Learning Visual Object Detection and Localisation Using *icVision*

Jürgen Leitner[a], Simon Harding[b], Pramod Chandrashekhariah[c], Mikhail Frank[a], Alexander Förster[a], Jochen Triesch[c], Jürgen Schmidhuber[a]

[a]*Dalle Molle Insitute for Aritificial Intelligence (IDSIA) / SUPSI / Università della Svizzera Italian (USI), Manno-Lugano, Switzerland*
[b]*Machine Intelligence Ltd, South Zeal, United Kingdom*
[c]*Frankfurt Institute of Advanced Studies (FIAS), Frankfurt am Main, Germany*

Abstract

We present a framework combining computer vision and machine learning for the learning of object recognition in humanoid robots. A biologically inspired, bottom-up architecture is introduced to facilitate visual perception and cognitve robotics research. A number of experiments with this *icVision* framework are described.

We showcase both detection and identification in the image plane (2D), using machine learning. In addition we show how a biologically inspired attention mechanism allows for fully autonomous learning. Furthermore localising those objects in 3D space is shown, which in turn can be used to create a model of the environment.

Keywords: humanoid robots, computer vision, perception, cognition, learning, object detection

1. Introduction

Vision is one of the most developed senses in humans. Every day vast amounts of visual information are presented to the human eye, yet we effortlessly are able to extract relevant information from this data stream. In comparison robots have so far not been able to handle large amount of visual data. The detection and identification of objects in the environment are still very hard problems in computer vision and robotics. However being able to make sense of the world, from vision, is becoming increasingly important to endow robots with autonomy and the ability to adapt to a changing environment. These skills are needed to fulfil future robotic scenarios in daily life, coexisting and helping humans. Proposed applications range from household tasks, elderly care, to grocery shopping, etc.

Perception is of critical importance, as the sensory feedback allows to make decisions, trigger certain behaviours, and adapt these to the current situation. The visual feedback enables robots to build up a cognitive mapping between sensory inputs and action outputs, therefore closing the sensorimotor loop. Thus being able to perform actions and adapt to dynamic environments.

On the other hand perception allows to build models of the world around us. The input we get from vision creates a definition of objects and object categories in our mind, which can be further refined using multiple other senses, such as, touch, taste, smell, etc. It furthermore enables the grounding of language and terminology.

We present a visual perception system with the aim of providing a framework for perception in robots. The system is based on models ofhuman perception and has been used to learn how to detect and identify objects. In the future we hope our system will help to create more autonomous and adaptive behaviours in robots.

2. Background and Related Work

Vision and the visual system are the focus of much research in psychology, cognitive science, neuroscience and biology. A major issue in visual perception is that what individuals 'see' is not just a simple translation of input stimuli (compare *optical illusions*). The research of Marr in the 1970s led to a theory of vision using different levels of abstraction Marr (1982). He described human vision as processing inputs, stemming from a two-dimensional visual array (on the retina), to build a three-dimensional description of the world as output. For this he defines three levels: a 2D (or primal) sketch of the scene (using feature extraction), a sketch of the scene using textures to provide more information, and finally a 3D model.

In humans hierarchies in the visual cortex (V1, V2, V3, ...), so called, visual pathways, play an important role (Hubel et al., 1995) in object detection. Furthermore fMRI observations have

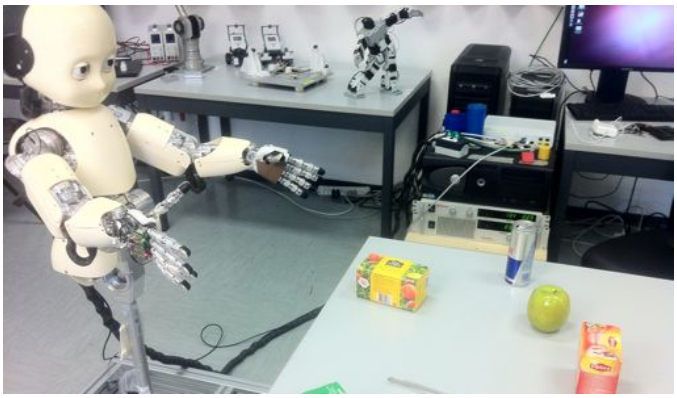

Figure 1: Our research platform, the *iCub* humanoid robot, in front of a table with various objects to be detected and identified.

found extensive activity in the visual cortex during a complicated pattern recognition task (Ress and Heeger, 2003).

There exists a vast body of work on all aspects of robotic vision and image processing (Gonzalez and Richard, 2002), using both classical and machine learning approaches. In computer vision 'image segmentation' is one of the most common approaches to perform object detection. In this process the foreground and background of the image are separated. However, image segmentation is especially challenging when the objects are static, occluded or the background model is not known.

An often used concept in computer vision are features. A feature is a point in the image that is interesting and described in a (brief) comprehensive representation in a repeatable way. Local image features (also called local descriptors) are the entities that capture the information of the region around identified feature points. Local image features have proven to be resistant to occlusions, local deformations of the objects, variation in illumination conditions, and background clutter (Mikolajczyk and Schmid, 2003; Agarwal and Roth, 2002). Various methods of detecting features in literature, such as, SIFT (Lowe, 1999) and SURF (Bay et al., 2006), as well as, the FAST corner detection algorithm (Rosten et al., 2010).

General frameworks for object detection have previously been developed. Early work showed that these detection problems can be 'learned' and a representation of the objects can be built (Papageorgiou et al., 1998). Serre et al. (2007), for example, developed a system, based on how humans perform this task, that would detect objects in scenes based on template matching and max-pooling operations. In Bianchini et al. (2004) a neural model for object detection was introduced. It required a pre-processing step to provide the recurrent network with a graph-based representation of an image combining spatial and visual features.

