Improving Robot Vision Models for Object Detection Through Interaction

Jürgen Leitner, Alexander Förster, Jürgen Schmidhuber

Abstract— **We propose a method for learning specific object representations that can be applied (and reused) in visual detection and identification tasks. A machine learning technique called Cartesian Genetic Programming (CGP) is used to create these models based on a series of images. Our research investigates how manipulation actions might allow for the development of better visual models and therefore better robot vision.**

This paper describes how visual object representations can be learned and improved by performing object manipulation actions, such as, poke, push and pick-up with a humanoid robot. The improvement can be measured and allows for the robot to select and perform the 'right' action, i.e. the action with the best possible improvement of the detector.

I. Introduction

COMPUTER VISION has become a more and more prominent topic of research over the past decades, also in the field of robotics. Like humans and animals, robots are able to interact with the world around it. While most robot vision research tends focus on understanding the world from just passive observations, these interactions with the environment provide and create valuable information to build better visual systems. Connecting manipulation commands with visual inputs allows for a robot to create methods to actively explore its surroundings. These connections between motor actions and observations exist in the human brain and are an important aspect of human development [1].

An important goal in robotics is to extend the application of robotic systems from automation tasks in rather fixed scenarios, to other more complex, domestic contexts. There a closer coordination between sensing and acting will be crucial to operate autonomously and adapt. To do so a robust perception and clear understanding of the surroundings is critical. From a robot vision point of view, this means that the robot is required to detect previously unknown objects in its environment and be able to build models to identify them in the future.

In infants various specialisations in the visual pathways may develop for extracting and encoding information relevant for visual cognition, as well as, information about the location and graspability of objects [2]. In this paper we propose a method for building such specific encodings for objects and object detection for use with a humanoid robot. We show how such models for detection can be generated using a machine learning technique called Cartesian Genetic Programming (CGP). To build more accurate and robust representations, allowing the detection and identification in a wide range of settings, interaction is of critical value. We describe how the models can be improved by performing an action that allows to gain more information about the specific object. We also show that we can measure the improvement and therefore are able to select the 'right' action, i.e. the action providing the best possible improvement of the detector.

The authors are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), Manno-Lugano, Switzerland, a research institute affiliated both with the Universitá della Svizzera italiana (USI) and the Scuola universitaria professionale della Svizzera italiana (SUPSI) in Lugano, Switzerland. The authors acknowledge the financial support by the EU under FP7 grant #270247 'NeuralDynamics'. (corresponding author email: juxi@idsia.ch).

II. Background and Related Work

Robotic vision has mainly been focussing on the understanding of the passive camera images received, neglecting the fact that robots are able to interact with these objects.

One of the main approaches to perform object detection in computer vision is to segment the images, i.e. separate the object of interest from the background and the rest of the scene. There is a vast body of work on all aspects of image processing and segmenting, using both classical and machine learning approaches [3]. However, image segmentation is especially challenging when the objects are static, occluded or the model is uncertain. A variety of feature detectors have been proposed to detect objects in the scene [4], [5]. These features are then used in various ways to solve the object classification problem, e.g. by using simple machine learning techniques such as simple clustering (e.g. k-means) or even more complex approaches, like, Artificial Neural Networks (ANN) [6].

Furthermore researchers started addressing how in robotic settings a more autonomous fashion to object detection can be devised [7], as well as, how the need for a human teacher can be minimised [8]. Yet all these approaches neglect the

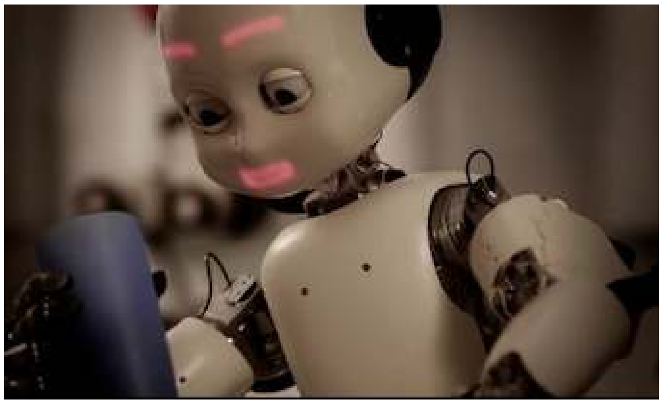


Fig. 1. The *iCub* humanoid robot manipulating a cup to build a better model for detecting the objects from vision.

possibility of the robot taking actions to improve its ability to detect objects – i.e. the scene the robot is looking at is considered static (at least during the learning phase).

In the primate brain visual stimuli and motor commands are closely intertwined [9], and it has been suggested that this enables more adaptive, more autonomous behaviours. For robots to act in such a fashion a higher level of integration and coordination between perception and control is necessary. In the last few years robot vision research has been extended to investigate how the embodiment of the robotic system can be used to create better visual perception skills.

