that rotate behave differently to those that do not by definition, and this may equate to a significant difference in terms of their respective affordances.

5 Learning Method

Our learning method is outlined visually in Fig. 5 and in algorithmic form in Algorithm 1. It involves 2 main stages: unsupervised clustering in the result space and training a supervised classifier in object property space using the clusters as labels. These are described later in Sections 5.1 and 5.2. In Stage 1, we chose to use a *self-organizing map (SOM)* [11] to vector quantize the result space. In [17], we provided a more detailed review of the SOM training algorithm, but to briefly summarise, a SOM is effectively a finite set of nodes that are connected to each other via a map topology and a neighbourhood relation on a low-dimensional (usually 2D) grid. These nodes contain weight vectors that share the same dimensionality as the data that the SOM is trained on. Training occurs by way of finding *best-matching unit* (BMU) nodes in the SOM for input data vectors and updating both the BMU and its neighbours using an update rule.

The overall result of this type of competitive learning amounts to a form of *vector quantization* that preserves the topological structure of the input data. We use SOMs in our algorithm as a first step towards making the algorithm capable of on-line learning. The SOMs may be trained incrementally and always maintain a fixed, pre-selected number of nodes. The neighbourhood function and learning rate are also easily adjustable. This means that if we have a small amount of data, the learning rate can be increased so that data clusters are approximated quickly, or if we have a large amount of data, the learning rate can be decreased to accomodate the additional information over a longer learning period, all while maintaining a constant dataspace size. The type of two-staged clustering of data described in the previ-

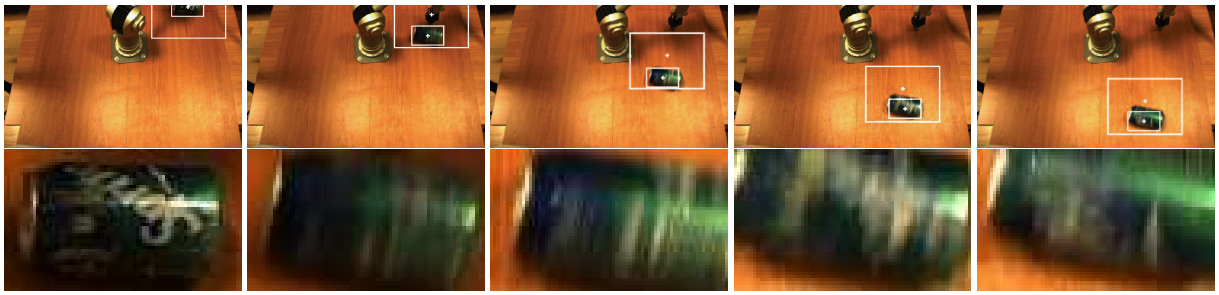

Figure 4: An example of the object tracking mechanism described in Section 4.2. The images in the first row show a progression of frames tracking a Sprite can being pushed by the arm. The outer rectangle is a likelihood window around the object obtained using the particle filter tracker. The inner rectangle 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.

ous paragraph, where a SOM is trained on the data before it itself is clustered using k-means or another unsupervised clustering algorithm, has previously been shown to perform well when compared to direct clustering of the data and also reduces the computation time [23]. This would obviously be a desirable trait to incorporate in an on-line learning system.

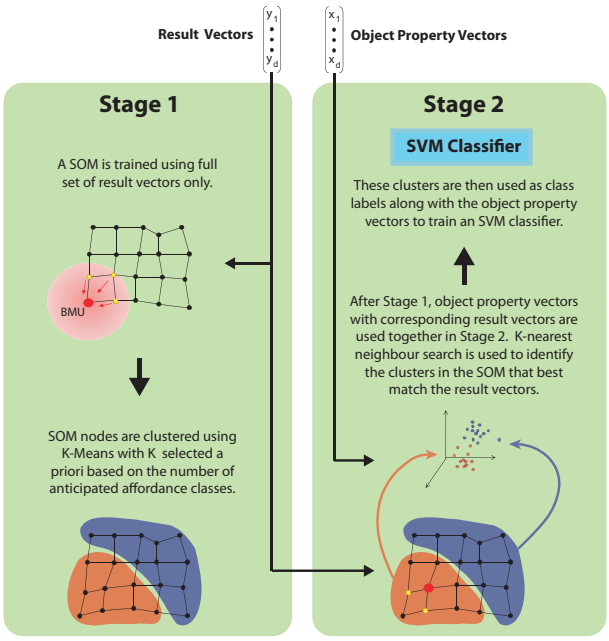

Figure 5: Our learning algorithm. See Section 5 for further details.

Figure 6: Classification of object property vector inputs.