Our experimental platform is the *iCub* humanoid robot (Tsagarakis et al., 2007), an open-system robotic platform, providing a 41 degree-of-freedom (DOF) upper-body, comprising two arms, a head and a torso (see Figure 1). The *iCub* is generally considered an interesting experimental platform for cognitive and sensorimotor development and embodied Artificial Intelligence (Metta et al., 2010) research, with a focus on object manipulation. To enable object manipulation first the object has to be detected and located using the available sensors. The robot has to rely, similarly to human perception, on a visual system. Two cameras are mounted in the *iCub*'s head to provide stereo vision. There are currently some efforts to implement a fully integrated cognitive architecture for the iCub in YARP (Vernon et al., 2007), yet the application is still limited to very early stage research.

An important area of research in robotics is trying to find an approach to object localisation that is robust enough to be deployed on a real humanoid robot. The term 'Stereo Vision' describes the extraction of 3D position information out of multiple digital images of the same scene and is similar to the biological process of stereopsis in humans. Its basic principle is the comparison of images taken from different viewpoints. To obtain a distance measure the relative displacement of a pixel between the two images is used (Hartley and Zisserman, 2000). While these approaches, based on projective geometry, have been proven effective under carefully controlled experimental circumstances, they are not easily transferred to robotics applications. As the cameras are mounted in the head of the robot the method for localisation must be able to cope with motion to work even while the robot is controlling its gaze and upper body for reaching. Various local reference frames (e.g. the left and right eye frames) need to be converted into the reference frame in which we seek to express object locations (Figure 2). For the *iCub* platform several approaches have previously been developed. Pattacini (2011) developed the 'cartesianController' module, which provides basic 3D position estimation functionality and gaze control. This module works well on the simulated *iCub*, however it is not fully supported and functional on the hardware platform, and therefore does not perform well. The most accurate currently available localisation module for the *iCub* exists in the 'stereoVision' module. It provides accuracy in the range of a few centimetres.

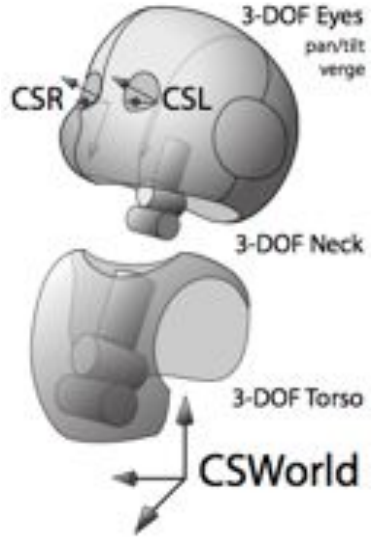

Figure 2: The object localisation problem, illustrated according to the kinematics of the iCub robot, is to process images from cameras located at the origin of *CSL* and *CSR* to express the position of objects with respect to *CSWorld*.

3. The *icVision* Framework

Research on perception has been an active component of developing artificial vision systems in robotics for research and industry. Our humanoid robot should be able to learn how to perceive and detect objects, in a manner similar to humans. It should have the ability to develop a representation that allows it to detect this object again and again, even when the lighting conditions change, e.g., during the course of a day. This is a necessary prerequisite to enable adaptive, autonomous behaviours based on visual feedback. Our goal is to apply a combination of robot learning approaches, artificial intelligence and machine learning techniques, with computer vision, to enable a variety of proposed tasks for robots.

Our biologically-inspired architecture, which is in line with the description of the human visual system by Marr, was developed to support current and future research in cognitive robotics. It processes the visual inputs received by the cameras and builds (internal) representations of objects. It facilitates the 3D localisation of the detected objects in the 2D image plane and provides this information to other systems, such as, e.g. a motion planner. The system consists of distributed YARP modules interacting with the *iCub* hardware and each other. YARP (Metta et al., 2006) is a popular open source robotics middleware, comparable to ROS (Quigley et al., 2009). It allows easy, distributed access to the sensors and actuators of the *iCub* humanoid robot, as well as, to other software modules. It allows to create distributed systems of loosely coupled modules and provides standardised interfaces. As with YARP modules, *icVision* can easily be used with a variety of robots. Figure 3 sketch our *icVision* architecture.

Special focus is put on the object detection in the received input images. Specialised modules, containing a specific model, are used to detect distinct patterns or objects. These specialised modules can be connected and form pathways to perform, e.g., object detection, similarly to the hierarchies in the visual cortex.

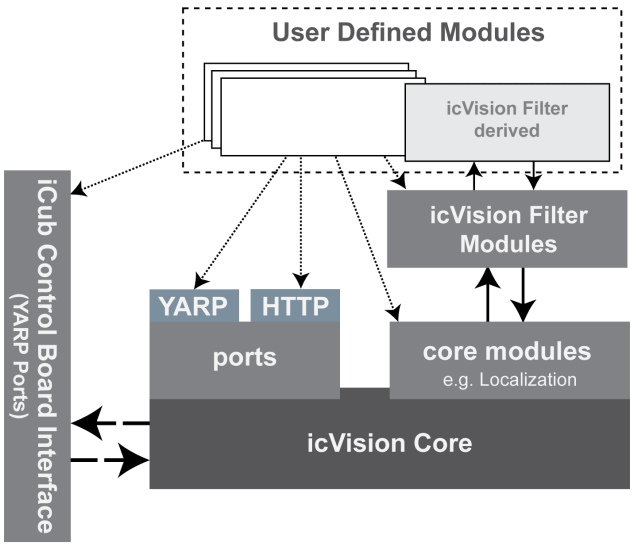

Figure 3: The architecture of our *icVision* framework.