How to do motion planning for a high DOF is another issue that needs to be tackled to perform useful actions. There has been a extensive research on this in the past (see e.g. [10], [11], [12]. A commonly used approach is based on Rapidly Exploring Random Trees (RRT) [13]. For example, Vahrenkamp et al. [14], used a variant of RRT for both planning the reach and the grasp on a high DOF humanoid robot. Few have been investigated on how to plan for a good action to allow better visual recognition.

One of the first to investigate robot actions for robot vision were Metta and Fitzpatrick [15]. They showed that performing actions can lead to a simple object segmentation based on using optical flow information. Their work extracts visual information through autonomous exploration of the environment. When the robot hits an object placed on a plane in front of it, a binary segmentation is performed based on the object's motion. The authors highlight that following the causal chain from the robots action allows to develop visual competence. While the theoretical impact is large it does not go into the details of how to generate suitable object representations for online object search and recognition.

With the rise of cheaper stereo vision systems and depth sensors most of the object recognition seems to focus on 3D geometry, or shape-based, detection. Welke et al. [16] showed an implementation that allows one to build visual detection based on a depth information and a 3D view sphere concept. Using a static camera setup they are able to generate a motion of the arm to rotate and cover as much of a 3D sphere as possible, allowing to build a model of the object from these multiple views. A similar approach by Gonzalez-Aguirre et al. [17] tried to overcome some of the issues when trying to detect general objects based on their 3D shapes purely from vision. By fusing multiple views and vantage points a more robust detection was generated. In contrast our work does not require or even try to build 3D models of the objects.

A complex, humanoid robot, with its extra DOF, allows for more and different types of 'interaction' with the environment, also with respect to robot vision applications. Yet little work has investigated these extended possibility, such as in our `LEAN` action (see Section III). Stasse et al. [18] suggest that using the hip can extend the view area and therefore the possibility for detecting known objects. Their work, based again on a next-best geometric view point approach, can derive motions of the hip to change view points. It is not clear though how their technique can handle redundant DOFs.

Object Segmentation, by itself, is a very challenging problem in computer vision. It has been extensively investigated over the last decades. Also the use of a robot to perform actions to help with the segmentation, similar to the approach mentioned above, has been used. Schiebener et al. [19] used a push action to detect and segment an object based on set of geometrical shapes and a hypothesis on how they push will change the view thereof. Building on these works we investigate multiple actions on how they provide a better classification and segmentation of objects without the use of any prior shape or 3D object knowledge.

III. Robot, Objects and Actions

Our research platform is the *iCub* humanoid [20], providing a 41 degree-of-freedom (DOF) open-system robotic platform. It comprises of an upper-body, two arms, a head and a torso (see Fig. 1). The *iCub* is currently used in more than 20 research labs worldwide and is considered an interesting experimental platform for the study of embodied Artificial Intelligence and cognitive, as well as, sensorimotor development [21].

We focus on developing models for the detection of four distinct objects from vision (Fig. 2) only, without the use of any 3D or shape models or other priors. Each model shall uniquely detect one object, also solving the object identification problem at the same time. The aim is to create models that can be run, in parallel, on the real robot and allow for robust detection and identification of these objects in changing environments even when other objects are present. We propose a machine learning method to build those models in Section IV.

To learn robust models of the objects our humanoid robot needs to perceive the objects in various poses and from various angles. For example, the tea boxes (in Fig. 2), differ visually between the front and back side. To learn better models for the detection of these objects, a pre-selected action set is available to the humanoid robot. These actions allow to see the object from various angles.
The available, scripted actions are the following:

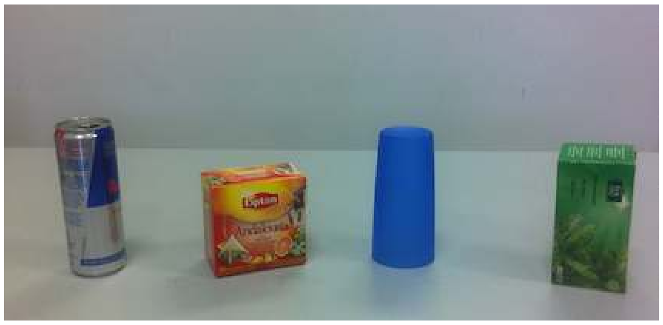


Fig. 2. The objects to be detected and distinguished in this scenario are: a soda can, a yellow-red tea box, a blue plastic cup and a green tea box.