5.1 Stage 1: Unsupervised Clustering in Result Space

In Stage 1 of our learning procedure, given a training set of object property feature vectors and corresponding result feature vectors, a SOM is trained using the full set of result vectors only. The nodes of the SOM are then clustered using k-Means clustering, similarly to a method reported in [23], using a value of k that is selected a priori based on the anticipated number of affordance classes present in the dataset. These SOM clusters are then used as class labels to train a supervised classifier in Stage 2 of the learning procedure.

5.2 Stage 2: Training a Supervised Classifier in Object Property Space

Here, the entire training set of both object property vectors and corresponding result vectors is then taken, and the best-matching SOM cluster is found for each result vector using a *k-nearest neighbours (KNN)* search. The best-matching clusters for each result vector are then used along with the corresponding object property vectors to train a *support vector machine (SVM)* classifier [4] with a *radial basis function (RBF)* kernel in the object property space. After training, test samples may be classified in terms of affordance classes (i.e. the SOM clusters) grounded in the result space using their object property feature vectors, as shown in Fig. 6.

6 Experimental Results

6.1 Description of Experiment

To test our affordance learning system, the experimental environment was set up as previously described and as shown in Fig. 2. The Bumblebee stereo camera was positioned on a tripod on the right side of the workspace above the table surface, while the monocular Flea camera was positioned in front of the workspace, also on a tripod, giving both cameras a top-down viewpoint of the scene.

A pushing action was provided to the system which involved keeping the forearm part of the arm orthogonal to the work surface and pushing from the top part of the workspace to the centre, through a fixed object start position as shown in Fig. 4. We selected 8 household objects to be used in the experiments as shown in Fig. 7: four non-rolling objects- a book, a CD box, a box of tea and a drink carton, and four

Input: Training dataset of matching object property vectors $\mathcal{OP}_i = \{x_1, \ldots, x_n\}$, result vectors $\mathcal{R}_i = \{y_1, \ldots, y_d\}$ and constant k.
Output: Affordance classifier C.

1: Normalise $\{\mathcal{OP}_i\}_{i=1}^{N_{\text{Data}}}$ and $\{\mathcal{R}_i\}_{i=1}^{N_{\text{Data}}}$.
2: Create SOM with N_{SOM} nodes $\{\mathcal{W}_i\}_{i=1}^{N_{\text{SOM}}}$ where $\{\mathcal{W}\}_i = \{w_1, \ldots, w_d\}$
3: Initialise the SOM by randomizing the weight vectors and setting an initial learning rate α.
4: **for** each $\mathcal{R}_i, i = 1 \ldots N_{\text{Data}}$ **do**
5: Find SOM BMU, $\mathcal{W}_{\mathcal{R}_i}$, for $\mathcal{R}_i$ using $||\mathcal{R}_i - \mathcal{W}_{\mathcal{R}_i}|| = \min_j\{||\mathcal{R}_j - \mathcal{W}_j||\}$
6: and update the $\mathcal{W}_j$ SOM nodes using the following update rule: $\mathcal{W}_j(t+1) = \mathcal{W}_j(t) + \alpha(t)H_{\mathcal{W}_{\mathcal{R}_i}(t)}[\mathcal{R}_i(t) - \mathcal{W}_j(t)]$ where t denotes time, $H_{\mathcal{W}_{\mathcal{R}_i}(t)}$ is the neighbourhood kernel around the BMU $\mathcal{W}_{\mathcal{R}_i}$ and $\alpha(t)$ is the learning rate at time t.
7: **end for**
8: k-means cluster the SOM nodes to get clusters $\{K\}_{i=1}^{k}$.
9: **for** each $\mathcal{OP}_i, \mathcal{R}_i, i = 1 \ldots N_{\text{Data}}$ **do**
10: Use KNN search to find $K_{\mathcal{R}_i}$, the best-matching SOM cluster for $\mathcal{R}_i$.
11: **end for**
12: Train SVN classifier C in object property space using an RBF kernel with $\{\mathcal{OP}_i, K_{\mathcal{R}_i}\}_{i=1_{\text{Data}}^{N}}$ as the dataset, where the $K_{\mathcal{R}_i}$ are treated as labels.

rolling objects- a box of cleaning wipes, a Pepsi can, a Sprite can and a tennis ball box. Fig. 7 contains examples of interactions with these objects and illustrates the distinct difference in behaviour between between them. During trials, each of these objects was placed centred at the start position with a consistent orientation, as in Fig. 3, and the Katana arm pushed the object at a fixed speed using the pushing action described earlier.

Before an action was performed on an object, both intensity and range images were gathered from the Bumblebee stereo camera. This data was then processed to produce the 12 object property features discussed in Section 4.1. After an action was performed on an object, images were gathered from the Flea camera and passed to the tracking system, as well as the other feature extractors described in Section 4 to produce 12 result features. The extracted features were then used to train the affordance classifier. We used a 100-node 10x10 hexagonal lattice SOM with a sheet-shaped topology and an initial α learning rate of 0.7.