3.1. Core Modules (icVision ***Core***)

The main module, the *icVision* **Core**, handles the connection with the hardware and provides housekeeping functionality (e.g., GUI, module start/stop). This core module also stores information about the state of the robot, its YARP connection and other relevant data. Implemented modules for the object detection (aka filters) are connected to this core module and can request this, as well as, provide information to be made available for other modules.

The core also contains other modules that are used for special purposes. One such module, e.g., provides information about 3D location of objects detected. Another module controls the gaze of the robot, based on detection information, and, e.g., allows simple switching of the gaze from one object to another. These are reachable via standardised interfaces allowing for easy swapping and reuse of modules and extending the functionality. In the experiments section we learn a new localisation model which we can then easily use instead of the *iCub* provided 'cartesianController'. There are currently multiple biologically- and neuroscience-inspired modules implemented in *icVision*:

- a module estimating the 3D location of an object perceived in both cameras (see Section 3.3 and Section 4.4)
- a module matching the inputs from the left and right camera based on features detected (Figure 3)
- a stereo camera disparity map providing rough distance estimations based on the two camera images (Figure 5)

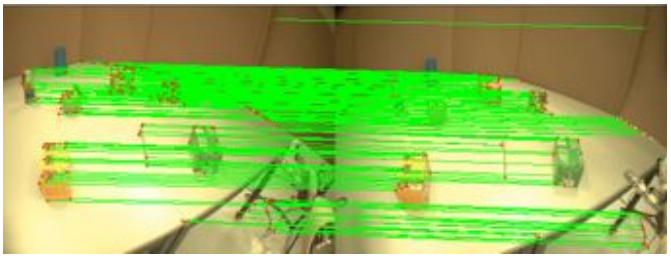

Figure 4: The matching of the camera images coming from both eyes. The feature extracted and matched in both images are connected by a green line.

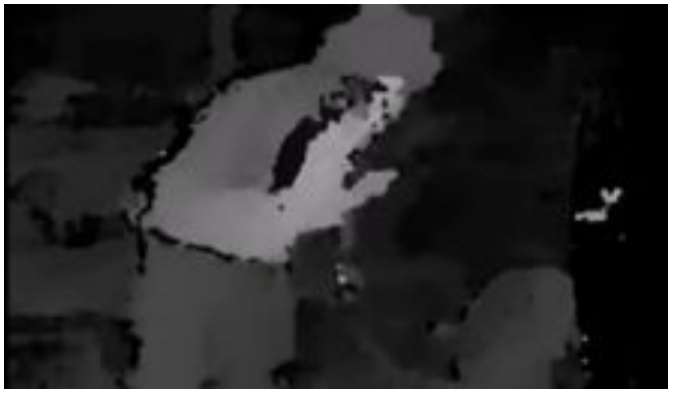

Figure 5: A disparity map generated by the *iCub* 'stereoVision' module. Bright areas are closer to the cameras then dark ones.

- a saliency map providing information about which points in the images are the most salient based on neuromorphic models (Figure 10)

3.2. *Detecting Objects (*icVision ***Filter****)*

The first thing done by the human visual system, and investigated by us, is the segmentation (detection) in the visual space. The *icVision* filter modules, which relate to Marr's first and second level, provide object detection in the images. Multiple filter modules can run in parallel. Each of these modules has a standardised interface and a coding template is provided to perform quick testing of new filters.

As input each filter module is provided with several separate single channel images. These are: the camera image in grayscale, the image's red, green and blue (RGB) channel, as well as, its hue, saturation and brightness value (HSV) channels. The result of the run filter is a binary segmentation for a specific object. Figure 6 shows a teabox being tracked by the *iCub* in real-time using a learned filter. In Figure 7 the binary output can be seen as a step in the 3D localisation.

The framework provides an easy-to-use library of codes. It allows to quickly prototype simple filters with just a few lines of code (see Listing 1). Furthermore testing functionality is available allowing the vision system to run without having YARP installed or being connected to the real robot. Using machine learning, more complicated filters can be generated automatically instead of engineered. We apply Cartesian Genetic Programming (CGP) (Miller, 1999, 2011) to provide automatic generation of computer programs making use of the functionality integrated in the OpenCV image processing library (Bradski, 2000), therefore incorporating domain knowledge. It provides an effective method to learn how to detect new objects robustly, if the training set is chosen correctly (see Section 4.2).

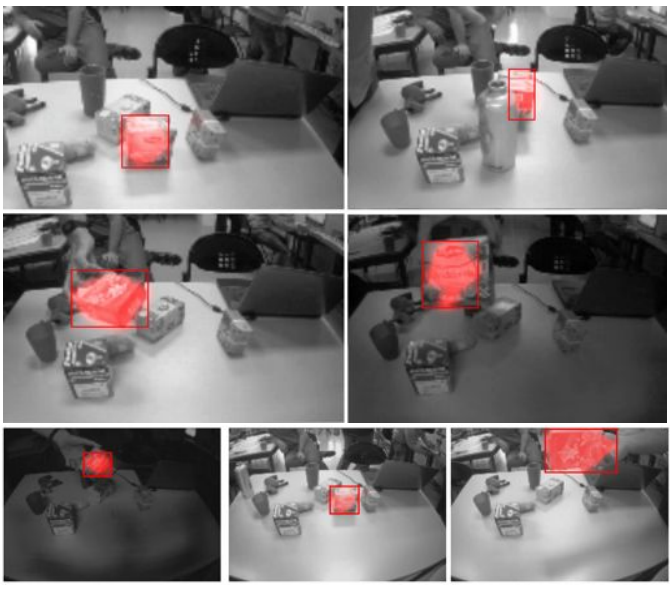

Figure 6: The detection of a teabox in changing lighting condition performed by a learned filter. The binary output of the filter is used as red overlay.