- **Action** LEAN**:** this simple action using 1 DOF, changes the position of the torso, by performing a hip motion. The robot will lean about 30° to its left and then 30° to the right, while keeping its gaze fixed at the position of the object on the table (Fig. 3).
- **Action** POKE**:** the extended index finger of the end-effector is used to poke the object from the right side. A (more or less) linear movement of about 10cm in operational space is performed, while the robot gazes at a fixed location (Fig. 4).
- **Action** PUSH**:** the palm of the right hand is used by the robot to push the object from right to left. This is again a linear movement, but with higher velocity and longer path. The gaze is controlled to continuously look at the end-effector (Fig. 5).
- **Action** CURIOUS**:** is the most complex action available and is modelled after a child curiously exploring an object. The object is grasped, picked-up (from a fixed location using a predefined grasp primitive) and then brought closer to the face while being rotated in various ways. These rotations involve almost all DOF of the wrist, elbow and shoulder. The gaze during this whole motion is continuously adjusted to look at the hand and the object it is holding (Fig. 6).

Camera pictures are recorded in a fixed interval of $2s$, while the robot is performing these actions. These are then used to improve the visual models. To allow for a fair comparison, as the actions have different runtimes, only 5 images are taken from each action for the experiment.

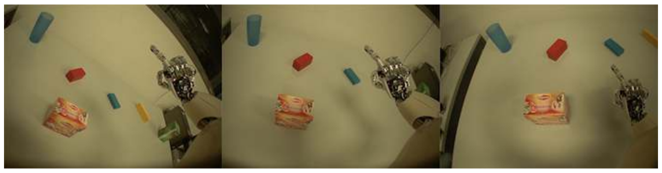


Fig. 3. An example of a series of images collected during a LEAN action with the *iCub* robot.

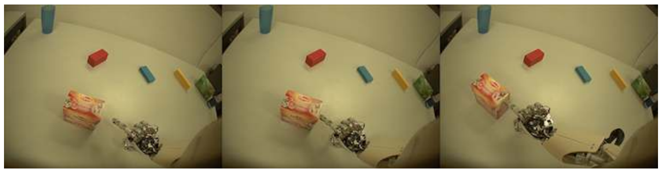


Fig. 4. An example of a series of images collected during a POKE action with the *iCub* robot.

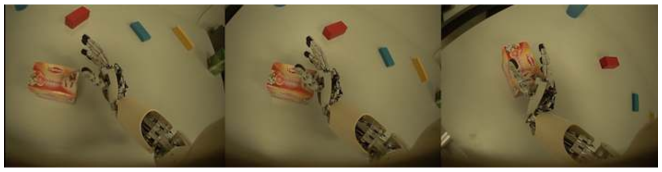


Fig. 5. An example of a series of images collected during a PUSH action with the *iCub* robot.

IV. Learning a Model for Visual Object Detection & Identification

We are interested in building object representations that allow for fast and robust detection in visual inputs. For the object detection (and identification) a novel approach using a model based on feed-forward graphs is proposed. The partially-connected graph consists of nodes, which perform specific image processing operations. The output, when executing the model on an input image, should be a binary-segmentation separating the object of interest from the rest of the scene. While such a graph can be derived and designed by hand, the real strength is the ability to learn these visual object representations autonomously. It allows to create unique models for specific objects, whenever a new one is encountered.

Herein we are using Cartesian Genetic Programming (CGP) [22], [23] method to find a model performing the visual detection and identification. CGP is a variation of Genetic Programming (GP), a search technique inspired by concepts from Darwinian evolution [24]. It can be used to generate formulae or programs to solve specific problems. It has been applied successfully in many areas [25][26].

The basic algorithm works as follows (see also Fig. 7 for reference): At first, a population of individuals is randomly generated (Fig. 7*a*). Each of these individuals is a candidate solution with its object detection program being represented by its genome. In the next step all these individuals are tested. This step generates a fitness value by testing how well each one of them solves the given (detection) problem. The individuals in the population are then ranked according to this value (Fig. 7*b*). New individuals are created from the best ranked, therefore the best performing individual, using functions analogous to recombination and mutation (Fig. 7*c*). (In the case of CGP it previously was shown that mutation works well enough without the need of recombination) The best individual with its offsprings turn into the next population (Fig. 7*d*). The process of testing and generating of new populations is repeated until a sufficient solution is found or a maximum of individuals have been evaluated.

Our implementation, named CGP for Image Processing (CGP-IP) [27], is inspired by previous work using CGP in the field of computer vision. In CGP-IP the model is

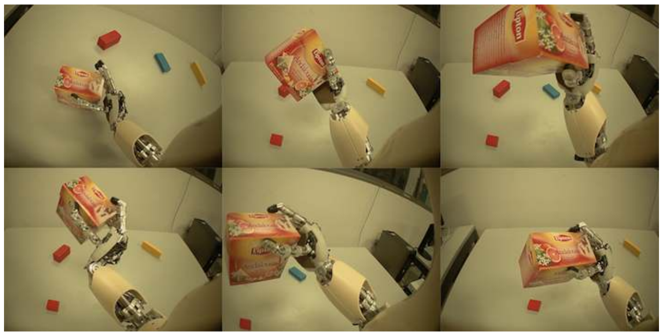


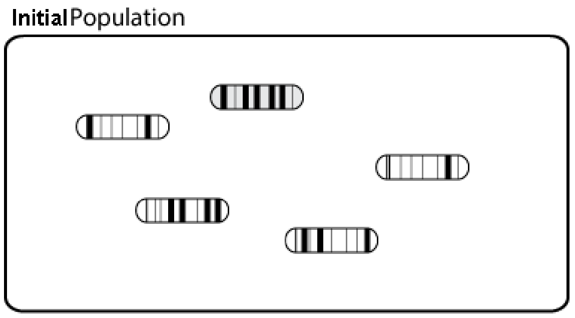
Fig. 6. An example of a series of images collected during a CURIOUS action, which enables the *iCub* to closer inspect the object.