In order to evaluate the affordance learning 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. In the following section's evaluations, leave-one-out cross validation was performed by spliting 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. Cross validation was performed by using each of the 8 objects in turn as the test object and averaging sets and the 20 test samples contained therein.

6.2 Results

There were two major questions we were interested in. Firstly, when performing actions on objects and observing how they behave as a result, is the system capable of recognising the results of its own actions? Secondly, when confronted with an object it has not yet interacted with, is the system capable of predicting what will happen when it does interact with it? This second question can be broken down further: does the system predict the same outcomes as a human would predict when presented with the same scenario? We devised 3 evaluations to answer these questions, detailed in the following sub-sections. Results are shown in Tab. 1.

In the following, the term *test sample ground truth* will refer to the affordance ground truth label attached to the data samples during data collection. These labels were manually added to the dataset by the authors. The ground truth estimates for the SOM affordance clusters in the classifier were estimated by counting the number of ground truth labels that gathered there during training.

6.2.1 Does the sytem recognise the results of its actions?

To answer this question, we assumed that the system had already observed the results of its actions contained in the test dataset. After training the classifier using 7 training objects, the best-matching SOM clusters for the test sample result vectors were found using KNN over the SOM nodes, with k set to 3. If the test sample ground truth corresponded to the estimated cluster ground truth for the winning cluster, this was deemed to be a true match. After performing leave-one-out cross validation, the average score for this test was **100%** true matches, as shown in Tab. 1.

6.2.2 Can the system predict the results of its actions?

Here we assumed that the system had not yet observed the results of its actions and trained the classifier using the object property vectors in the training set. The test sample object property vectors were then classified using the classifier as in Fig. 6 and the cluster label outputs were used to identify the relevant clusters in the SOM. If the cluster selected by the classifier was the same as the cluster selected by KNN-matching the result vector (again with k set to 3), this was deemed to be a true match. Averaging over the cross-validation scores, the result was **91.56%**.

6.2.3 Does the system predict the same results as a human?

This final test once again assumed that the system had not yet observed the results of its actions, and again involved classifying the test sample object property vectors, but this time, the estimated cluster ground truth was checked against the test sample ground truth. If they matched, this indicated that the system correctly predicted what the human predicted in the same scenario. Averaging over the cross-validation scores, the result was also **91.56%**.

7 Conclusion & Future Work

To conclude, we have provided a discussion on the problem of affordance learning, as well as a review of related works

Recognition	N/A	**100%**
Prediction	**91.56%**	**91.56%**

Table 1: Experimental results: recognition of the observed effect (based on the extracted result features), prediction of the effect (based on the extracted object property features); with respect to estimated affordance classes (obtained by unsupervised clustering) or to manually labelled affordance ground truth classes.

in the area and how our work contributes. We have outlined the major technical aspects of our system and we have proposed a novel learning algorithm that can learn basic affordances of objects in a real-world environment by interacting with them using a robotic arm. Finally, we have proven the efficacy of this method with an experimental evaluation.

Interesting topics for future work include possible techniques for automatically estimating k (the number of affordance classes), on-line learning, training the system on more than one action, and trying a different affordance learning scenario, e.g., grasping objects.

Acknowledgement

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).

References

[1] 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*, pages 1124–1137, 2004.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1222–1239, 2001.

[3] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619, 2002.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[5] I. Cos-Aguilera, L. Canamero, and G. Hayes. Motivation-driven learning of object affordances: First experiments using a simulated khepera robot. *The Logic of Cognitive Systems. Proceedings of the Fifth International Conference on Cognitive Modelling*, 1001:57–62, 2003.

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

[7] 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.

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

[9] J.J. Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1986.

[10] T. Kohonen. Self-organizing maps. *Springer Series In Information Sciences; Vol. 30*, page 426, 1997.

[11] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 147–159, 2004.

[12] M. Kristan, J. Perš, A. Leonardis, and S. Kovačič. A hierarchical dynamic model for tracking in sports. In *Proceedings of the sixteen Electrotechnical and Computer Science Conference, ERK07*, September 2007.

[13] M. Kristan, J. Perš, M. Perše, and S. Kovačič. Towards fast and efficient methods for tracking players in sports. In J. Perš and D. R. Magee, editors, *Proceedings of the ECCV Workshop on Computer Vision Based Analysis in Sport Environments*, pages 14–25, May 2006.

[14] 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.

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

[16] B. Ridge, D. Skočaj, and A. Leonardis. A system for learning basic object affordances using a self-organizing map. In *Proceedings of First International Conference on Cognitive Systems (CogSys))*, 2008.