3.3. *Locating Objects (*icVision ***3D****)*

To enable the robot to interact with the environment it is important to localise the objects first. Developing an approach to perform robust localisation to be deployed on a real humanoid robot is necessary to provide the necessary inputs for on-line motion planning, reaching, and object manipulation.

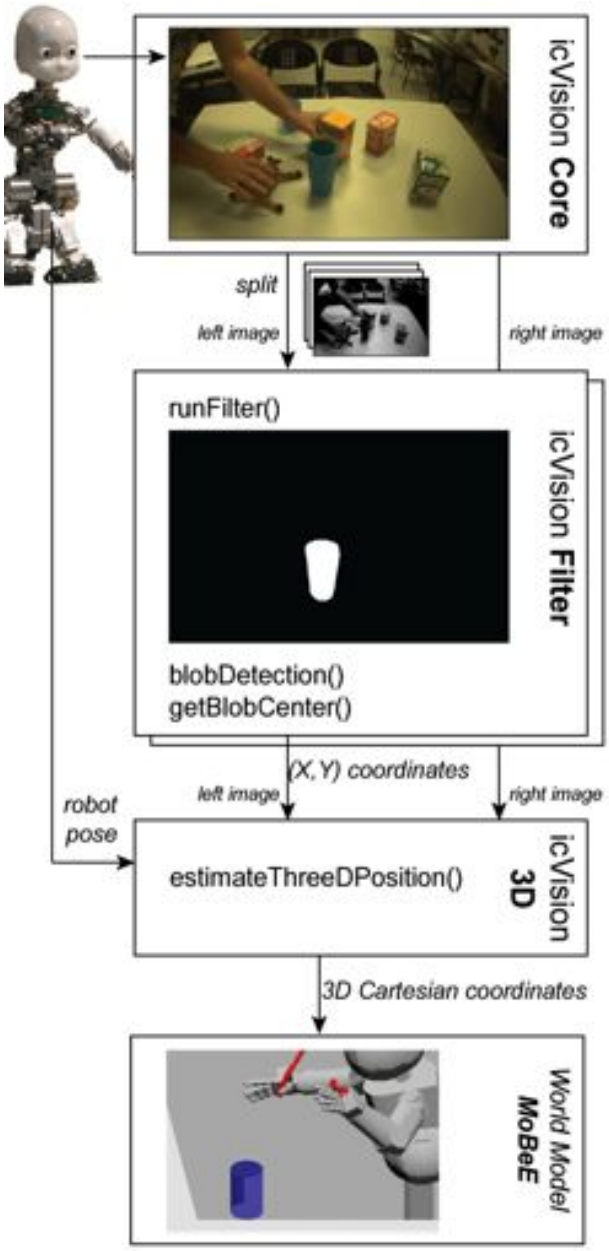

Figure 7: The 3D location estimation works the following: At first the camera images are acquired from the hardware via YARP. The images are converted into grayscale, as well as, split into RGB/HSV channels and distributed to all active *icVision* filters. Each filter then processes the images received using OpenCV functions. The output of this is a binary image, segmenting the object to be localised. A blob detection algorithm is run on these binary images to find the (centre) location of the detected object in the image frame. The position of the object in both the right and left camera images is sent to the 3D localisation module, where together with the robots pose, i.e. the joint encoder valiues, a 3D location estimation is generated. As the last step the localised object is then placed in the existing world model.

The *icVision* 3D localisation module is one of the modules provided by the core framework and corresponds to Marr's last level, a spatial model of the environment. It allows for conversion between camera image coordinates and 3D coordinates in the robot reference frame. Using the objects location in the cameras (provided by an *icVision* Filter module) and pose information from the hardware, this module calculates where the object is in the world. This information is then used to update the world model. Figure 7 describes the full 3D location estimation process in *icVision*, starting with the camera images received from the robot and ending with the localised object being placed in our MoBeE world model (Frank et al., 2012).

The *icVision* 3D localisation module provides an easy way of swapping between various localisation implementations, including the exiting ones for the *iCub*.

4. Experiments

Our *icVision* framework has already successfully been used in our research. Hereafter we present a short list of experiments and use cases.

4.1. Simple Filters

As mentioned above *icVision* allows for multiple filters, i.e. pattern or object detectors, to run at the same time. Each of these filters is provided with the input images split into channels (grayscale, RGB, HSV) and each of these can be used in the filter to base a segmentation on. The framework provides an easy-to-use abstraction allowing to create filters with just a few lines of code (see Listing 1). Here the only the blue channel of the input image is used and then thresholded. This binary segmentation allows detects areas in the image where the blue content is rather high. While this example might not be very useful, it shows that in just a few lines of C# code a filter can be engineered that can be run on the real hardware.

4.2. Learning Filters

Instead of manually developing filters for each object, we chose to use a machine learning approach to train specific filters for certain objects. This was done by applying Genetic Programming (GP). GP is a search technique inspired by concepts from Darwinian evolution (Koza, 1992). It can be used to generate formulae or whole programs to solve problems in many domains, including image processing.

```
icImage BlueFilter::runFilter() {
  icImage blueCh = InputImages[BLUE_CH];
  icImage out = blueCh.threshold(150.0f);
  return out;
}
```

Listing 1: A simple filter implementation to detect the areas with the highest blue content in the input image.

Background. We are using Cartesian Genetic Programming (CGP), in which genotypes encode programs in partially connected feed forward graphs (Miller, 1999, 2011). For each node in the genome there is a vertex, represented by a function, and a description of the nodes from where the incoming edges are attached (Figure 8). CGP offers some nice features, for instance, not all of the nodes of a solution need to be connected to the output node of the program. As a result there are 'neutral' nodes in the representation, having no effect on the output. This has been shown to be very useful (Miller and Smith, 2006) in the evolutionary process. Also, because the genotype encodes a graph, nodes can be reused, which makes the representation distinct from a classically tree-based GP representation. These programs (also referred to as filters) perform object identification based on binary image segmentation.