encoded in the genotype, where each gene represents a node in the graph. The contents of the gene represent the node's functionality and describe which nodes the incoming edges are connected to (Fig. 8). CGP was chosen as it offers some nice functionalities, for instance, not all of the nodes of a solution representation (the genotype) need to be connected to the output node of the program, a feature known as 'neutrality'. This has been shown to be very useful in the evolutionary process [28]. Another feature is the graph encoding of the genotype allowing reuse of nodes making the representation distinct from classic GP.

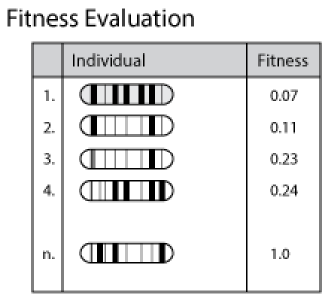
An important difference to previous GP approaches is the ability to include domain knowledge, i.e. each node can contain a high-level operation by itself (e.g. a Gabor filter) not just a simple mathematical operator. In fact each node in CGP-IP is performing a certain functionality from the OpenCV image processing library [29]. Around 60 unique functions are available to CGP-IP to automatically generate computer programs to perform the detection. A complete list can be found in Harding et al. [27]. While this function set is larger than the typical setup for CGP it does not seem to hinder evolution, on the contrary, the higher number of functions seems to provide greater flexibility. The efficacy of our CGP-IP implementation has been shown for several different domains by Harding et al. [27] and Leitner et al. [30], [31].

To execute a filter evolved with CGP-IP a genotype-phenotype mapping has been performed first. This is pretty straightforward, by starting at the output node and following the links to its inputs the active nodes are identified. Then a forward pass of the phenotype is done generating the output image at each node. To do so the inputs are collected and the function encoded in the genotype is applied to generate the node's output. The genome also contains the specified parameters required by the function. Our implementation of CGP-IP generates human-readable code based on OpenCV functions, during the forward pass (e.g. Listing 10). Due to the high quality, high speed implementation of OpenCV CGP-IP individuals are evaluated at the rate of hundreds per second on a single core CPU. This makes it both efficient to evolve solutions and run the final programs. Our existing computer vision framework enables this code to be used directly with our robots [32].

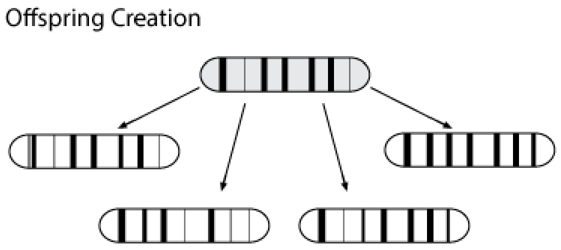
To establish the fitness of each individual in the population the Matthews Correlation Coefficient (MCC) [33] between the output from the run of the model, i.e., a binary image, and a target segmentation ('mask'), is computed. It was previously observed that MCC is a useful metric for CGP



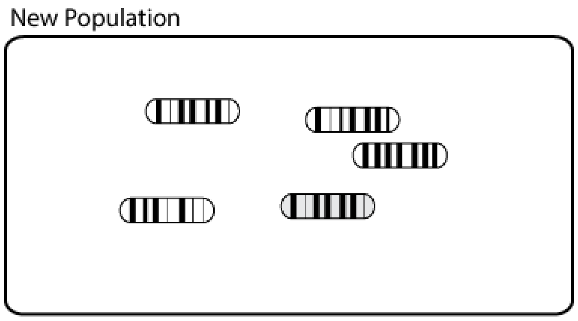
(a) The first population with its randomly generated individuals. The gene values are represented by the vertical line thickness.



(b) The ranked individuals with their respective fitness values.



(c) Offsprings are created from the best individual by mutating each gene with a given probability.



(d) The offsprings and the best individual from the previous round created the new population.

Fig. 7. Illustration of a typical iteration for an evolutionary search using (Cartesian) Genetic Programming.