[17] 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 (NIPS) Conference*, Vancouver, Canada, 2006.

[18] 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.

[19] A. Stoytchev. Behavior-grounded representation of tool affordances. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3060–3065, 2005.

[20] A. Stoytchev. Toward learning the binding affordances of objects: A behavior-grounded approach. *Proceedings of AAAI Symposium on Developmental Robotics*, pages 21–23, 2005.

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

[22] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *Neural Networks, IEEE Transactions on*, 11(3):586–600, 2000.

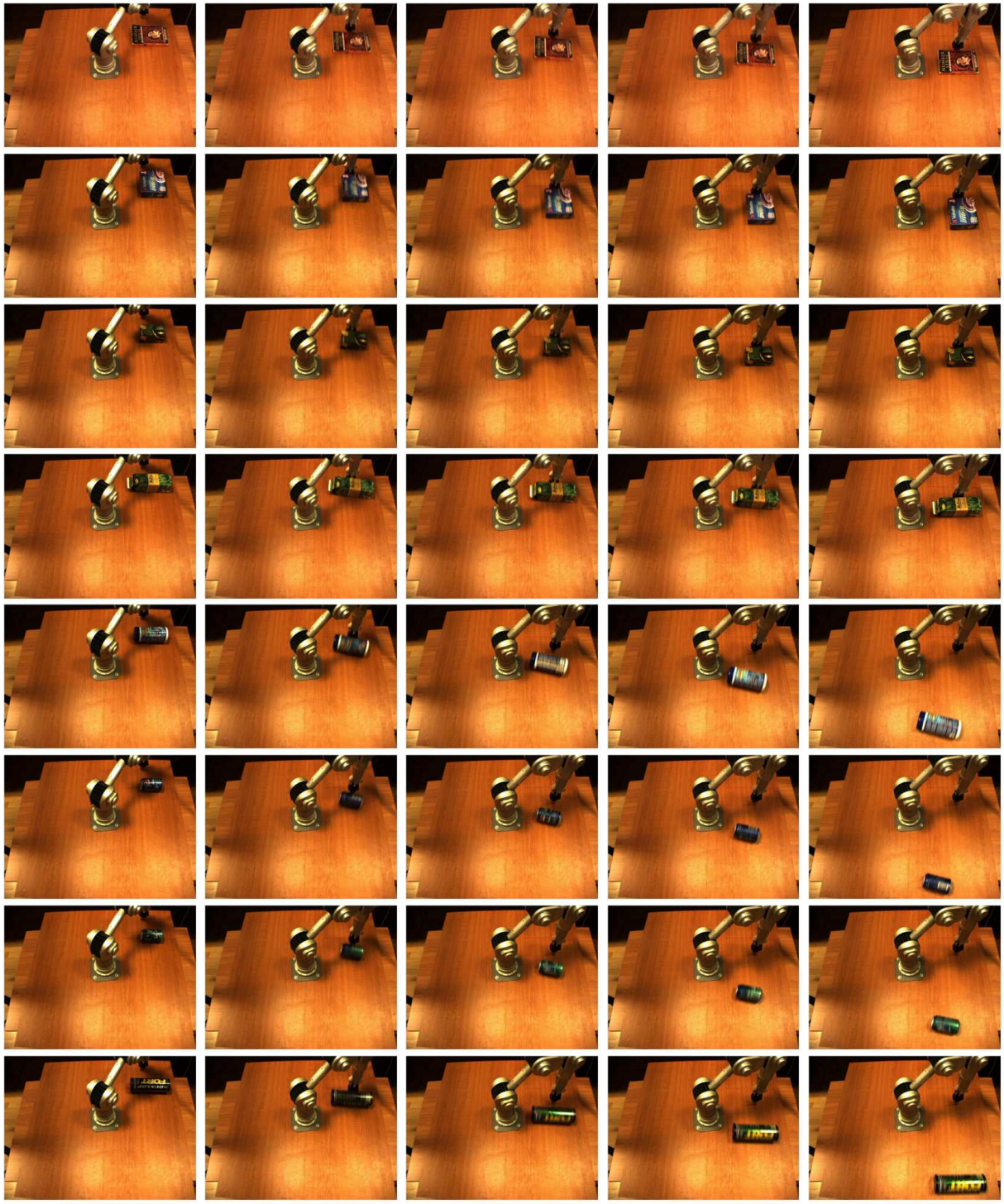

Figure 7: Sample interactions as seen from the Flea camera of the test objects from Section 6 being interacted with. From top row to bottom: a book, a CD box, a box of tea, a drink carton, a box of cleaning wipes, a Pepsi can, a Sprite can, and a tennis ball box. The first four objects tend to slide, while the last four tend to roll.