Our implementation 'CGP for Image Processing' (CGP-IP) draws inspiration from much of the previous work in the field, see e.g. (Sekanina et al., 2011). Its main difference to previous implementation is that it encompasses domain knowledge, i.e. it allows for the automatic generation of computer programs using a large subset of the OpenCV image processing library (Bradski, 2000). With over 60 unique functions[1], this set is considerably larger than those typically used with GP. Such a complex representation would normally be considered too large to efficiently evolve, however this does not appear to hinder evolution in our experiments. The efficacy of this approach was shown for several different domains (including medical imaging, terrain classification and defect detection in industrial application) (Harding et al., 2013; Leitner et al., 2012a, 2013).

The primitive operators work on entire images. Working at the functional level allows for the easy inclusion of many standard image processing operations, for example, Gabor filtering. In CGP-IP, individuals are evaluated at the rate of 100s per second on a single core. This makes it both efficient to evolve solutions, but also means that the evolved programs will run quickly when deployed. The output of a CGP-IP individual is based on the image at the final node in the genotype. This is thresholded using an additional parameter (real number) included in the genotype.

CGP-IP needs a number of parameters encoded in *each node*. These (listed in Table 1) are needed because often the functions

[1] A complete list of functions available in CGP-IP can be found in (Harding et al., 2013).

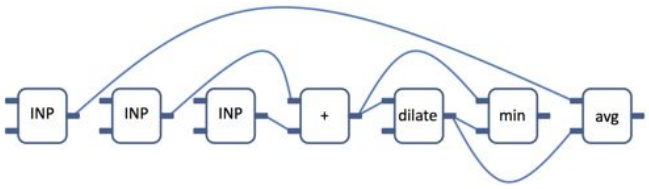

Figure 8: In this example illustration of a CGP-IP genotype, the first three nodes obtain the components from the current test image (e.g. grey scale version, red and green channels). The fourth node adds the green and red images together. This is then dilated by the fifth node. The sixth node is not referenced by any node connected to the output (i.e. it is neutral), and is therefore ignored. The final node takes the average of the fifth node and the grey scale input.

Table 1: The additional parameters encoded at each node in CGP-IP.

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	$[-16, +16]$
Gabor Filter Frequency	Int	$[0, 16]$
Gabor Filter Orientation	Int	$[-8, +8]$

used require additional parameters, with specific requirements as to their type and range. CGP-IP does not require many *global* parameters to be set:

- Graph length (i.e. nodes in the genotype), set to 50.
- Mutation rate, 10% of all genes in the graph are mutated at offspring generation.
- Number of islands, set to 8.
- Number of individuals per island, set to 5 for a typical 1+4 evolutionary strategy.
- Synchronisation interval between islands, set to 10.

All parameters are kept constant throughout our experiments. It is important to note that these parameters have not been optimised other than by casual experimentation. It may be possible to improve the performance by careful selection.

For this application the thresholded output image of an individual is compared to a target image using the MCC (Matthews, 1975), which has previously been observed to be useful when solving classification problems using CGP (Harding et al., 2012). The MCC is calculated based on the count of the true positives, false positives, true negatives, and false negatives. A coefficient of 0 indicates that the classifier is working no better than chance. 1 represents a perfect classification, a score of -1 indicates the classifier, though perfect, has inverted the output classes. Therefore, an individual's fitness $f = 1 - |MCC|$ with values closer to 0 being more fit.

In line with the *icVision* filter modules CGP-IP operates on single channel images. Each available channel (grayscale, RGB and HSV) is presented as an input to CGP-IP, and evolution/-mutation selects the inputs to be used. This approach allows for increased performance and flexibility.

Filter Generation. Executing the CGP genotype is a straightforward process. First, the active nodes are identified. This is done recursively, starting at the output node, and following the connections used to provide the inputs for that node. After that genotype can be matched to the phenotype, which in turn can be executed to obtain a fitness score.

Our implementation of CGP-IP generates human readable C# or C++ code based on OpenCV (see Listing 2). CGP, though offering bloat free evolution, appears to leave redundant nodes in the evolved program. A pruning step was therefore implemented to optimise the final code. It is typically found that this process reduces the number of used instructions, and hence reduces the execution time of the evolved program.

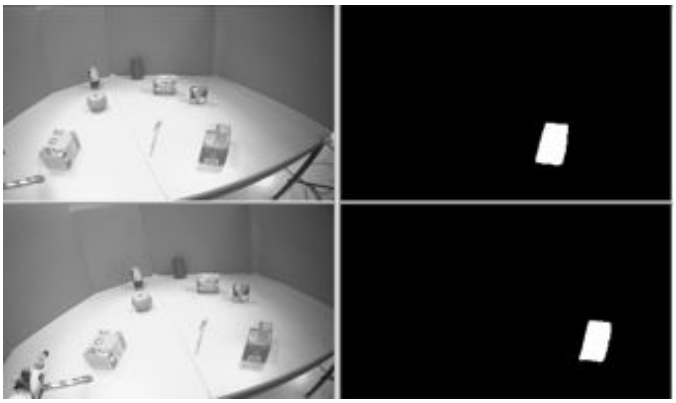

Figure 9: A training set to to learn the a filter for a green teabox. On the right the hand-labelled binary segmentation of the training set is shown.