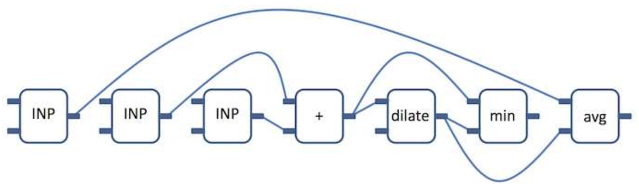


Fig. 8. Example illustration of a CGP-IP genotype. In this example, the first three nodes obtain the first three input image channels, the grey scale, the red and the green channels, respectively. The fourth node adds two inputs together. The output is then dilated by the fifth node. The sixth node (`min`) is not relevant for the output (i.e. it is neutral) and is ignored. The last node averages the fifth node and the grey scale input to generate the output.

TABLE I
PARAMETERS OF CGP-IP ENCODED AT EACH NODE.

Parameter	Type	Range
Function	Int	# of functions
Connection 0	Int	# of nodes and inputs
Connection 1	Int	# of nodes and inputs
Parameter 0	Real	no limitation
Parameter 1	Int	$[-16, +16]$
Parameter 2	Int	$[-16, +16]$

approaches to classification problems [34]. The coefficient is based on the true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) at a pixel-level. It is computed as follows:

$$c = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

As we are doing binary segmentation, coefficients of 1 (a perfect classifier) and -1 (a perfect classifier just switching the two classes) are both valid solutions. The fitness f of an individual is hence given by

$$f = 1 - |c| \qquad (1)$$

with values closer to 0 representing more fit individuals, i.e. representations performing a better classification.

CGP-IP provides single channel images to each node, expecting to output again a single channel image. It splits the incoming colour images from the robot into both RGB and HSV channels and provides these as inputs. Furthermore a grey-scale version of the image is provided. Evolution selects which inputs will be used for each detector. To select a specific channel of the input image special functions are provided. These functions return the current input channel (`INP`) or can be used to decide which input image channel is used (e.g. `SKIP`).

CGP implementations require a low number of parameters for configuration. The main parameters in CGP-IP are:

- number of nodes in the genotype (graph length): 50
- mutation rate (and size), fraction of all genes that are mutated when an offspring is generated: 10%
- number of separate populations, also known as islands, providing a distributed evolutionary process, which has been shown to improve the overall performance [35]: 8
- number of individuals per island: 5
- interval of synchronisation between islands: 10

Currently these values are set by casual experimentation. Improving the performance may be possible by parameter tweaking, especially the values for mutation rate, genotype size and number of populations. Like with every CGP implementation every gene encoding a node contains a set of parameters (Table I), in addition to these global parameters. *Connection* elements contain the relative address of the node used as inputs. *Parameter* values are required for OpenCV functions. Certain of these functions have specific requirements, as to the type and range of the parameters used. The genotype contains also an additional parameter used for thresholding the output.

V. EXPERIMENTS AND RESULTS

A. Learning a Simple Model Using CGP-IP

In this first experiment we use CGP-IP and supervised learning to generate a model for visual detection.

We define an area of interest in the camera image and place the object within this mask. This is used as training set to build the visual object representation. The mask does not precisely segment the image from the background, but highlights the area in the image the object is visible. The exact segmentation of the object is part of the first learning phase. Herein we use a fixed mask and manually place the object to be within. In the future this preliminary mask could very easily be generated from stereo vision information [36]. In this experiment the robot is static. Yet to not simply detect objects on simple visual cues, e.g. blue cup vs. red tea-box, we scatter a variety of objects (kids toy blocks) with different colours in the scene (visible in the background in Fig. 9). We use just one input image with a fixed mask for all objects as training set to start with. In this case we assume a static environment where the input images will not change over time.

Though only one training image is used, and the mask is not very accurate, our CGP-IP approach learns to segment and detect the object quite accurately. The simple mask used for all objects does not specify the detailed outline of the item, nevertheless the detectors manage to come close to a precise segmentation. A specific detector is trained for each object individually therefore allowing identification as well.

The learned detection model for the visually rather simple blue cup object (see Fig. 2) is used as a representative example here. The object representation, converted into executable code is shown in the listing in Fig. 10. The representation also allows for manual improvement by a skilled engineer. The solution was found after only 1214 individuals were evaluated, taking a few seconds on a standard desktop computer. The detection is shown in the middle of Fig. 9, where the binary segmentation is used as a red overlay in the input image. The execution of the code takes $140ms$ for a 320×240 pixel image.

The resulting fitness values from this first experiments are not particularly high. There are multiple reasons for this, first the mask or predefined segmentation is not very accurate, due to fact that it is the same mask for every object. Secondly the CGP approach is limited in its training time (to around 20k evaluations). This limit is chosen as to avoid over-fitting to the single input image available.

To show this we ran another experiment for the red tea-box for over 2.6m individuals. The found solution was very fit with $f = 0.06$. To achieve this the detector tried to artificially increase the found area to match the input mask as precisely as possible. This is in comparison to another solution evaluating only about 22.7k individuals to find a detector with $f = 0.27$ (see Table II for more details).

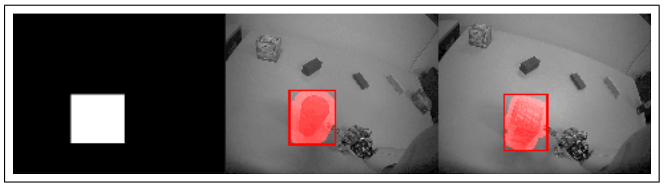


Fig. 9. The mask (left) used as input to the supervised training, next to two frames showing learned detection (as red overlay) for two distinct objects, the blue cup and the green tea box respectively.

```
icImage BlueCupFilter::RunFilter() {
    icImage *node43 = InputImages[4];
    icImage *node49 = node43->LocalAvg(15);
    icImage *out = node49->threshold(81.53244f);
    return out;
}
```

Fig. 10. The generated C++ code from the first learned object representation for the blue cup. This detector is rather simple. Although it detects the blue cup in all test images it has also a few false positives, due to its simplicity. `icImage` is a wrapper class for the OpenCV functionality and memory management within our framework [32].

TABLE II

COMPARING THE VARIOUS VISUAL DETECTION MODELS DERIVED FROM MANIPULATING THE OBJECTS USING A SET OF ACTIONS.

Detector	f [1]	**Ind**[2]	**Runtime**[3]	**Accuracy**[4]
BlueCup Start	0.28	1214	140.69	100%
BlueCup LEAN	0.31	1835	234.03	100%
BlueCup POKE	0.18	1214	5.04	100%
BlueCup PUSH	0.39	12110	162.75	93%
BlueCup CURIOUS	0.42	2445	156.34	100%
GreenTbox Start	0.18	10122	73.08	93%
GreenTbox LEAN	0.29	3524	64.00	100%
GreenTbox POKE	0.28	1432	73.11	100%
GreenTbox PUSH	0.26	1374	119.29	100%
GreenTbox CUR.	0.33	3678	71.12	100%
RedTeabox Start	0.27	22697	121.05	100%
RedTeabox LEAN	0.45	1426	141.40	100%
RedTeabox POKE	0.35	2090	70.11	87%
RedTeabox PUSH	0.45	2011	53.96	100%
RedTeabox CUR.	0.36	738	114.45	100%
SodaCan Start	0.46	1882	107.79	60%
SodaCan LEAN	0.38	6581	140.62	67%
SodaCan POKE	0.68	3673	3.68	87%
SodaCan PUSH	0.38	1049	223.28	80%
SodaCan CUR.	0.31	16078	213.50	87%

B. *Learning Better Models Trough Interaction*

To improve the visual models for the object detection the robot can choose to perform an action. The images collected during these interactions are used for learning a better detector. The learning experiment is started by placing the object to be learned in a specific position, as in the first experiment. Similar to the first experiment a single input image is collected to generate a preliminary detector. After this the robot selects one out of four possible actions described in Section III.

The robot observes the object while performing the action. As described above, images in fixed intervals are collected. From these new observations – masks are provided by hand for the supervised learning step – together with the image from the start, a new object representation is learned. Depending on the action and its duration the number of images collected varies. The LEAN action, being the shortest, allows only for the collection of 3 new, different images, whereas the CURIOUS action can be used to collect 12 images.

Trials are performed for each combination of one object and an action and separate detectors are trained. Table II shows the learning and performance details of the various detectors. Learning in CGP-IP, like in other evolutionary methods, is non-deterministic, therefore the results shown are based on the best out of five runs. For each of these detectors the fitness during training is reported, as well as, the number of individuals evaluated.

The runtime reported is the average of three runs over the 15 images. These images have been collected separately and build a validation set, as they have not been seen during training. The accuracy reported in the last column refers to this training set. It specifies the number of times the object was detected in those images. From this performance we can conclude the 'right' action to chose. For all of the four objects in this paper the CURIOUS action seems to be the best choice for improving the detector. This is not too surprising, as this action allows to perceive the object from various viewing angles.

Even for objects like the blue cup, where detection should be easy, the action allows for an improvement in the detector. There is no increase in the number of times the object is detected, but there is a signification increase on how much of the object is detected. An increase is also visible in the precision of the segmentation, i.e. the improved detector finds matches that are very close to the contours of the objects. The improvement after the CURIOUS action is visible in Fig. 11. The figure also shows the vanishing of the false positive of the red tea box (a red block in the camera image, visible on the table in the background). The changes of the object representation, after observing the execution of a CURIOUS action, can be seen in the listing shown in Fig. 12 (compared to the code above in Fig. 10).