Supervised Learning. A handful of examples, in which the object of interest has been correctly segmented, are used as a training set for CGP-IP. The selection of this training set is of importance for the capabilities of the found solution. Figure 9 shows the very simple training set for a green teabox. Only two hand-segmented images are used to generate a filter using CGP-IP. This small training does not contain much variation. If chosen correctly though, the resulting programs are very robust to changing light conditions, as shown in Figure 6.[2] The generated code is human readable and similar to Listing 1 and Listing 2.

On a single core of a modern desktop PC, the evolved programs for objects like teaboxes and kids toys, run within 2ms. As these are largely linear in form, the memory requirements are low. Speed and simplicity of processing is important in many embedded systems, making this approach suitable for implementation in constrained computing environments.

We are currently using this same technique to learn filters for the robot's fingers (Leitner et al., 2013) to perform research in the development of sensorimotor control on the *iCub*.

4.3. Autonomous Learning of Filters

This experiment shows a more autonomous acquisition of object representations using *icVision*. We developed a visual system that actively explore the unfamiliar environment and provides the robot with the ability to learn visual representation for objects in the scene. A combination of different techniques allows the robot to perform this learning task:

- To allow for autonomous generation of these unique identifiers a biologically inspired scene exploration approach is used.
- A feature based approach is used to create segmented images around a salient point.

[2] A video can be found at `http://Juxi.net/projects/icVision/`

- With this pre-segmentation we are able to provide the needed inputs to our CGP learner for building robust filter using the same approach as in the previous experiment.

Scene Exploration. Our aim is to let the robot explore the scene and learn to detect and identify objects in an autonomous fashion. An attention mechanism coupled with a stereo segmentation scheme avoids manual supervision for segmenting the objects in the scene required for learning with CGP-IP.

A neuromorphic model, based on Itti et al. (1998), was used to identify the elements of a visual scene that are salient, i.e., are most likely to attract the attention of an observer. The various visual stimulus over the visual scene are topographically encoded in the form of a single 2D saliency map using a bottom-up control strategy. The saliency maps from both the eyes are used by the decision module to select the single most salient point in the scene (Figure 10) and gaze upon it before segmentation can begin. As this technique is quite robust to variations, the same point will be selected at every run. To avoid this and actually be able to detect multiple objects, a temporary inhabitation of the saliency map around the winning location is applied. Recently visited location will be inhibited, in decreasing order, to find the most salient objects in the scene.

Feature-based Pre-Segmentation. The feature detection is performed by the FAST corner detection algorithm, as it provides both fast and high quality features (Rosten et al., 2010). The local descriptor is calculated by Gabor wavelets, of these kernel

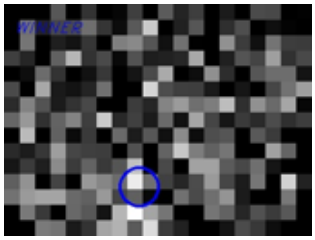

Figure 10: A saliency map provided by an *icVision* module, highlighting the most salient point (blue circle).

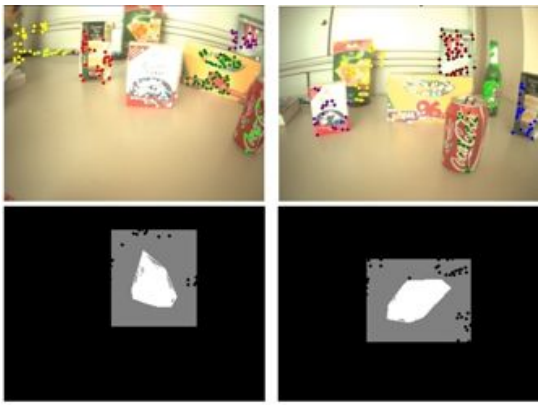

Figure 11: Two examples for the initial segmentation using features. Top row: features detected on the input images are clustered, if a matching exists. Bottom row: initial segmentation based on the features detected.

```
icImage GreenTeaBoxDetector::runFilter() {
  icImage node0 = InputImages[6];
  icImage node1 = InputImages[1];
  icImage node2 = node0.absdiff(node1);
  icImage node5 = node2.SmoothBilataral(11);
  icImage node12 = InputImages[0];
  icImage node16 = node12.Sqrt();
  icImage node33 = node16.erode(6);
  icImage node34 = node33.log();
  icImage node36 = node34.min(node5);
  icImage node49 = node36.Normalize();
  icImage out = node49.threshold(230.7218f);
  return out;
}
```

Listing 2: The generated code from CGP-IP for detecting the teabox. `icImage` is a wrapper class to allow portability within our framework.

features, having the shape of plane waves restricted by a Gaussian envelope function (Wiskott et al., 1997). The choice is also motivated by the fact that they have a similar shape as the receptive field of simple cells found in the visual cortex of vertebrate animals (Jones and Palmer, 1987; Pollen and Ronner, 1981). The extracted features were then matched between the two camera images. This is required as we need the stereo information to segment the object from the background. Correspondences are found by comparing the local descriptors around each feature using normalised Euclidean inner product. Each feature in the left image is associated with the best matching interest point in the right image. Figure 3 shows the result of such a matching during our experiments as provided by an *icVision* module.

We cluster the matched interest points, using a greedy scheme, into different groups according to their image location and disparity. Figure 11 shows two examples of this clustering, with cluster of dots of different colours belong to different objects. The object at the centre of gaze is segmented (white dots) from other objects using spatial location and depth. As seen this feature segmentation only provides a rough estimate for the correct binary segmentation. The minimum bounding rectangle enclosing the features, with additional 20 pixel margin on every side is used to build a training set for CGP-IP.