The detection of the soda can, made of quite reflective

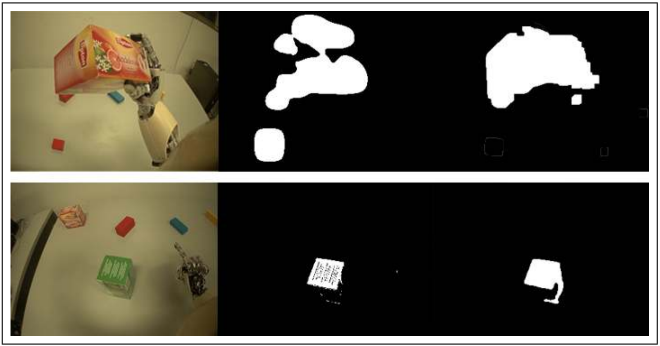


Fig. 11. Comparing the generated segmentation of an input image (left), using the detector at the start — using one image (middle), with the detector trained after the CURIOUS action (right). The first row shows an example from the red tea box the second from the green.

```
icImage BlueCupFilter::RunFilter() {
    icImage node0 = InputImages[4].Exp();
    icImage node5 = InputImages[0];
    icImage node16 = node0.Gabor(-8, 14, 1, 13);
    icImage node17 = InputImages[4].LocalAvg(6);
    icImage node18 = node16.Laplace(5);
    icImage node19 = node5.Sobel(13,9);
    icImage node24 = node17.Erode(5);
    icImage node28 = node19.Min(node18);
    icImage node29 = node28.Min(node24);
    icImage node41 = node29.LocalAvg(7);
    icImage node49 = node41.LocalMax(7);
    icImage out = node49.Threshold(68.03109f);
    return out;
}
```

Fig. 12. The generated C++ code for detecting the blue cup after performing the CURIOUS action. Compared to the detector in Fig. 10, this more complicated model reduces the number of false positives to only 1 in the 15 images of the test set.

aluminium, is not as good as for the other objects. One issue here is the visual similarity of the material with the fingers and other parts of the humanoid's body. This issue is reinforced by the use of simple masks. The fingers are quite often within the masked area, especially during the CURIOUS action. One possibility we are investigating for future research is the use of a disparity map, optical flow or similar approaches to help generating the first masks. Another idea is to use the model learned in one action as the mask for another data collection.

Once the models are learned they can be used on the *iCub* to detect objects in the environment. The system can be fully integrated in the currently available frameworks available in the *iCub* community. Running the detector for both 'eyes' the object's location can be determined [37], [38] to update the robot's world model [39]. By combining all these the *iCub* is able to learn object representations of unseen items, then localise and plan around them.

VI. Conclusions

We showed that our *iCub* humanoid robot is able to create object representations using a machine learning approach to computer vision which we call Cartesian Genetic Programming for Image Processing (CGP-IP). By interacting with the objects, the robot was able to further improve its object detection and identification skills. It did so by collecting observations during action execution. These new observations allowed to learn a better object detection model. An advantage of our model is that it can be directly mapped into human read-able source code and instantly be compiled to run on the real hardware.

Furthermore, we demonstrated that our system can learn to select the right action, i.e. the action leading to the largest improvement in detection. Our experiments show that our CURIOUS action, which contains a pick-up and a variety of rotations to inspect the object, allows for the best improvement. During this action the object can be viewed from almost every angle allowing to build a robust model. This has been observed to be especially useful for visually complex objects, e.g. a tea box. A video of the experiments is attached available on the author's webpage at http://Juxi.net/projects/iCub/#wcci2014.

Although we have a limited number of actions, we showed that a closer integration of actions with vision allows for more information gained from the environment. We believe that in the future a better sensorimotor coordination can be achieved using this approach. We are especially interested in evaluating possible machine learning, e.g. reinforcement learning, techniques to learn the best possible action directly on the robot. We would also like to extend the number of actions and their granularity, as well as, their robustness, e.g. in non-static environments.

References

[1] N. Berthier, R. Clifton, V. Gullapalli, D. McCall, and D. Robin, "Visual information and object size in the control of reaching," *Journal of Motor Behavior*, vol. 28, no. 3, pp. 187–197, 1996.

[2] M. H. Johnson and Y. Munakata, "Processes of change in brain and cognitive development," *Trends in cognitive sciences*, vol. 9, no. 3, pp. 152–158, 2005.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," 2006, pp. 404–417.

[5] D. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proceedings of the International Conference on Computer Vision*. IEEE Computer Society, Sep. 1999.

[6] H. Bischof, W. Schneider, and A. Pinz, "Multispectral classification of landsat-images using neural networks," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 30, no. 3, pp. 482–490, 1992.

[7] H. Kim, E. Murphy-Chutorian, and J. Triesch, "Semi-autonomous learning of objects," *Computer Vision & Pattern Recognition Workshop*, 2006.

[8] Y. Gatsoulis, C. Burbridge, and T. M. McGinnity, "Online unsupervised cumulative learning for life-long robot operation," in *Proc. of the Intl. Conference on Robotics and Biomimetics*, 2011.

[9] G. Rizzolatti and L. Craighero, "The mirror-neuron system," *Annu. Rev. Neurosci.*, vol. 27, pp. 169–192, 2004.

[10] S. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[11] T. Li and Y. Shie, "An incremental learning approach to motion planning with roadmap management," *Journal of Information Science and Engineering*, vol. 23, no. 2, pp. 525–538, 2007.

[12] J. Peters and S. Schaal, "Learning to control in operational space," *The International Journal of Robotics Research*, vol. 27, no. 2, p. 197, 2008.

[13] J. J. Kuffner Jr and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[14] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simultaneous grasp and motion planning: humanoid robot armar-iii," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 2, pp. 43–57, 2012.

[15] G. Metta and P. Fitzpatrick, "Better vision through manipulation," *Adaptive Behavior*, vol. 11, no. 2, pp. 109–128, 2003.

[16] K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann, "Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 2012–2019.

[17] D. Gonzalez-Aguirre, J. Hoch, S. Rohl, T. Asfour, E. Bayro-Corrochano, and R. Dillmann, "Towards shape-based visual object categorization for humanoid robots," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 5226–5232.

[18] O. Stasse, T. Foissotte, D. Larlus, A. Kheddar, K. Yokoi *et al.*, "Treasure hunting for humanoids robot," in *humanoids' 08: International Conference on Humanoids Robots, Workshop on Cognitive Humanoid Vision*, 2008.

[19] D. Schiebener, A. Ude, J. Morimotot, T. Asfour, and R. Dillmann, “Segmentation and learning of unknown objects through physical interaction,” in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 500–506.

[20] N. G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. J. Ijspeert, M. C. Carrozza, and D. G. Caldwell, “iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research,” *Advanced Robotics*, vol. 21, pp. 1151–1175, 2007.

[21] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, “The iCub humanoid robot: An open-systems platform for research in cognitive development,” *Neural Networks*, vol. 23, no. 8-9, pp. 1125–1134, 2010.

[22] J. F. Miller, “An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach,” in *Proc. of the Genetic and Evolutionary Computation Conf.*, 1999, p. 1135.

[23] J. F. Miller, Ed., *Cartesian Genetic Programming*, ser. Natural Computing Series. Springer, 2011.

[24] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[25] S. Handley, “Automatic learning of a detector for alpha-helices in protein sequences via genetic programming,” in *Proc. of the Intl. Conference on Genetic Algorithms*, 1993, pp. 271–278.

[26] M. Oltean, “Evolving evolutionary algorithms using linear genetic programming,” *Evolutionary Computation*, vol. 13, no. 3, p. 387, 2005.

[27] S. Harding, J. Leitner, and J. Schmidhuber, “Cartesian genetic programming for image processing,” in *Genetic Programming Theory and Practice X (in press)*. Springer, 2013.

[28] J. F. Miller and S. L. Smith, “Redundancy and computational efficiency in cartesian genetic programming,” in *IEEE Transactions on Evoluationary Computation*, vol. 10, 2006, pp. 167–174.

[29] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[30] J. Leitner, S. Harding, A. Förster, and J. Schmidhuber, “Mars terrain image classification using cartesian genetic programming,” in *11th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, September 2012.

[31] J. Leitner, S. Harding, M. Frank, A. Forster, and J. Schmidhuber, “Humanoid learns to detect its own hands,” in *IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 1411–1418.

[32] J. Leitner, S. Harding, M. Frank, A. Förster, and J. Schmidhuber, “An integrated, modular framework for computer vision and cognitive robotics research (icVision),” in *Biologically Inspired Cognitive Architectures 2012*, ser. Advances in Intelligent Systems and Computing, 2013, vol. 196, pp. 205–210.

[33] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme.” *Biochimica et Biophysica Acta*, vol. 405, no. 2, pp. 442–451, 1975.

[34] S. Harding, V. Graziano, J. Leitner, and J. Schmidhuber, “Mt-cgp: Mixed type cartesian genetic programming,” in *Proc. of the Genetic and Evolutionary Computation Conf.*, 2012, pp. 751–758.

[35] D. Izzo, M. Ruciński, and F. Biscani, “The generalized island model,” *Parallel Architectures and Bioinspired Algorithms*, pp. 151–169, 2012.

[36] J. Leitner, A. Bernardino, and J. Santos-Victor, “A benchmark on stereo disparity estimation for humanoid robots,” in *Robotica, 8th Conference on Autonomous Robot Systems and Competitions*, 2008.

[37] U. Pattacini, “Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub,” Ph.D. dissertation, RBCS, Italian Institute of Technology, Genova, 2011.

[38] J. Leitner, S. Harding, M. Frank, A. Förster, and J. Schmidhuber, “Transferring spatial perception between robots operating in a shared workspace,” in *Proc. of the Intl. Conference on Intelligent Robots and Systems*, 2012.

[39] J. Leitner, P. Chandrashekhariah, S. Harding, M. Frank, G. Spina, A. Foerster, J. Triesch, and J. Schmidhuber, “Autonomous learning of robust visual object detection on a humanoid,” in *Proc. of the Intl. Conference on Developmental Learning and Epigenetic Robotics (ICDL)*, 2012.