Training For Robust Image Segmentation. Extracting features that allow for robust object detection is not an easy task. Figure 12 shows the same teabox and the FAST features extracted in two frames roughly 5 seconds apart. The features are quite different, making it hard to know whether this object is the same in both images. To overcome this we use the features detected to build a training set for our CGP-IP approach. CGP-IP quickly learns, in about 1000 evaluations, how to segment the green teabox. The evolved program detects the object with the same filter in both images allowing for unique identification (Figure 13) . The resulting C++ code, shown in Listing 2, performs the detection on the hardware in real-time, well below the frame update of our robot's cameras. More information can be found in (Leitner et al., 2012d).

Once the training is complete, the learned identifiers are

Figure 12: The FAST features (in white) found of an object in two images.

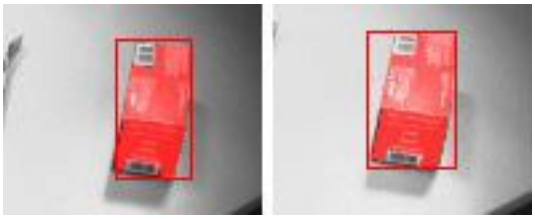

Figure 13: The teabox as detected with a learned filter using CGP. The binary output of the filter is used as a red overlay.

tested by placing the objects in different locations on the table. The tests included different poses and varying lighting conditions. The learned programs can also be further used to build a model of the environment, by integrating the system with existing software (Frank et al., 2012) (Figure 14).

4.4. Learning Localisation

To create a world model, such as the one depicted in Figure 14, a 3D localisation of the objects is required. While we could use the *iCub* provided methods, we wanted to see if we could use machine learning to predict these positions for us.

We used a red block, placed in the shared workspace, by a Katana robotic arm to teach the *iCub* how to perceive the location of the objects it sees. The Katana has a very precise kinematic model and allows to place the object with millimetre accuracy. An *icVision* filter provides the position in the images (Figure 15), with the *icVision* core yielding the robot's pose. While the two robots move around the data is collected, generating a training set for a supervised machine learning approach.

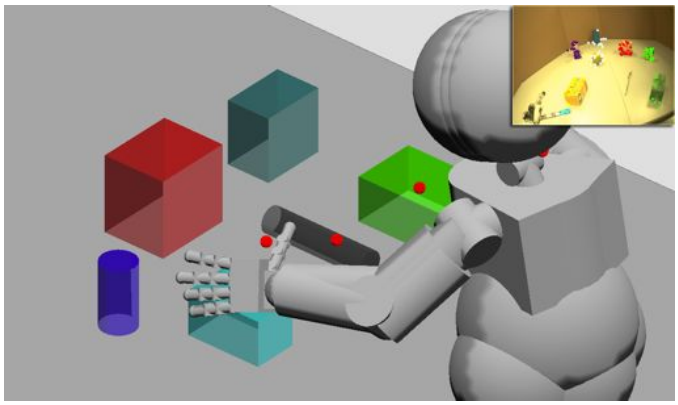

Figure 14: The objects placed in the MoBeE (Frank et al., 2012) model of the environment using the learned filters. Inset shows the actual camera image.

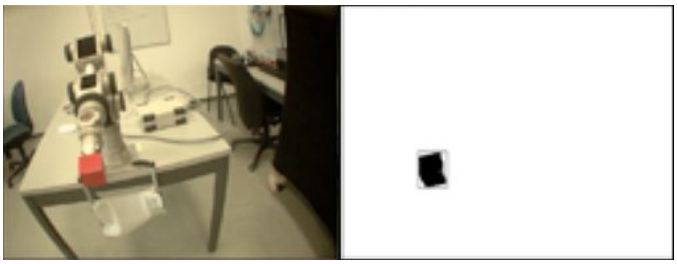

Figure 15: The red object placed at a precise position by the Katana robot in front of the *iCub*. On the right side is the binary segmentation providing the position of the object in the image.

The robot pose and the 2D position outputs provided by *icVision* are used to train artificial neural networks (ANN) to estimate the object's 3D Cartesian location (see Leitner et al. (2012b) for more information about the learning setup). We show that satisfactory results can be obtained for localisation (Leitner et al., 2012c). Furthermore, we demonstrate that this task can be accomplished safely using collision avoidance software to prevent collisions between multiple robots in the same workspace (Frank et al., 2012). A video of an object detected by an icVision filter and then placed into a MoBeE world model using the learned 3D localisation can be found online at `http://Juxi.net/projects/icVision/`.

5. Conclusions

We combine the current machine learning and computer vision research to build a biologically-inspired, cognitive framework for the *iCub* humanoid robot. The developed *icVision* framework facilitates the autonomous development of new object detectors. These cognitive perception skills are seen as the foundation to developmental mechanisms, such as as sensorimotor coordination, intrinsic motivation and hierarchical learning, which are investigated on the robotic platform.

The reason for the focus on vision is twofold, firstly the limited sensing capabilities of the robotic platform and secondly, vision is the most important sense for humans. As we use a humanoid robot investigating how humans do this tasks of perception, detection and tracking of objects is of interest.

We showed how filters for specific objects can be learned in a supervised and an autonomous fashion. We also showed how a spatial perception model can be learned. The whole system together has already been extensively used in our research. The learned *icVision* filter modules, provide binary classification, extracted using machine learning, for specific and complex objects, such as, tea boxes, soda cans and bottles, even under changing lightning conditions.

Furthermore together with a reactive controller to enable the *iCub* to autonomously re-plan a motion to avoid an object it sees Frank et al. (2012). The object was placed into the world model purely from vision, it is able to update the position of the object in real-time, even while the robot is moving. Learning to grasp and basic hand-eye coordination are the areas of research this framework is currently applied.

Acknowledgment

This research has been funded in parts by the European Community's Seventh Framework Programme FP7/2007– 2013, under grant agreement no. FP7-ICT-IP-231722 "IM-CLeVeR".

References

Agarwal, S., Roth, D., 2002. Learning a sparse representation for object detection. In: ECCV. pp. 113–130.

Bay, H., Tuytelaars, T., Van Gool, L., 2006. Surf: Speeded up robust features. Computer Vision – ECCV 2006 3951, 404–417.

Bianchini, M., Maggini, M., Sarti, L., Scarselli, F., 2004. Recursive neural networks for object detection. In: IEEE International Joint Conference on Neural Networks. Vol. 3. pp. 1911–1915 vol.3.

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools.

Frank, M., et al., 2012. The modular behavioral environment for humanoids and other robots (MoBeE). In: International Conference on Informatics in Control, Automation and Robotics (ICINCO).

Gonzalez, R., Richard, E. W., 2002. Digital image processing.

Harding, S., Graziano, V., Leitner, J., Schmidhuber, J., 2012. Mt-cgp: Mixed type cartesian genetic programming. In: Genetic and Evolutionary Computation Conference (GECCO).

Harding, S., Leitner, J., Schmidhuber, J., 2013. Cartesian genetic programming for image processing. In: Genetic Programming Theory and Practice X. Springer, in press.

Hartley, R., Zisserman, A., 2000. Multiple view geometry in computer vision. Cambridge University Press.

Hubel, D., Wensveen, J., Wick, B., 1995. Eye, brain, and vision. Scientific American Library.

Itti, L., Koch, C., Niebur, E., 1998. A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (11), 1254–1259.

Jones, J. P., Palmer, L. A., 1987. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. J. of Neurophysiology 58, 1233—1258.

Koza, J. R., 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA.

Leitner, J., Harding, S., Förster, A., Schmidhuber, J., 2012a. Mars terrain image classification using cartesian genetic programming. In: International Symposium on Artificial Intelligence, Robotics & Automation in Space.

Leitner, J., Harding, S., Förster, A., Schmidhuber, J., 2013. Humanoid learns to detect its own hands (submitted). In: IEEE Congress on Evolutionary Computation (CEC).

Leitner, J., Harding, S., Frank, M., Förster, A., Schmidhuber, J., 2012b. Learning spatial object localisation from vision on a humanoid robot. International Journal of Advanced Robotics Systems 9 (1).

Leitner, J., Harding, S., Frank, M., Förster, A., Schmidhuber, J., 2012c. Transferring spatial perception between robots operating in a shared workspace. In: IEEE/RSJ International Conference on Intelligent Robots and Systems.

Leitner, J., et al., 2012d. Autonomous learning of robust visual object detection on a humanoid. In: IEEE International Conference on Developmental Learning and Epigenetic Robotics (ICDL).

Lowe, D., Sep. 1999. Object Recognition from Local Scale-Invariant Features. In: International Conference on Computer Vision (ICCV).

Marr, D., 1982. Vision: A Computational Approach. Freeman & Co., San Francisco.

Matthews, B. W., 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta 405 (2), 442–451.

Metta, G., Fitzpatrick, P., Natale, L., 2006. YARP: Yet Another Robot Platform. International Journal of Advanced Robotics Systems 3 (1).

Metta, G., et al., Oct. 2010. The iCub humanoid robot: An open-systems platform for research in cognitive development. Neural Networks 23 (8-9), 1125–1134.

Mikolajczyk, K., Schmid, C., 2003. A performance evaluation of local descriptors. In: IEEE Conference on Computer Vision and Pattern Recognition.

Miller, J., 1999. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In: Genetic and Evolutionary Computation Conference (GECCO).

Miller, J., 2011. Cartesian genetic programming. Cartesian Genetic Programming, 17–34.

Miller, J., Smith, S., 2006. Redundancy and computational efficiency in cartesian genetic programming. In: IEEE Transactions on Evoluationary Computation. Vol. 10. pp. 167–174.

Papageorgiou, C. P., Oren, M., Poggio, T., 1998. A general framework for object detection. In: Computer Vision, 1998. Sixth International Conference on. IEEE, pp. 555–562.

Pattacini, U., 2011. Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub. Ph.D. thesis, RBCS, Italian Institute of Technology, Genova.

Pollen, D. A., Ronner, S. F., 1981. Phase relationship between adjacent simple cells in the visual cortex. Science 212, 1409–1411.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A., 2009. ROS: an open-source Robot Operating System. In: International Conference on Robotics and Automation, Open-Source Software workshop.

Ress, D., Heeger, D. J., 2003. Neuronal correlates of perception in early visual cortex. Nature neuroscience 6 (4), 414–420.

Rosten, E., Porter, R., Drummond, T., 2010. Faster and better: A machine learning approach to corner detection. IEEE Trans. Pattern Anal. Mach. Intell 32 (1), 105–119.

Sekanina, L., Harding, S. L., Banzhaf, W., Kowaliw, T., 2011. Image processing and CGP. In: Miller, J. F. (Ed.), Cartesian Genetic Programming. Natural Computing Series. Springer, Ch. 6, pp. 181–215.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., Poggio, T., 2007. Robust object recognition with cortex-like mechanisms. Pattern Analysis and Machine Intelligence, IEEE Transactions on 29 (3), 411–426.

Tsagarakis, N. G., et al., Jan. 2007. iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research. Advanced Robotics 21, 1151–1175.

Vernon, D., Metta, G., Sandini, G., 2007. The iCub Cognitive Architecture: Interactive Development in a Humanoid Robot. In: Proc. of the International Conference on Development and Learning (ICDL). pp. 122–127.

Wiskott, L., Fellous, J., Krüger, N., v.d. Malsburg, C., 1997. Face recognition by elastic bunch graph matching. IEEE Trans. Pattern Anal. Mach. Intell., 775–779.